

Enhanced Networking on IBM z/VSE

Use LFP on Z/VM and in LPAR for
selected applications

Experience the benefits
of OpenSSL

Learn how to use IPv6
on z/VSE



Joerg Schmidbauer
Jeffrey Barnard
Ingo Franzki
Karsten Graul
Don Stoevers
Rene Trumpp

Redbooks



International Technical Support Organization

Enhanced Networking on IBM z/VSE

December 2014

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

Second Edition (December 2014)

This edition (SG24-8091-01) applies to z/VSE V5 R2.

© Copyright International Business Machines Corporation 2014. All rights reserved.

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

Contents

Notices	ix
Trademarks	x
Preface	xi
The team who wrote this book	xi
Now you can become a published author, too!	xii
Comments welcome	xii
Stay connected to IBM Redbooks	xiii
IBM Redbooks promotions	xv
Summary of changes	xvii
December 2014, Second Edition	xvii
Chapter 1. Networking options overview	1
1.1 Overview	2
1.2 Hardware options	2
1.2.1 OSA-Express	2
1.2.2 OSA-Integrated Console Controller	3
1.2.3 OSA-Express in QDIO mode	4
1.2.4 OSA-Express	5
1.2.5 OSA for NCP support	5
1.2.6 Intra-Ensemble Data Network support	6
1.2.7 OSA-Express multi-port support	7
1.2.8 Using VTAM (SNA) and TCP/IP (non-QDIO) parallel on the same CHPID	8
1.2.9 HiperSockets (IQD)	9
1.2.10 Virtual local area network	10
1.2.11 Shared OSA adapter versus HiperSockets	11
1.2.12 Using HiperSockets to communicate with Linux on System z	11
1.2.13 QDIO buffer configuration	12
1.2.14 Virtual OSA devices and VMAC	14
1.2.15 OSAX Hotswap support	15
1.3 Software options	17
1.3.1 IPv4	17
1.3.2 IPv6	17
1.3.3 Why IPv6?	18
1.3.4 Dual stack support	18
1.3.5 Migration from IPv4 to IPv6	18
1.3.6 IPv6 products for z/VSE	19
1.3.7 Securing your connections with Secure Sockets Layer	19
1.3.8 Options for printing	20
1.3.9 Overview of APIs	21
1.3.10 Available applications	22
1.3.11 Choosing a socket API when designing your applications	23
1.3.12 Enabling your applications for IPv6	24
1.4 Known problems	31
1.4.1 ERROR DURING OSA EXPRESS PROCESSING,REASON=002C CUU=nnnn,RETCODE=E00A	31

Chapter 2. TCP/IP for VSE/ESA	33
2.1 Overview	34
2.2 Standard features	34
2.3 Other optional features	34
2.4 Applications that are provided with TCP/IP for VSE	35
2.5 Application programming interfaces	35
2.6 Setting up and running TCP/IP for VSE	36
2.7 FTP hints	38
2.7.1 Internal FTP server suggestions	38
2.7.2 Using external FTPBATCH servers	38
2.8 Partition priorities	39
2.9 Security	40
2.10 Remote running with REXX	40
2.11 Version checking	43
2.12 Datagram analysis	43
2.13 Known problems	44
2.13.1 Routing in a subnet	44
2.13.2 Using SSL ciphers	44
2.13.3 Secure SSL port	44
2.13.4 SSL client does not verify the server certificate	45
2.13.5 TLS issue with IBM Personal Communications 6.0.7	45
Chapter 3. IPv6/VSE	47
3.1 Overview	48
3.2 Obtaining and activating a license key	48
3.3 Stack setup	49
3.3.1 IPv4 stack setup	50
3.3.2 IPv6 stack setup	51
3.3.3 Mixed IPv4 and IPv6 network setup	56
3.3.4 Setting up a dual-stacked system	57
3.3.5 UDPv4 in a coupled stack environment	58
3.3.6 Connectivity considerations	59
3.4 Setting up FTP	59
3.4.1 Security	59
3.4.2 VSE as a server	60
3.4.3 VSE as a client	61
3.5 Setting up TN3270	62
3.5.1 Setting up VTAM	62
3.5.2 Starting a TN3270 server	63
3.5.3 Controlling the terminal type	64
3.5.4 Connecting with IBM Personal Communications	65
3.5.5 Connecting with Open Text Exceed	66
3.5.6 Connecting with wc3270	66
3.5.7 Recovering from broken connections	67
3.6 Setting up TN3270E printing	68
3.6.1 TN3270E setup	68
3.6.2 BSTTVNET JCL	69
3.6.3 Node error program	69
3.7 Setting up SSL	71
3.7.1 Installing the prerequisite programs	71
3.7.2 Creating the keystore	71
3.7.3 Removing the private CA key from the client keyring file	78
3.7.4 Deciding whether to use the SSL proxy server or AT-TLS	80

3.7.5	Specifying parameters	81
3.7.6	Configuring the SSL proxy server	82
3.7.7	Configuring the AT-TLS server	83
3.7.8	Considerations on SSL performance	83
3.7.9	Considerations on blocking clear ports	84
3.7.10	Configuring wc3270 for SSL	85
3.7.11	Configuring IBM Personal Communications for SSL	88
3.7.12	Configuring secure FTP	91
3.7.13	Configuring VSE Connector Server	102
3.7.14	SSL client authentication	107
3.7.15	Using TLSv1.2	110
3.8	Setting up CICS Web Support	115
3.8.1	Modifying the CICS startup job	116
3.8.2	Defining the TCP/IP host name	116
3.8.3	Considerations for SSL	116
3.9	Known problems	119
3.9.1	VSE cannot be reached	119
3.9.2	BSTT075E LUNAME NOT AVAIL	120
3.9.3	SSL connect error with wc3270	120
3.9.4	Other SSL connect errors	120
3.9.5	Hang situation with BSTTATLS/BSTTPRXY	121
3.9.6	Return codes 3100 / 1700 from IJBCRLIB	121
3.9.7	BSTTFTPC fails to connect to Windows Server 2008	122
3.9.8	Batch email cannot relay mail	122
3.9.9	LDAP sign on by using SSL does not work	122
3.9.10	GnuTLS error -53: Error in the push function	123
Chapter 4.	Fast Path to Linux on System z	125
4.1	Overview	126
4.2	Concept of LFP instances and LFP daemons	127
4.3	LFP in a z/VM environment	127
4.3.1	Linux guest setup	128
4.3.2	VSE guest setup	130
4.4	z/VM IP Assist	133
4.4.1	Configuration user setup	133
4.4.2	Setting up the VIA guest	135
4.4.3	Setting up VSE guest	137
4.4.4	Using the LFP trace with VIA	138
4.5	LFP in an LPAR environment	140
4.5.1	Prerequisites	141
4.5.2	Hardware setup	141
4.5.3	VSE setup	142
4.5.4	Linux setup	143
4.6	IBM applications that support LFP	144
4.6.1	VSE Connector Server	144
4.6.2	Using the Virtual z/VSE FTP Daemon	144
4.7	Using secure connections with SSL	146
4.7.1	Using a VIA guest	146
4.7.2	Using a Linux on System z guest	147
4.7.3	Configuring the z/VSE virtual FTP daemon for SSL	147
4.8	Known problems	148
4.8.1	Error accessing the config disk	148
4.8.2	SE file transfer had a problem	148

4.8.3	User ID not authorized for SMSG	148
4.8.4	Invalid command response from VIA user	149
4.8.5	No response from VIA user.	149
4.8.6	Profile cannot be loaded	150
Chapter 5.	OpenSSL	151
5.1	Overview	152
5.1.1	What is available on z/VSE	152
5.1.2	What is unique in z/VSE	153
5.1.3	Runtime variables	153
5.1.4	What is not available in z/VSE	153
5.2	Access to the OpenSSL API	154
5.3	Creating random numbers	154
5.3.1	Characteristics of random number generators	155
5.3.2	Random number generation in OpenSSL.	155
5.3.3	Alternatives	156
5.3.4	Considerations for the z/VSE crypto device driver	156
5.3.5	Performance	157
5.4	Keystore considerations	158
5.4.1	Creating a PEM keystore	159
5.4.2	Exporting PEM to PFX	159
5.4.3	Importing PFX to PEM	159
5.4.4	Password-protected keystores	159
5.5	Programming.	160
5.5.1	Include files	161
5.5.2	Passed socket number	161
5.5.3	Socket calls.	162
5.5.4	Switching between GSK and OpenSSL socket calls	163
5.5.5	Specifying the key ring	163
5.5.6	Using a password-protected keyring.	164
5.5.7	Supported cipher suites	165
5.5.8	Specifying cipher suites	166
5.5.9	Supported RSA key lengths	166
5.5.10	Debugging.	167
5.5.11	Hardware crypto support.	167
5.5.12	Programming example	168
5.6	Performing the OpenSSL speed test	170
5.6.1	Test parameters	170
5.6.2	Test results	171
5.7	How OpenSSL is used on z/VSE	173
5.8	OpenSSL vulnerabilities	175
5.9	Considerations on TLSv1.2.	175
5.10	Considerations about Diffie-Hellman.	176
5.10.1	RSA.	176
5.10.2	Diffie-Hellman	178
5.10.3	Variants of Diffie-Hellman	178
5.11	Using DHE-RSA with OpenSSL on z/VSE	179
5.11.1	Generating DH parameters.	179
5.11.2	Using DHE-RSA with Java-based connector	183
5.12	Considerations on Elliptic Curve Cryptography.	184
5.13	Using ECDHE-RSA with OpenSSL on z/VSE.	185
5.13.1	Generating the EC key	185
5.13.2	Uploading the EC key to VSE	185

5.13.3 Using ECDHE-RSA with Java-based connector	186
5.14 Restrictions	187
5.14.1 No SHA-512 support	187
Chapter 6. Comparison of stacks and protocols	189
6.1 Stacks comparison	190
6.1.1 Licensing	190
6.1.2 Installation libraries	190
6.1.3 Virtual storage organization	190
6.1.4 Commands	191
6.1.5 Comparison of protocols	192
6.2 Applications comparison	193
6.2.1 FTP server	193
6.2.2 FTP clients	199
6.2.3 Uploading a virtual tape into VSAM	200
6.2.4 AutoFTP	202
6.2.5 TN3270	203
6.2.6 Printing	204
6.2.7 AutoLPR	205
6.2.8 Inserts coding	205
6.2.9 Email	206
6.2.10 Creating PDF documents	208
6.2.11 Remote EXEC client	208
6.3 Performance	209
6.4 SSL	210
6.5 Comparison of APIs	211
6.5.1 Socket APIs	211
6.5.2 SSL APIs	212
6.5.3 Crypto APIs	212
6.6 Considerations for DB2 Server for VSE interfaces	213
6.7 Considerations for IBM applications	213
6.7.1 VSE Connector Server	213
6.7.2 Virtual tape	213
6.7.3 CICS Web Support	214
6.7.4 Encryption Facility for z/VSE	215
6.7.5 Basic Security Manager	215
6.7.6 Uploading PTF files to IJSYSPF	216
6.8 Known problem: ftp.exe hangs on Windows 7	218
6.8.1 Symptom	218
6.8.2 Solution	218
Appendix A. API reference	219
Socket APIs	220
SSL APIs	221
z/OS SSL API	221
OpenSSL API	222
Related publications	227
IBM Redbooks	227
Other publications	227
Online resources	228
Help from IBM	229

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®	Parallel Sysplex®	VIA®
DB2®	POWER®	VTAM®
DRDA®	Rational®	WebSphere®
GDPS®	Redbooks®	z/OS®
Geographically Dispersed Parallel Sysplex™	Redbooks (logo)  ®	z/VM®
HiperSockets™	System z10®	z/VSE®
IBM®	System z9®	z10™
Language Environment®	System z®	z9®
	UC™	zEnterprise®

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

The importance of modern computer networks is steadily growing as increasing amounts of data are exchanged over company intranets and the Internet. Understanding current networking technologies and communication protocols that are available for the IBM® mainframe and System z® operating systems is essential for setting up your network infrastructure with IBM z/VSE®.

This IBM Redbooks® publication helps you install, tailor, and configure new networking options for z/VSE that are available with TCP/IP for VSE/ESA, IPv6/VSE, and Fast Path to Linux on System z (Linux Fast Path). We put a strong focus on network security and describe how the new OpenSSL-based SSL runtime component can be used to enhance the security of your business.

This IBM Redbooks publication extends the information that is provided in *Security on IBM z/VSE*, SG24-7691.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the IBM Development Laboratory in Boblingen, Germany and at the International Technical Support Organization, Poughkeepsie Center.

Joerg Schmidbauer is a z/VSE developer in the IBM Development Laboratory, Boblingen, Germany. He has worked for IBM since 1989 in VSE Development. In addition to z/VSE connector-related work, he works on z/VSE hardware cryptographic support. His latest project was porting OpenSSL to z/VSE.

Jeffrey Barnard is a Communications and Security Specialist and co-founder of Barnard Software, Inc. He has over 30 years of experience with z/VSE, communications software, and security. He is also one of the authors of IPv6/VSE and is well-known in the z/VSE community.

Ingo Franzki is part of the z/VSE Development Team at the IBM Development Laboratory, Boblingen, Germany. He joined IBM in 1997. His main focus is on z/VSE connectors, security, and performance. He is also consultant for solutions that involve distributed environments with z/VSE. Ingo is often a speaker at IBM events, user group meetings, and workshops.

Karsten Graul has worked in z/VSE development since 2000. His main work areas are the z/VSE connectors and Linux Fast Path. In recent years, he worked on the IBM z/VM® TCP/IP development team. Currently, Karsten is a member of the z/VSE TCP/IP team.

Don Stoeve is a Senior Product Developer for CSI International in Columbus, OH and he brings 30 years of VSE systems programming, product development, and technical support experience. He has worked on the TCP/IP for VSE product since 1998 and now oversees its development and support.

Rene Trumpp is a z/VSE software engineer in the IBM Research and Development Laboratory, Boblingen, Germany. After joining IBM in 2006, he became an expert in System z networks. Besides implementing z/VSE SNMP support (z/VSE Monitoring Agent and z/VSE SNMP Trap Client) and the IBM z/VSE GDPS® Client, he also worked in the z/VM TCP/IP development team. Today, he is the key focal point for the z/VSE network device driver and part of the z/VSE TCP/IP team.

Thanks to the following people for their contributions to this project:

- ▶ Pavel Kurilov, IBM Lab Moscow, Russia
- ▶ Geir Fladby, AutoData, Norway
- ▶ Heiko Schnell, IBM Lab Boblingen, Germany
- ▶ Jens Remus, IBM Lab Boblingen, Germany
- ▶ Rich Smrcina, Sytek Services
- ▶ Gregg Miller, South Eastern Technology Group
- ▶ Tom Grossheider, z/VSE Service and Support, US

Special thanks to the following people in the ITSO center in Poughkeepsie, New York:

- ▶ Mike Ebberts, Project Leader
- ▶ Michael B. Schwartz, Redbooks Webmaster

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>

Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get up-to-the-minute Redbooks news and announcements
- ▶ Link to the latest Redbooks blogs and videos

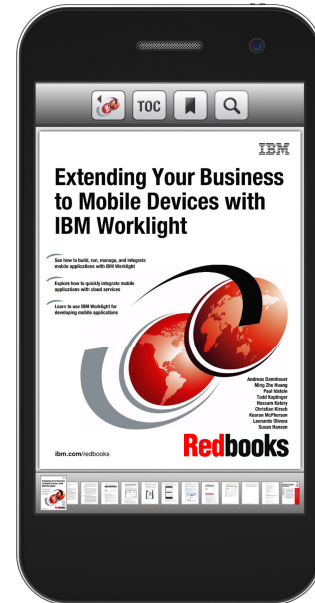
Get the latest version of the Redbooks Mobile App



iOS

Download
Now

Android



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-8091-01
for Enhanced Networking on IBM z/VSE
as created or updated on December 29, 2014.

December 2014, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

- ▶ New section 1.2.8, “Using VTAM (SNA) and TCP/IP (non-QDIO) parallel on the same CHPID” on page 8.
- ▶ New section 1.2.15, “OSAX Hotswap support” on page 15.
- ▶ New section 1.4, “Known problems” on page 31.
- ▶ New section 3.7.8, “Considerations on SSL performance” on page 83.
- ▶ New section 5.10, “Considerations about Diffie-Hellman” on page 176.
- ▶ New section 5.11, “Using DHE-RSA with OpenSSL on z/VSE” on page 179.
- ▶ New section 5.12, “Considerations on Elliptic Curve Cryptography” on page 184.
- ▶ New section 5.13, “Using ECDHE-RSA with OpenSSL on z/VSE” on page 185.
- ▶ New section 6.1.1, “Licensing” on page 190.
- ▶ New section 6.2.3, “Uploading a virtual tape into VSAM” on page 200.
- ▶ New section 6.7.6, “Uploading PTF files to IJSYSPF” on page 216.

Changed information

- ▶ Updated section “Configuring Firefox” on page 117.
- ▶ Updated section 5.5.7, “Supported cipher suites” on page 165.
- ▶ Updated section 5.7, “How OpenSSL is used on z/VSE” on page 173.
- ▶ Updated section 6.2.4, “AutoFTP” on page 202.
- ▶ Various updates in Appendix A, “API reference” on page 219.



Networking options overview

This chapter provides an overview of networking options that are available for IBM z/VSE.

Hardware options include the features of the latest IBM System z and z Enterprise processors and various types of networking adapters, such as Open Systems Adapter-Express, IBM HiperSockets™, and Intra-Ensemble Data Network (IEDN).

Software options include the different IP stacks that are available for z/VSE, TCP/IP for VSE/ESA, IPv6/VSE, and Fast Path to Linux on System z (Linux Fast Path), and different networking protocols, such as IPv4 and IPv6.

There are applications that are designed for a specific stack. Other applications can be used together with all stacks.

Finally, we describe how you can secure your connections by using Secure Sockets Layer (SSL).

This chapter includes the following topics:

- ▶ Overview
- ▶ Hardware options
- ▶ Software options
- ▶ Known problems

1.1 Overview

The following sections provide an overview of the networking options that are available for z/VSE.

In this book, we describe only the hardware options that are based on Open Systems Adapter (OSA) technology. Earlier options, such as Common Link Access to Workstation (CLAW) or Channel to Channel Adapter (CTCA), are no longer relevant for most customers. For more information about CLAW, see *Networking with z/OS and Cisco Routers: An Interoperability Guide*, SG24-6297.

Software options for z/VSE include IPv4 and IPv6 in different environments and IP stacks.

1.2 Hardware options

This section describes various connectivity options that are based on OSA technology as they are relevant for z/VSE. For a general overview on OSA, see the following resources:

- ▶ *OSA Express Implementation Guide*, SG24-5948, which is available at this website:
<http://www.redbooks.ibm.com/redbooks/pdfs/sg245948.pdf>
- ▶ *Open Systems Adapter-Express Customer's Guide and Reference*, SA22-7935, which is available at this website:
<http://www.ibm.com/support/docview.wss?uid=isg2bc4ae2e43bfcf12c85256cee000d1130&aid=1>
- ▶ *IBM System z Networking Features*:
<http://www.ibm.com/systems/z/hardware/networking/features.html>
- ▶ *Introducing OSA-Express, OSA-Express2, and OSA-Express3*:
<http://publib.boulder.ibm.com/infocenter/zos/v1r12/index.jsp?topic=/com.ibm.zos.r12.ioaz100%2Fstart.htm>
- ▶ *IBM System z Connectivity Handbook*, SG24-5444, which is available at this website:
<http://www.redbooks.ibm.com/redpieces/abstracts/sg245444.html?Open>

1.2.1 OSA-Express

The OSA is a network controller that you can install in a mainframe I/O cage. The adapter integrates several hardware features and supports many networking transport protocols. The OSA card is the strategic communications device for the mainframe architecture. It has several key features that distinguish it from CCW-based communications.

The OSA integrates the control unit and device into the same hardware by placing it on a single card that directly connects to the central processor complex I/O bus.¹

¹ From *Networking on z/OS*, which is available at this website:
<http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp?topic=/com.ibm.zos.zbasics/lcmain.html>

The following versions of the OSA are available:

- ▶ OSA-Express
- ▶ OSA-Express2
- ▶ OSA-Express3
- ▶ OSA-Express4S
- ▶ OSA-Express5S

z/VSE supports OSA-Express adapters in IBM System z environments. OSA-Express in QDIO mode is supported since VSE/ESA 2.6 and IBM HiperSockets since VSE/ESA 2.7.

The OSA-Express family of network adapters (OSA-Express5S, OSA-Express4S, OSA-Express3, and OSA-Express2) supports the following features:

- ▶ 10-Gigabit Ethernet
- ▶ Gigabit Ethernet
- ▶ 1000BASE-T Ethernet

Note: Starting with OSA-Express4S, group addressing is no longer supported.

Table 1-1 shows the channel-path identifier (CHPID) types that are supported by z/VSE.

Table 1-1 OSA adapter CHPID types

CHPID type	Description
SA-Integrated Console Controller (OSC)	OSA-ICC for emulation of TN3270E and non-SNA DFT 3270
OSA-Express in QDIO mode (OSD)	Queued Direct Input/Output (QDIO) architecture
OSA-Express (OSE)	non-QDIO Mode (OSA-2, for SNA/APPN connections)
OSA for NCP support (OSN)	OSA-Express for NCP: Appears to z/VSE as a device that is supporting Channel Data Link Control (CDLC) protocol.
OSA-Express for zBX (OSX)	OSA-Express for zBX. Provides connectivity and access control to the Intra-Ensemble Data Network (IEDN) from z196, z114, zEC12, and zBC12 to Unified Resource Manager functions.

These CHPID types are described next. For each type, there is a sample IOCDS definition, the related statement in the IPL procedure, and how TCP/IP is configured to use this CHPID type.

1.2.2 OSA-Integrated Console Controller

The OSC CHPID type is available on mainframes that are running an OSA-Express2 card or an OSA-Express card with the Gigabit Ethernet feature. The OSC is a special channel type that eliminates the need for an external console controller. The OSC CHPID can also be used to connect TN3270 sessions (with some limitations).

IOCDs definition

The IOCDs definition is shown in the following example:

```
CHPID PATH=(CSS(0,2,3),0C),SHARED,*
      PARTITION=((CSS(0),(0),(POE)),(CSS(2),(GRY2LP41,GRY2LP42*,
      ,GRY2LP43,GRY2LP44,GRY2LP45),(=))),*
NOTPART=((CSS(3),(GRY2LP60))),PCHID=5D1,TYPE=OSC
CNTLUNIT CUNUMBR=0003,PATH=((CSS(2),0C),(CSS(3),0C)),UNIT=OSC
IODEVICE ADDRESS=(cuu1,num),MODEL=X,UNITADD=00,CUNUMBR=(0003),*
      UNIT=3270
```

IPL procedure

The IPL procedure is shown in the following example:

```
ADD cuu1:cuu2,3277
```

CHPID type OSC does not support TCP/IP traffic.

1.2.3 OSA-Express in QDIO mode

The Queued Direct Input/Output (QDIO) architecture provides increased performance because of minimized I/O interrupts and I/O path-lengths. It dramatically improves data transfer speed and efficiency for TCP/IP traffic.

To access an OSA-Express adapter in QDIO mode, you need the following OSA-Express devices:

- ▶ Read device
- ▶ Write device
- ▶ Data path device

Note: Use QDIO mode wherever possible.

IOCDs definition for OSD

The IOCDs definition for OSD is shown in the following example:

```
CHPID PATH=(CSS(0,1,2),B6),SHARED,**
      PARTITION=((CSS(0),(COH1,VMA,VMCOV),(VMC)),(CSS(2),(LNXT**
      ST1,VMUNI),(=))),PCHID=540,TYPE=OSD
CNTLUNIT CUNUMBR=0003,*
      PATH=((CSS(0),B6),(CSS(1),B6),(CSS(2),B6)),UNIT=OSA
IODEVICE ADDRESS=(cuu1,num),CUNUMBR=(0003),UNIT=OSA
```

IPL procedure

Add the device addresses in the IPL procedure as device type OSAX, as shown in the following example:

```
ADD cuu1-cuu3,OSAX
```

TCP/IP for VSE/ESA

In TCP/IP for VSE/ESA, you define a LINK with type OSAX, as shown in the following example:

```
DEFINE LINK,ID=...,TYPE=OSAX,DEV=cuu1,DATAPATH=cuu3,IPADDR=addr,...
```


IPv6/VSE

In IPv6/VSE, you define a DEVICE with type OSAX.

```
DEVICE device_name OSAX cuu1 portname cuu3
```

The device numbers in the DEV parameter of the DEFINE LINK statement (TCP/IP for VSE/ESA) or DEVICE statement (IPv6/VSE) must be an even/odd pair for the read and write device.

Note: If you are running under z/VM, ensure that the real device numbers as generated in the IOCP are an even/odd pair.

Additionally, the device number of the write device must be the successor (+1) of the device number of the read device.

1.2.4 OSA-Express

The CHPID type OSE emulates former OSA-2 devices.

IOCDs definition for OSE

The IOCDs definition for OSE is shown in the following example:

```
CHPID PATH=(CSS(0),94), *  
      PARTITION=( (R35LP01), (R35LP02,R35LP03,R35LP04,R35LP05,R3*  
      5LP06,R35LP07,R35LP08,R35LP09,R35LP10,R35LP13,R35LP14,R3*  
      5LP15),REC),PCHID=550,TYPE=OSE  
CNTLUNIT CUNUMBR=0003,PATH=((CSS(0),94),(CSS(1),94)),UNIT=OSA  
IODEVICE ADDRESS=(cuu1,num),CUNUMBR=(0003),UNIT=OSA
```

IPL procedure

Add the device addresses in the IPL procedure as device type OSA, as shown in the following example:

```
ADD cuu1-cuu2,OSA
```

TCP/IP for VSE/ESA

The TCP/IP for VSE/ESA is shown in the following example:

```
DEFINE LINK,ID=...,TYPE=OSA,DEV=(cuu1,cuu2), .....
```

IPv6/VSE

When IPv4 is used, you use the following DEVICE command:

```
DEVICE device_name OSA cuu1 ETHERNET
```

IPv6 is not supported by CHPID type OSE.

1.2.5 OSA for NCP support

Open Systems Adapter for NCP (OSA for NCP) support provides channel connectivity from the operating systems to the Communication Controller for Linux on System z (CCL) image. Each operating system that supports CDLC can use this connectivity option without changes to the operating system. OSN was available since IBM System z9® and IBM System z10® servers.

CHPID type OSN requires 1000BASE-T Ethernet with two ports per feature.

IOCDs definition

The IOCDs definition is shown in the following example:

```
CHPID PATH=(CSS(0,1,2,3),0D),SHARED,*
      NOTPART=((CSS(1),(IRD6),(=)),(CSS(2),(TUXMAKER),(=)),(CS*
      S(3),(IRD1,IRD2))),PCHID=370,TYPE=OSN
CNTLUNIT CUNUMBR=0003,*
      PATH=((CSS(0),0D),(CSS(1),0D),(CSS(2),0D),(CSS(3),0D)),*
      UNIT=OSN
IODEVICE ADDRESS=(070,008),UNITADD=00,CUNUMBR=(0003),UNIT=OSN
IODEVICE ADDRESS=(078,003),UNITADD=08,CUNUMBR=(0003),UNIT=3745

IODEVICE ADDRESS=07F,UNITADD=FE,CUNUMBR=(0003),UNIT=0SAD
```

IPL procedure

Emulates a 3745 adapter and is used like a real 3745 adapter in z/VSE.

There is no TCP/IP support for CHPID type OSN.

1.2.6 Intra-Ensemble Data Network support

The OSA-Express for zBX (CHPID type OSX) provides connectivity and access control to the IEDN from IBM zEnterprise® 196, z114, zEC12, and zBC12 to Unified Resource Manager functions. IEDN is supported by OSA-Express 10 GbE features only.

An IEDN is a private 10-Gigabit Ethernet network for application data communications within an ensemble. Data communications for workloads can flow over the IEDN within and between nodes of an ensemble. All of the physical and logical resources of the IEDN are configured, provisioned, and managed by the Unified Resource Manager.

An OSA-Express adapter that is connected to an IEDN has fast and secure network access to other nodes and systems that also connected to the IEDN.

An IEDN provides connectivity between the following components:

- ▶ A zEnterprise Central Electrical Complex (CEC) and IBM zEnterprise BladeCenter Extensions (zBXs)
- ▶ Two or more zEnterprise CECs

z/VSE supports the IEDN network of a zEnterprise 196, z114, or zEC12 in the following environments:

- ▶ z/VSE V4.2, V4.3 and V5.1: IBM z/VM VSWITCH and OSDSIM mode in a z/VM 6.1 guest environment.
- ▶ z/VSE V5.1: OSA-Express for zBX devices in an LPAR or z/VM guest environment with dedicated OSAX devices. This requires VLAN support.

IOCDs definition

The IOCDs definition is shown in the following example:

```
CHPID PATH=(CSS(0,2),CA),SHARED, *
      PARTITION=((CSS(0),(DWB1,DWB2),(=)),(CSS(2),(LNXTST1),(V*
      MUNI))),PCHID=3F8,TYPE=OSX
CNTLUNIT CUNUMBR=0003,PATH=((CSS(0),CA),(CSS(2),CA)),UNIT=OSX
IODEVICE ADDRESS=(cuu1,num),MODEL=X,CUNUMBR=(0003),UNIT=OSA
```

IPL procedure

The IPL procedure is shown in the following example:

```
ADD cuu1-cuu3,OSAX
```

TCP/IP for VSE/ESA

From a TCP/IP for VSE/ESA perspective, OSX is identical to an OSAX link, as shown in the following example:

```
DEFINE LINK,ID=...,TYPE=OSAX,DEV=cuu1,DATAPATH=cuu3
```

IPv6/VSE

The IPv6/VSE product is shown in the following example:

```
DEVICE device_name OSAX cuu1 portname cuu3
```

The next sections provide more information about the following OSA-Express features:

- ▶ OSA-Express Multi-Port Support
- ▶ HiperSockets
- ▶ Virtual LAN

1.2.7 OSA-Express multi-port support

OSA-Express3 or later provides two ports per CHPID for selected features. The default port is port 0.

For CHPID type OSE (non-QDIO mode), you must use OSA/SF to select the OSA port. Because there are multiple ports per CHPID, both of the ports that are associated with the CHPID are the same; that is, OSD, OSE, or OSC. It is not possible, for example, to have one port that is defined as OSC and the other as OSD. However, it is possible to have different CHPID types for the same card where the card does support multiple CHPIDs.

Because the number of CHPIDs is limited, the Multi-Port support increases the number of ports you can use. When OSA-Express3 adapters are used, the port number that is used is specified as the adapter number.

IOCDs definition

OSA-Express Multi-Port support is available for CHPID types OSC, OSD, OSE, and OSN. For more information, see 1.2.2, “OSA-Integrated Console Controller” on page 3, 1.2.3, “OSA-Express in QDIO mode” on page 4, 1.2.4, “OSA-Express” on page 5, and 1.2.5, “OSA for NCP support” on page 5.

IPL procedure

For more information, see the IPL procedure headings in the following sections:

- ▶ 1.2.2, “OSA-Integrated Console Controller” on page 3
- ▶ 1.2.3, “OSA-Express in QDIO mode” on page 4
- ▶ 1.2.4, “OSA-Express” on page 5
- ▶ 1.2.5, “OSA for NCP support” on page 5
- ▶ 1.2.6, “Intra-Ensemble Data Network support” on page 6

TCP/IP for VSE/ESA

The default port is port 0. To use port 1, you must specify the port number by using the **OSAPORT** parameter of the **DEFINE LINK** statement, as shown in the following example for CHPID type OSD:

```
DEFINE LINK, ID=linkname, TYPE=OSAX, DEV=cuu1, DATAPATH=cuu3, OSAPORT=1, ...
DEFINE ROUTE ID=local, LINKID=linkname, IPADDR=subnetaddr
DEFINE ROUTE ID=default, LINKID=linkname, IPADDR=0.0.0.0, GATEWAY=ipaddr
```

With OSAX QDIO, you do not specify the OSA port in the **DEFINE ROUTE** statement. The correct adapter is identified by the OSA card. The IP address of the gateway must be a router that is attached to the local subnet.

IPv6/VSE

For IPv6/VSE, you specify the adapter number immediately after the device name in the **LINK** command, as shown in the following example for CHPID type OSD:

```
DEVICE device_name OSAX cuu1 portname cuu3
LINK device_name 1 IPv6_addr netmask mtu
```

1.2.8 Using VTAM (SNA) and TCP/IP (non-QDIO) parallel on the same CHPID

IOCDs definitions needed for IBM VTAM® (SNA) and TCP/IP using OSA (non-QDIO):

```
CHPID TYPE=OSE
CNTLUNIT UNIT=OSA
IODEVICE UNIT=OSA
```

Definitions in z/VSE IPL procedure for VTAM (SNA):

```
ADD cuu,OSA
```

VTAM definitions for using SNA:

```
VBUILD TYPE=XCA
PORT MEDIUM=CSMACD
ADAPNO=1
CUADDR=cuu
```

Definitions in z/VSE IPL procedure for TCP/IP using OSA (non-QDIO):

```
ADD cuu1:cuu2,OSA
```

For non-QDIO, you need a cuu even-odd pair, for example **ADD 260:261,OSA** or **ADD 270:271,OSA**.

TCP/IP definitions for using OSA (non-QDIO):

```
SET IPADDR=vseipaddr
SET MASK=subnetmask
```

```

DEFINE LINK ID=linkname,TYPE=LCS|OSA|OSA2|3172,DEVICES=cuu1
DEFINE ADAPTER LINKID=linkname,TYPE=ETHERNET,NUMBER=1
DEFINE ROUTE ID=local,LINKID=linkname,ADAPTER=1,IPADDR=subnetaddr
DEFINE ROUTE ID=default,LINKID=linkname,ADAPTER=1,IPADDR=0.0.0.0,GATEWAY=ipaddr

```

The IP address of the gateway must be a router attached to the local subnet.

1.2.9 HiperSockets (IQD)

The HiperSockets implementation is based on the OSA-Express Queued Direct I/O (QDIO) protocol. HiperSockets is also called internal QDIO (iQDIO).

HiperSockets provide “network within the box” functionality, with which high speed any-to-any connectivity among operating systems is possible without requiring any physical cabling. Because a HiperSockets connection does not leave the box, it is secure. This means that there is no attack possible from outside. The entire communication is done in memory, which results in increased performance.

IOCDs definition

HiperSockets must be defined in the IOCDs as CHPID type IQD, as shown in the following example:

```

CHPID PATH=(CSS(0,1),E5),SHARED,*
      PARTITION=((CSS(0),(0),(DOE)),(CSS(1),(COH2,COH3,VMCOH),*
      (=))),CHPARM=40,TYPE=IQD
CNTLUNIT CUNUMBR=0003,PATH=((CSS(0),E5),(CSS(1),E5)),UNIT=IQD
IODEVICE ADDRESS=(cuu1,num),CUNUMBR=(0003),UNIT=IQD

```

The frame size is defined by using CHPARM parameter (formerly OS=nn), as shown in the following example:

```

CHPARM=00 16K (MTU=8K) (default)
CHPARM=40 24K (MTU=16K)
CHPARM=80 40K (MTU=32K)
CHPARM=C0 64K (MTU=56K)

```

IPL procedure

You must add HiperSockets devices in the IPL procedure as device type OSAX with mode 01, as shown in the following example:

```
ADD cuu1-cuu3,OSAX,01
```

TCP/IP for VSE/ESA

The DEFINE LINK statement for HiperSockets is similar to those that we use for defining CHPIDs, as shown in the following example:

```
DEFINE LINK,ID=...,TYPE=OSAX,DEV=cuu1,DATAPATH=cuu3
```

IPv6/VSE

Defining HiperSockets in the IPv6/VSE product is similar to defining OSA CHPIDs, as shown in the following example:

```
DEVICE OSA HIPR cuu1 TRLHF9 cuu3
```

1.2.10 Virtual local area network

Virtual local area network (VLAN) requires OSA-Express2, OSA-Express3, or later features that are configured in QDIO mode. By using VLAN, a physical network can be divided administratively into separate logical networks. These logical networks operate as though they are physically independent of each other.

z/VSE provides VLAN support for OSA-Express CHPID types OSD and OSX, and HiperSockets devices since z/VSE 5.1.

Note: In a Layer 3 configuration, VLANs can be used transparently by IPv6/VSE and TCP/IP for VSE/ESA.

If you want to configure VLANs for OSA-Express (CHPID type OSD and OSX) devices in a Layer 2 configuration that carries IPv6 traffic, the IPv6/VSE product is required.

You can use one of the following two methods to configure your system to use VLAN:

- ▶ Configure one or more VLANs in the TCP/IP stack of IPv6/VSE by using the LINK command. For more information about IPv6/VSE commands, see *IPv6/VSE Installation Guide*, SC34-2616.
- ▶ Generate and catalog phase IJBOCONF that includes the global VLANs that are to be used with your OSAX devices. z/VSE provides skeleton SKOSACFG to generate phase IJBOCONF. The VLANs that are contained in IJBOCONF can be used transparently for Layer 3 links by IPv6/VSE and TCP/IP for VSE/ESA. For more information, see *z/VSE Planning*, SC34-2635.

Example 1-1 shows a sample configuration setting VLAN ID 200 for Device D00.

Example 1-1 Sample SKOSACFG configuration

```
* $$ JOB JNM=IJBOCONF,CLASS=A,DISP=D
// JOB IJBOCONF GENERATE IJBOSA MODULE CONFIGURATION PHASE
// LIBDEF *,CATALOG=PRD2.CONFIG
// LIBDEF *,SEARCH=PRD1.BASE
// OPTION ERRS,SXREF,SYM,NODECK,CATAL,LISTX
// PHASE IJBOCONF,*
// EXEC ASMA90,SIZE=(ASMA90,64K)
IJBOCONF CSECT
IJBOCONF AMODE ANY
IJBOCONF RMODE ANY
*
* * * * *
*
* GLOBAL VLAN DEFINITION
*
* DEFGVLAN DEVNO=<CUU>, VLAN_ID=<ID>, VLAN_Prio=<PRIO>
* <CUU> VSE DEVICE NUMBER IN HEX FORMAT
* <ID> VLAN ID IN DECIMAL FORMAT (1 ... 4095)
* <PRIO> VLAN PRIORITY (VALID VALUES 0 ... 7)
*
* * * * *
*
* DEFGVLAN DEVNO=0D00,VLAN_ID=200
*
* * * * *
```

The job template SKOSACFG is in ICCF library 59.

1.2.11 Shared OSA adapter versus HiperSockets

- Using a shared OSA adapter

► Using HiperSockets

1.2.12 Using HiperSockets to communicate with Linux on System z

The default value of 16 was increased to 128 with RHEL6.2 and SLES11 SP2.

IPv6/VSE does not have this timer delay performance problem.

The problem can become dramatic if more than 16 packets are queued up to be sent after a BUSY situation. The resend immediately floods the Linux buffers again, which leads to the next BUSY situation.

You can check for the issue by using **QUERY STATS, LINKID=xxxx [,RESET]** if you ever encounter the BUSY situation (RESET resets the counters), as shown in the following example:

```
C1 0065 0004: IPL615I  Busy mode.....0
C1 0065 0004: IPL615I  Busy mode, longest.....0
```

In TCP/IP for VSE/ESA, you can configure a shorter BUSY wait time with a **DEFINE LINK** command, where **BUSY=nnn** (shortest possible wait time is 100 msec.).

Tip: For more information about optimizing HiperSockets with IPv6/VSE, see the Bernard Software, Inc. blog, which is available at this website:

<http://bsitcpip.blogspot.com/2010/10/optimizing-hipersockets-in-zvse.html>

APAR DY47394 with PTF UD53905 provides a configuration option with which you can configure the number of QDIO input buffers (the default is 8). Use macro QDIOBUFF (newly shipped with this APAR) in skeleton SKOSACFG in ICCF library 59. It improves the performance in a Linux on System z and z/VSE HiperSockets environment when the number of queues is unbalanced.

The next section describes how to configure the QDIO buffers in z/VSE.

1.2.13 QDIO buffer configuration

Since z/VSE v5.1 (with PTF UD53905), the number of QDIO input queue buffers can be configured. This ability might be needed by clients who extend z/VSE solutions by using Linux on System z that use a HiperSockets network for high-speed TCP/IP connectivity between z/VSE and Linux on System z. Best performance can be achieved when data is always delivered successfully without the need for resending.

Because HiperSockets transfers data synchronously, successful delivery of data depends on free Queued Direct I/O (QDIO) input buffers of the target system. z/VSE uses a default of eight QDIO input buffers. This might not always be sufficient, especially if clients increased the number of QDIO input buffers on the Linux on System z system.

To configure the number of QDIO input buffers for HiperSockets (CHPID type IQD) and OSA-Express devices (CHPID types OSD and OSX), the configuration skeleton SKOSACFG in ICCF library 59 can be used. More QDIO input buffers might require increasing the size of the TCP/IP partition.

Note: z/VSE needs 1 MB of 31-bit partition GETVIS space per link (for eight input queue buffers). For each additional input queue buffer, z/VSE needs 64K (OSA-Express) and up to 64K (HiperSockets, depending on your IOCDs definition) more 31-bit partition GETVIS space. Therefore, when the client increases the count of input queue buffers, they also might increase the partition GETVIS space. Because the input queue buffers are PFIXed, the client also must increase the value of the JCL SETPFIX LIMIT statement in their TCP/IP startup job accordingly.

Example 1-2 shows a sample configuration setting 16 QDIO input buffers for device D00. Skeleton SKOSACFG is provided in ICCF library 59.

Example 1-2 Sample SKOSACFG configuration

```

* $$ JOB JNM=IJBOCONF,CLASS=A,DISP=D
// JOB IJBOCONF GENERATE IJBOSA MODULE CONFIGURATION PHASE
// LIBDEF *,CATALOG=PRD2.CONFIG
// LIBDEF *,SEARCH=PRD1.BASE
// OPTION ERRS,SXREF,SYM,NODECK,CATAL,LISTX
  PHASE IJBOCONF,*
// EXEC ASMA90,SIZE=(ASMA90,64K)
IJBOCONF CSECT
IJBOCONF AMODE ANY
IJBOCONF RMODE ANY
*
* * * * *
*
*  GENERAL CONFIGURATION
*
* * * * *
*
*  QDIO BUFFER COUNT
*  YOU CAN SPECIFY THE NUMBER OF QDIO INPUT QUEUE BUFFERS BY
*  ADJUSTING THE QDIO BUFFER COUNT. THE NUMBER OF QUEUE BUFFERS
*  IS ALLOWED TO BE 8, 16, 32 or 64.
*  THE DEFAULT IS 8.
*
*  QDIOBUFF DEVNO=<CUU>,IQBUF=<COUNT>
*      <CUU>      VSE DEVICE NUMBER IN HEX FORMAT
*      <COUNT>   QDIO BUFFER COUNT
*                  (VALID VALUE: 8, 16, 32 or 64)
*
*      QDIOBUFF DEVNO=0D00,IQBUF=16
*
*  NOTE:
*  UP TO Z/VSE VERSION 5.1 WE NEED 1MB OF 31-BIT PARTITION GETVIS
*  SPACE PER LINK (FOR 8 INPUT QUEUE BUFFERS). FOR EACH ADDITIONAL
*  INPUT QUEUE BUFFER, WE NEED 64K (OSA-EXPRESS) AND UP TO 64K
*  (HIPERSOCKETS - DEPENDING ON YOUR IOCDs DEFINITION) ADDITIONAL
*  31-BIT PARTITION GETVIS SPACE. SO IF YOU INCREASES THE COUNT OF
*  INPUT QUEUE BUFFERS, YOU MAY ALSO HAVE TO INCREASE THE PARTITION
*  GETVIS SPACE. SINCE THE INPUT QUEUE BUFFERS ARE PFIXED, YOU ALSO
*  HAVE TO INCREASE THE ABOVE VALUE OF THE JCL SETPFIX LIMIT
*  STATEMENT IN YOUR TCP/IP STARTUP JOB ACCORDINGLY.
*
* * * * *
*
*      END
*/
// IF $MRC GT 4 THEN
// GOTO NOLINK
// EXEC LNKEDT,PARM='MSHP'
/. NOLINK
/&
* $$ EOJ

```

1.2.14 Virtual OSA devices and VMAC

Before the introduction of the virtual MAC (VMAC) function, an OSA interface had only one MAC address. This restriction caused problems when load balancing technologies were used with TCP/IP stacks that share OSA interfaces. The single MAC address of the OSA also causes a problem when TCP/IP stacks are used as a forwarding router for packets that are destined to unregistered IP addresses.

VMAC support enables an OSA interface to have a physical MAC address and many distinct virtual MAC addresses for each device or interface in a stack. That is, each stack can define up to eight VMACs per protocol (IPv4 or IPv6) for each OSA interface. VMAC is supported on z9 and later processors with all OSA types.

For more information, see *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798.

VMAC support helps to simplify the infrastructure and provide load balancing when a logical partition is sharing the OSA MAC address with another logical partition. Each operating system instance can now have its own unique virtual MAC (VMAC) address. All IP addresses that are associated with a TCP/IP stack are accessible by using their own VMAC address, instead of sharing the MAC address of an OSA port. This applies to Layer 3 mode and to an OSA port shared among logical partitions.

Note: Layer 3 means that the stack sees only the IP packets. The hardware Ethernet frames, MAC addresses, and ARP processing are offloaded from the host TCP/IP stack. This reduces stack processor usage, but limits some types of processing.

Layer 2 means that the stack is responsible for managing the TCP/IP packets at the Ethernet frame level. The stack must perform its own ARP/Neighbor Discovery (IPv6) and track or manage Ethernet MAC addresses.

In z/VSE, the idea of defining a VMAC is a Layer 2 concept. When an OSA-Express adapter is running in Layer 3 mode, all MAC processing (ARP, and so on) is managed by or offloaded to the adapter. However, when the adapter is running in Layer 2 mode, the stack is responsible for managing and tracking the MAC and IP addresses. This allows the stack to inform the adapter that multiple MAC addresses are being used, which allows the stack to respond to multiple IP addresses.

IOCDs definition

There is no change to the OSA-Express definition in the IOCDs.

TCP/IP for VSE/ESA

TCP/IP for VSE/ESA does currently not support VMAC.

IPv6/VSE

IPv6/VSE supports virtual OSA-Express devices in Layer 2 mode. This means you can define virtual OSA-Express Layer 2 devices, each with their own IPv4/IPv6 address and VMAC address.

Note: z/VSE VMAC support requires z/VSE 5.1 plus IPv6/VSE at GA Build 254.

When you are running under z/VSE 5.1, the OSA-Express adapter can be started by the stack to run in Layer 2 mode. This is done by using the LAYER 2 literal on the DEVICE statement. Also, multiple MAC addresses (VMAC) can be defined for the adapter.

Example 1-3 shows a setup with two OSA-Express devices defined. The DEVICE that is called VIRTUALx (x=0-9,A-Z) is a virtual device where the portname field is used to reference the real OSA-Express device. Each is on a different subnet and the virtual OSA-Express device is using a manually defined VMAC address.

Example 1-3 Configuring virtual OSA devices in IPv6/VSE

```

DEVICE VIRTUAL0 OSAX 780 OSAX780 782 LAYER2 020000008001
LINK   VIRTUAL0 0 192.168.11.238 255.255.255.0 9000
ROUTE  VIRTUAL0 192.168.11.0 255.255.255.0 0.0.0.0 0
*

DEVICE OSAX780 OSAX 780 L2TEST 782 LAYER2
LINK   OSAX780 0 192.168.10.238 255.255.255.0 9000
ROUTE  OSAX780 192.168.10.0 255.255.255.0 0.0.0.0 0
*

ROUTE  OSAX780 0.0.0.0 0.0.0.0 192.168.10.1 1
*
```

By using this configuration, you can have multiple IP/VMAC addresses for each physical OSA-Express adapter. Each device (real or virtual) also can be on a different VLAN.

Linux Fast Path

Linux Fast Path (LFP) does not access any OSA device directly. Support for VMAC depends on Linux on System z. For more information about network configuration for Linux on System z, see the following resources:

- ▶ http://www.ibm.com/developerworks/linux/linux390/documentation_dev.html
- ▶ Chapter 9, “What you should know about the qeth device driver” of *Device Drivers, Features, and Commands (kernel 3.10)*, SC33-8411.

1.2.15 OSAX Hotswap support

IPv6/VSE allows you to define a spare or Hotswap device for each OSA Express adapter. If the primary OSA Express adapter fails, the Hotswap adapter and port take over operations without losing connections. Example 1-4 shows an OSAX Hotswap configuration.

Example 1-4 IPv6/VSE Hotswap support for OSAX

```

// EXEC BSTTINET,SIZE=BSTTINET,OS390,TASKS=ANY
ID 00 NAME OSAIPV4
INTERVAL 120
*

DEVICE OSAX620 OSAX 620 EXPRESS 622
LINK   OSAX620 0 10.1.1.239 255.255.255.0 9000
ROUTE  OSAX620 10.1.1.0 255.255.255.0 0.0.0.0 0
*

DEVICE OSAX610 OSAX 610 EXPRESS 612
LINK   OSAX610 0 192.168.1.239 255.255.255.0 1500
ROUTE  OSAX610 192.168.1.0 255.255.255.0 0.0.0.0 0
*

ROUTE  OSAX610 0.0.0.0 0.0.0.0 192.168.1.100 1
*
```

```

HOTSWAP OSAX610 0630 0632 0
HOTSWAP OSAX620 0640 0642 0
*
ATTACH TCP/IP
/*

```

The **HOTSWAP** command specifies the device name and cuu addresses of the read/write pair and the control cuu address. The port number to use is also specified. The cuu addresses must be specified as four character hexadecimal values.

If your primary **DEVICE** command is **DEVICE OSAX610 OSAX 610 portname 612**, you can use this **HOTSWAP** command:

```

HOTSWAP OSAX610 0630 0632 0

```

Note: Do not HOTSWAP to the same adapter.

Ideally, if you have a z box with two OSA Express adapters and two z/VSE images, you might use port 0 on each adapter as the primary NIC and use port 1 as the HOTSWAP NIC. For example:

```

Adapter 0, port 0 VSEPROD
Adapter 0, port 1 VSETEST HOTSWAP
Adapter 1, port 0 VSETEST
Adapter 1, port 1 VSEPROD HOTSWAP

```

Hotswap support requires IPv6/VSE build 255pre08. Example 1-5 shows the console log from the case where the primary OSAX adapter gets detached and the hotswap device immediately takes over control.

Example 1-5 IPv6/VSE Hotswap example

```

R1 0495 BSTT613I      OSAX610  Interface  1 IP address 192.168.1.239
R1 0495 BSTT600I      13 INITIATED  DEVOSAX  Build255 05/23/14 11.16
R1 0495 BSTT605I      13 DEVICE 01 OSAX610   TYPE OSAX   UNIT 0610
R1 0495 BSTT014I IJBOSA  LOADED A=80AB8000 L=00010CE8
R1 0495 BSTT010I IJBOSA DY47264 11318 145512L
R1 0495 BSTT717I IJBOSA FLAGS=20000000 VLAN=0000 RPLL=000C XPLL=000C
R1 0495 BSTT613I      OSAX620  Interface  2 IP address 10.1.1.239
R1 0495 BSTT600I      14 INITIATED  DEVOSAX  Build255 05/23/14 11.16
R1 0495 BSTT605I      14 DEVICE 02 OSAX620   TYPE OSAX   UNIT 0620
R1 0495 BSTT014I IJBOSA  LOADED A=80ADA000 L=00010CE8
R1 0495 BSTT010I IJBOSA DY47264 11318 145512L
R1 0495 BSTT717I IJBOSA FLAGS=20000000 VLAN=0000 RPLL=000C XPLL=000C
R1 0495 BSTT600I      15 INITIATED  rhomec   Build254 04/25/13 10.32
R1 0495 BSTT613I TCP/IP-TOOLS TCP/IP Stack Initialization Complete
R1 0495 BSTT601I      2 TERMINATED netstart
R1 0495 BSTT601I      15 TERMINATED rhomec
R1 0045 BSTT010I HOTSWAP OSAX610 0630 0632 0
R1 0496 BSTT011I COMMAND PROCESSING COMPLETE

```

*** cp detach nic 610**

```

AR 0015 NIC 0610 is destroyed; devices 0610-0612 detached

```

```

R1 0495 BSTT013E IJBOSA  ERROR R15=00000020 R0=0000007A R1=00AA7554
R1 0495 OS39I  ERROR DURING OSA EXPRESS PROCESSING,REASON=0030

```

```
CUU=0611
R1 0495 BSTT732W    13 HOTSWAP ACTIVATED OSAX610    0630 0632 0000
R1 0495 BSTT717I IJBOSA FLAGS=20000000 VLAN=0000 RPLL=000C XPLL=000C
```

You can virtualize the OSA Express adapter in your IOCP to allow many z/VSE images to share port 0 of the available adapters while using port 1 of different adapters as HOTSWAP devices.

1.3 Software options

This section describes the available software options, which include the use of different IP stacks (TCP/IP for VSE/ESA, IPv6/VSE, and Fast Path to Linux on System z Linux Fast Path) and IP protocols (IPv4, IPv6, and a mixture of both).

1.3.1 IPv4

Internet Protocol version 4 (IPv4) is the standard protocol for routing traffic through the Internet. IPv4 is described in IETF publication RFC 791 from September 1981. An IPv4 IP address is a number with 32 bits (4 bytes). The resulting limit of 2^{32} (approximately 4.3 billion) different addresses was one of the main reasons that lead to the development of IPv6.

1.3.2 IPv6

Internet Protocol version 6 (IPv6) is the successor to Internet Protocol version 4 (IPv4), which was the first protocol version that was used in the Internet. In “old economy” countries, such as Europe and North America, IPv4 is still used today. Emerging countries, such as South America, India, and China, are forced to use IPv6 because of the exhaustion of IPv4 addresses. A version 5 was developed experimentally, but was never publicly used.

Each IPv4 IP address is a 4-byte number, which provides a maximum of 2^{32} (about 4.3 billion) addresses. In the early 1990s, it became obvious that because of the rapid growth of the number of computers in the world, this number would be exceeded in the foreseeable future. On 3 February 2011, the Number Resource Organization (NRO) announced that the free pool of available IPv4 addresses was fully depleted.

An IPv6 IP address has 16 bytes, which allows for a total number of 2^{128} different addresses. This number is so large that all people alive in 2013 (approximately 7.0 billion) can each have billions of unique addresses. Therefore, it is likely that the pool of IPv6 addresses is never going to be used up.

IPv6 addresses often are written as eight groups of four hexadecimal digits (each group representing 16 bits, or 2 bytes), where each group is separated by a colon (:), as shown in the following example:

```
2001:0db8:85a3:08d3:1319:8a2e:0370:7344
```

Leading zeros in a group can be omitted (but at least one digit per group must be left), so the following notations can be used for the same address:

- ▶ 2001:0db8:0000:08d3:0000:8a2e:0070:7344
- ▶ 2001:db8:0:8d3:0:8a2e:70:7344

A string of consecutive all-zero groups can be replaced by two colons. To avoid ambiguity, this simplification can be applied only once, so again, the following notations refer to the same address:

- ▶ 2001:db8:0:0:0:0:1428:57ab
- ▶ 2001:db8::1428:57ab

The IPv6 protocol is described in Internet standard document RFC 2460, which was published in December 1998.

1.3.3 Why IPv6?

Consider the following points as you decide whether to upgrade your networking environment to IPv6:

- ▶ Your Internet service provider (ISP) upgrades to IPv6.
- ▶ It might happen that your customers or business partners are only reachable by using IPv6 (for example, China).
- ▶ Governmental organizations might allow only manufacturers of IPv6 capable-products and applications to participate in advertised biddings. For example, the US Department of Defense (DoD) allows only products that are on the “Unified Capabilities Approved Products List” (IBM UC™ APL) for its advertised biddings.
- ▶ No Network Address Translation (NAT). Because every device in the world can have a globally unique IPv6 address, NAT becomes unnecessary.

1.3.4 Dual stack support

Dual stack means that a host can run IPv4 and IPv6 applications at the same time. Therefore, hosts simultaneously reach IPv4 and IPv6 destinations, which makes dual stack the most popular coexistence strategy.

Dual stack includes the following benefits:

- ▶ Native dual stack does not require any tunneling mechanisms on internal networks.
- ▶ IPv4 and IPv6 run independent of each other.
- ▶ Dual stack supports gradual migration of endpoints, networks, and applications.

IPv6/VSE has two TCP/IP stacks: an IPv4 stack and an IPv6 stack. These TCP/IP stacks can be run individually, together (COUPLED), or stand alone.

1.3.5 Migration from IPv4 to IPv6

Contrary to popular belief, IPv6 is not compatible with an earlier version, but IPv4 and IPv6 networks can be used concurrently over the same cable and with the same endpoint.

The following transition methods are available:

- ▶ Dual IP Stacks

This is the easiest method. The IP stack supports both protocols concurrently. Examples are Linux since Kernel 2.6 and Windows since XP SP1. Existing IPv4 applications can continue to run unchanged. Applications can be IPv6-enabled over time, one after the other.

- ▶ Tunneling

IPv6 packets are sent as payload of other protocols (usually IPv4) to a tunneling broker, which is in an IPv6 network. The broker extracts the IPv6 packet from the payload and sends it as IPv6 packet through IPv6 routing to the final destination. An example is “6in4” that uses Tunneling-Broker.

The following infrastructure components must be upgraded:

- ▶ Layer 1 devices (for example hubs), which are not apparent for IPv6.
- ▶ Layer 2 devices (switches) that were purchased within the last 10 years most likely support IPv6 already.
- ▶ Layer 3 devices (routers), which often are not required for local LANs. Today, most router manufacturers provide IPv6-capable routers. Routers that use Multiprotocol Label Switching (MPLS) are protocol-independent.
- ▶ Endpoints (PCs, Server). Most modern operating systems support IPv6.
- ▶ Applications might need to be adapted (IPv6-enabled) to work with IPv6 addresses.

1.3.6 IPv6 products for z/VSE

In this section, we describe the available IPv6 products for z/VSE.

IPv6/VSE

IPv6/VSE is a registered trademark of Barnard Software, Inc. (BSI). The IPv6/VSE V1 product is designed to provide the following IPv6 solution for z/VSE:

- ▶ z/VSE users can participate in an IPv6 network.
- ▶ It brings the benefits of IPv6 functionality to z/VSE users.
- ▶ It helps z/VSE users to meet the requirements of the commercial community and governmental agencies, thus it fulfills the statement of direction in Software Announcement 209-319, dated October 20, 2009.

IPv6/VSE V1 is designed to provide an IPv6 TCP/IP stack, IPv6 application programming interfaces (APIs), and IPv6-enabled applications.

The IPv6/VSE product also includes a full-function IPv4 TCP/IP stack, IPv4 application programming interfaces, and IPv4 applications. The IPv4 TCP/IP stack does not require the IPv6 TCP/IP stack to be active.

IPv6/VSE V1 supports the IPv6 and IPv4 protocols, whereas TCP/IP for VSE/ESA V1.5 supports the IPv4 protocol only.

LFP

LFP is part of z/VSE since z/VSE 4.3. LFP is IPv6-enabled since z/VSE 5.1.

1.3.7 Securing your connections with Secure Sockets Layer

Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) are cryptographic protocols that provide network security. There are several versions of these protocols in use by all kinds of applications, such as web browsers, FTP clients and servers, Telnet, and instant messaging.

Up to now, the supported protocol versions on z/VSE were SSL 3.0 (which is sometimes referred to as SSL 3.1) and TLS 1.0.

There already is a push towards TLS 1.2, which has several advantages to previous versions. Most important is the increased number of cipher suites, including the SHA-2 family of hash algorithms. Earlier protocol versions support only SHA-1, which today is considered broken. Various sources about cryptographic attacks against SHA-1 in 2005 and following years are available.

APAR DY47499 provides support for TLSv1.2 with OpenSSL 1.0.1e for z/VSE 5.1. For more information about how to use this new support, see 3.7.15, “Using TLSv1.2” on page 110.

Table 9 on page 13 of NIST Special Publication *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*, 800-131A, shows that the use of the SHA-1 hash function was disallowed after December 31, 2013, except for non-digital signature applications. This document is available at this website:

<http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf>

Because the new NIST SP800-131a standard was mandatory by the end of 2013, IBM products with cryptographic functionality perform their updates immediately to give customers time to test and upgrade their applications to become compliant.

1.3.8 Options for printing

When setting up for printing, you have several choices available.

Line printer requester (LPR)/Line printer daemon (LPD) is a platform-independent printing protocol for remote printing with which multiple platforms can print to the same printer without any extra configurations. This requires an LPD on the platform where the printer is connected.

TN3270E provides printing to workstation printers without the need for an LPD. However, it requires an emulator program with TN3270E support. With TN3270E, the 3270 print data stream is passed unchanged to the TN3270E printer client software that is running on a remote workstation. It is the function of the TN3270E printer client to translate the data and adjust the print output including font and pitch.

A TN3270E server has more than 3270 terminal display capabilities. It can also support the SNA print data stream. It accepts the print data stream from the application and forwards it to the TN3270E client that is running on the workstation. The workstation can then print the data by using normal workstation printer facilities.

Table 1-2 shows the printing options that are provided by TCP/IP for VSE/ESA and IPv6/VSE.

Table 1-2 Printing options by stack

	TCP/IP for VSE/ESA	IPv6/VSE
LPR	Yes	Batch LPR and Auto LPR
LPD	Yes	Yes
General Printer Server (GPS)	Yes	LPR printer sessions (these are the same as GPS printers)
TN3270E	No	TN3270E printer sessions (both SNA and non-SNA)
DIRECT	No	DIRECT printer sessions that use port 9100
FTP	No	FTP print output to FTP server or printer

1.3.9 Overview of APIs

Table 1-3 provides an overview of the socket APIs that are available for z/VSE and where they are documented.

Table 1-3 Overview on APIs

API	Programming languages	Standard/Proprietary	SSL support	IPv6 support	Stack support	Doc ^a
LE/C Socket API	C	Standard BSD Sockets, mostly compatible with IBM z/OS and other distributed systems (UNIX, Linux, Windows, Mac, ...)	Yes	Since z/VSE 5.1, IPv6/VSE and LFP only	TCP/IP for VSE/ESA, IPv6/VSE, LFP	1
EZA APIs ► EZASMI ► EZASOCKET	Assembler COBOL, PL/I, Assembler	Standard BSD Sockets, mostly compatible with z/OS	Yes	Since z/VSE 4.2 with IPv6/VSE, since z/VSE 5.1 with LFP, too	TCP/IP for VSE/ESA, IPv6/VSE, LFP	1
CSI SOCKET Macro	Assembler	CSI proprietary	No	No (IPv6/VSE and LFP support IPv6 extensions)	TCP/IP for VSE/ESA, BSI*, LFP* *) Not all features supported	2
CSI Preprocessor API (EXEC TCP)	COBOL, PL/I, Assembler	CSI proprietary	No	No	TCP/IP for VSE/ESA (BSI provides conversion tool for EXEC TCP to EZA for COBOL programs)	2
CSI BSD Socket API	C, Assembler (COBOL, PL/I)	BSD-like socket API with limited standard compatibility CSI proprietary	Yes	No	TCP/IP for VSE/ESA	2
IBM REXX Sockets	REXX	Standard BSD Sockets	Yes	No	TCP/IP for VSE/ESA, BSI, LFP (internally uses LE/C Sockets)	3
CSI REXX Sockets	REXX	CSI proprietary	No	No	TCP/IP for VSE/ESA	2

a. The numbers that are used in the Doc column correspond to the following publications:

1. *z/VSE TCP/IP Support*, SC34-2640, which is available at this website:

<http://www.ibm.com/systems/z/os/zvse/documentation/#tcpip>

2. *TCP/IP for VSE Programmer's Guide*, which is available at this website:

<http://www.csi-international.com>

3. *REXX/VSE Reference*, SC33-6642, which is available at this website:

<http://www.ibm.com/systems/z/os/zvse/documentation/#rexx>

1.3.10 Available applications

In this section, we describe the applications that are available for the different IP stacks.

IBM

IBM provides the following applications, which can be used with all IP stacks:

- ▶ VSE Connector Server
- ▶ IBM CICS® Web Support (CWS)
- ▶ DB Call Level Interface (CLI)
- ▶ VSAM Redirector
- ▶ Web Services (SOAP)
- ▶ IBM WebSphere® MQ for z/VSE
- ▶ z/VSE SNMP Monitoring Agent
- ▶ z/VSE SNMP Trap Client
- ▶ IBM DB2® Server / Client
- ▶ z/VSE Virtual FTP daemon

The LFP stack supports IPv6 since z/VSE 5.1, but the applications currently are not IPv6-enabled. The Virtual z/VSE FTP Daemon is a Java application and is IPv6-capable. However, it uses the VSE Connector Server as the back end; therefore, if the VSE Connector Server does not support IPv6 addresses, the FTP daemon has the same restriction.

TCP/IP for VSE/ESA

The following applications are part of the TCP/IP for VSE/ESA product:

- ▶ File Transfer Protocol (FTPD, FTPBATCH, interactive FTP client in CICS, Auto-FTP)
- ▶ Telnet TN3270 Server (TELNETD)
- ▶ Email
- ▶ LPR client for printing
- ▶ General Printer Server (GPS)
- ▶ Web Server (HTTPd)
- ▶ TLS daemon (TLSD)
- ▶ PDF conversion
- ▶ Automation daemon for automatically sending Power LST queue entries by using FTP, email, or LPR
- ▶ REXEC client for sending commands to a remote REXEC daemon
- ▶ Ping, Traceroute, Discover, and other tools to analyze the network from VSE

A structured file system is also provided that allows transparent access to SAM (Disk and Tape), VSAM (ESDS and KSDS), Power (RDR PUN and LST queues), Librarian, VTOC,

HFS, ICCF, Epic managed, Data Spaces, and user-defined file I/O drivers from all these applications.

Applications that are provided with TCP/IP for VSE/ESA currently are not IPv6-enabled.

IPv6/VSE

Barnard Software Inc. provides the following applications with the IPv6/VSE product:

- ▶ FTP server (IBM VSE/POWER®, VSAM, SAM, LIBR, Tape, Userexit)
- ▶ Batch FTP Client
- ▶ TN3270E server (TN3270/TN3270E Terminal & TN3270E Printer Sessions)
- ▶ Network Time Protocol Server (NTP server)
- ▶ Network Time Protocol Client (NTP client)
- ▶ System Logger Client
- ▶ Batch Email Client
- ▶ Batch LPR
- ▶ LPD
- ▶ Batch Telnet
- ▶ Batch Remote Execution client (REXEC)
- ▶ Batch PING
- ▶ GZIP data compression
- ▶ REXX automation
- ▶ PDF Generation by using VSE2PDF/Lite
- ▶ BSTTPRXY SSL Proxy Server
- ▶ BSTTATLS Automatic TLS facility

All applications that are provided with IPv6/VSE are IPv6-enabled and work with LFP and TCP/IP for VSE/ESA stacks.

1.3.11 Choosing a socket API when designing your applications

When you are designing an application that uses network communications, you should carefully choose the correct socket application. The available socket APIs differ not only in functionality, but in support when they are used with different stack versions or vendors. Because of this issue, you should use one of the following standard conforming APIs that are provided by IBM:

- ▶ EZA APIs (EZASMI and EZASOCKET)
- ▶ LE/C Socket API
- ▶ IBM REXX Sockets

These APIs work with all available TCP/IP stacks. Also, these APIs follow the Berkeley Sockets standard (BSD). The Berkeley Sockets API was introduced in the early 1980s for the UNIX operating system and was the de facto standard for many years. Meanwhile, the BSD API is part of the Portable Operating System Interface (POSIX) specification.

Technically, the IBM socket APIs separate the API layer from the implementation. That way, the API can be kept stable, but the implementation can be exchanged when different TCP/IP Stacks are used.

Figure 1-1 shows that the LE/C and the EZA APIs consist of a stack independent API interface part and a stack-dependent implementation that is provided with the stack.

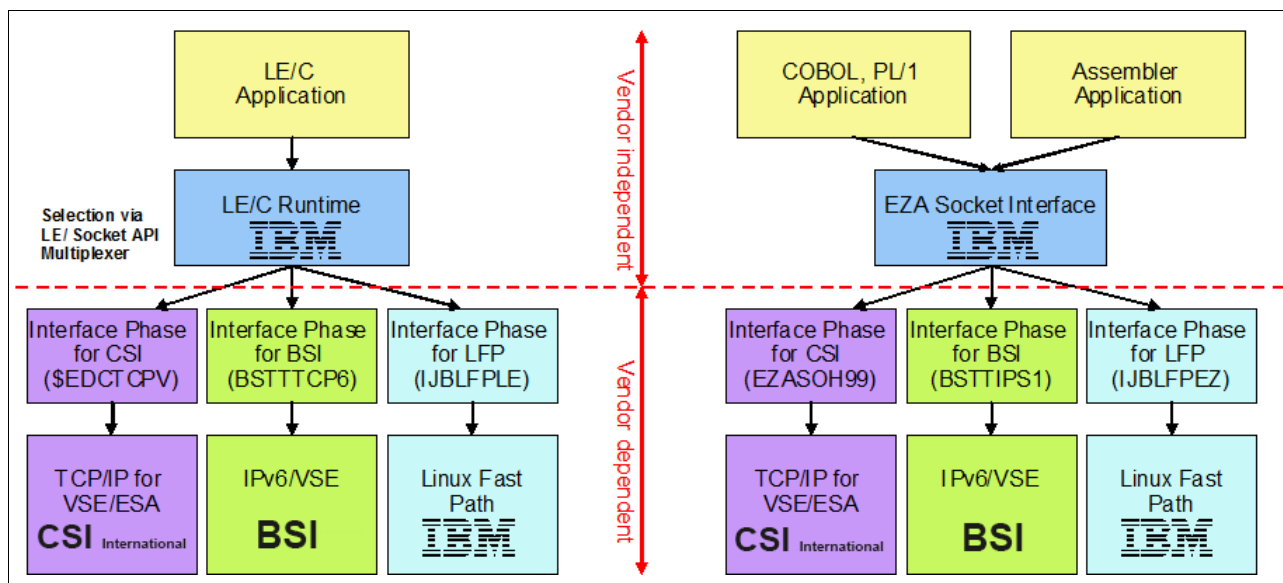


Figure 1-1 Vendor-related parts of socket APIs

If you choose a proprietary Socket API that is supported by only one specific TCP/IP stack, you are bound to this stack version or vendor. When you are migrating to another TCP/IP stack, you might have to change your applications.

1.3.12 Enabling your applications for IPv6

You must adapt your applications to enable them to support IPv6. This is necessary because the application must work with longer IPv6 addresses (16 bytes instead of 4 bytes with IPv4). There also are some other socket functions that are available that must be used with IPv6. Some of them are replacements for older socket functions that work only with IPv4 addresses. Some of the new functions are optional, but also can be used with IPv4 addresses.

Note: Not all APIs support IPv6. For more information about available APIs and their IPv6 support status, see Table 1-3 on page 21.

In the following sections, we describe some general considerations for when you are enabling applications for IPv6. We also describe hints and tips for how to design your code for best compatibility with various TCP/IP stacks and vendors.

Changes in the LE/C and EZA Socket API to support IPv6

The following socket calls now support AF_INET6 (in addition to AF_INET):

- ▶ socket: allocate an IPv6 socket
- ▶ getclientid: Get the callers ID (needed for givesocket and takesocket)
- ▶ givesocket: Give a socket to another thread, task, or process
- ▶ takesocket: Take a socket from another thread, task, or process
- ▶ gethostbyaddr: Get domain and alias names of an address

The following functions now also accept a `sockaddr_in6` structure:

- ▶ `bind`: Bind a socket to a local address (IP and port)
- ▶ `connect`: Establish a connection to a foreign address
- ▶ `getsockname`: Get the socket's local address
- ▶ `getpeername`: Get the address of the communication partner
- ▶ `accept`: Accept a new client and get its foreign address
- ▶ `sendto`: Send a message to a foreign address (typically used for UDP)
- ▶ `recvfrom`: Receive a message and get the senders address (typically used for UDP)

The following new functions were added to support IPv6:

- ▶ `getaddrinfo`: Translate a domain name or service name into a socket address (replaces `gethostbyname`)
- ▶ `getnameinfo`: Translate a socket address into a domain name or service name (replaces `gethostbyaddr`)
- ▶ `freeaddrinfo`: Frees information that is returned by `getaddrinfo`
- ▶ `inet_ntop`: Convert a binary address into its textual form (replaces `inet_ntoa`)
- ▶ `inet_pton`: Convert a textual IP address (IPv4 or IPv6) into a binary address (replaces `inet_addr`)

IPv6 address handling

Internally, IPv4 addresses are represented by 32 bits (4 bytes) whereas IPv6 addresses are represented by 128 bits (16 bytes). Externally, IPv4 addresses are displayed in the format `a.b.c.d` where `a - d` are decimal numbers 0 - 255, which are separated by a period.

IPv6 addresses are displayed in the format `aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh`, where `aaaa - hhhh` are hexadecimal, 2-byte blocks that are separated by a colon. (Leading zeros can be omitted in each block.) You also can replace consecutive zeros with a double colon once per IPv6 address.

Because converting binary IPv6 addresses into their presentation format and vice versa is not as easy as for IPv4 addresses, you should not try to implement the conversion algorithm on your own. Instead, you should use the following API functions that are provided by the Socket API:

- ▶ `NTOP (EZA)`, `inet_ntop (LE/C)`: Converts binary IP addresses from their network format to their (textual) presentation format.
- ▶ `PTON (EZA)`, `inet_pton (LE/C)`: Converts IP addresses from their (textual) presentation format to their network format (binary).

Note: These functions work for IPv6 and IPv4 addresses.

In addition to these functions, the socket functions `GETADDRINFO` and `GETNAMEINFO` can be useful to convert textual IP addresses and host names to binary IP addresses and vice versa.

The length of a field that is holding a textual IPv4 address must be at least 15 characters long ($4 * 3 \text{ digits} + 3 \text{ periods}$). The length of a field holding a textual IPv6 address must be at least 39 characters long ($8 * 4 \text{ hex digits} + 7 \text{ colons}$).

Specifying port numbers as part of an IPv6 address

Some existing client applications allow specifying a port number as part of the IP address or host name. Usually, a colon is used to separate the IP address or host name from the port number; for example, 1.2.3.4:80 or ibm.com:80.

This configuration works fine for IPv4 addresses and host names. However, it does not work for IPv6 addresses because IPv6 addresses contain colons. Appending only the port number that is separated by another colon is ambiguous. Therefore, IPv6 addresses are encapsulated in square brackets and the port number is appended with a colon; for example, [2000:123::4]:80.

For more information, see *Domain Names - Implementation and Specification*, RFC 2732, which is available at this website:

<http://www.rfc-base.org/rfc-1035.html>

The problem with square brackets in z/VSE is that these characters are on different code page positions, depending on which code page is used. Users might not be able to type the square brackets in the same code page that the code uses to parse the data. Thus, the code might not recognize the square brackets and shows errors.

To avoid such code page problems, angle brackets (< ... >) can be used as an alternative; for example, <2000:123::4>:80. Some implementations also use single quotation marks instead of brackets; for example, '2000:123::4':80. However, because single quotation marks are often used by JCL and other parsers to enclose parameters, the use of single quotation marks might not be the best way for z/VSE.

To have a common look and feel, all IPv6-enabled z/VSE applications should follow the following rules if they must specify port numbers as part of IP addresses:

- ▶ They must accept IPv6 addresses to be enclosed in angle brackets (< .. >).
- ▶ They can accept IPv6 addresses to be enclosed in squared brackets ([..]), where code page 1047 should be used.
- ▶ They can accept IPv6 addresses to be enclosed in single quotation marks (' .. ').
- ▶ If an IPv6 address does not contain a port specification, it might not be enclosed, but it can be.
- ▶ IPv4 addresses and host names must not be enclosed.

IPv6 addresses as input

Typically, IP addresses are entered by users at several places in z/VSE applications that use network communications.

All places where IP addresses are used as input should follow the following rules:

- ▶ Enable entering an IPv4 address, IPv6 address, or a host name.
- ▶ You should not use different fields for IPv4 address, IPv6 address, or host name. Instead, one single field should be used, which allows specification of the three options.
- ▶ The field should be long enough to hold a complete host name of 255 characters. For more information, see RFC 1035, which is available at this website:

<http://www.ietf.org/rfc/rfc2732.txt>

When a URL is expected as input, the field must be long enough for 255+1+5 characters (host:65535).

- ▶ The user should not have to specify separately whether the entered data is an IPv4 address, IPv6 address, or host name. Instead, the underlying code should detect that status on its own based on the entered data (which can be done by using PTON or GETADDRINFO functions).
- ▶ Interfaces should not have preformatted input fields for IP addresses (such as, four fields with periods in between for entering IPv4 addresses). This setup avoids problems with copy and paste from other sources.

Note: The preferred method of specifying a target host address should be the use of host names. Host names are independent of IPv4 or IPv6, which makes IPv4-to-IPv6 transitions much easier for the user.

IPv6 addresses as output

IP addresses are often part of output, such as messages and listings. The textual presentation format of such addresses should be constructed by using the NTOP Socket API function (at least for IPv6 addresses). Alternatively, you can format the IP addresses on your own (for example, if the NTOP functions not available). In this case, you should print all IPv6 2 byte blocks with full 4 hex digits. Do not try to shorten the IPv6 address by skipping leading zeros or omitting zero blocks.

Enabling client/server applications for IPv6

IPv6-enabled applications should work with IPv4 and IPv6 sockets concurrently. Thus, you should not design your application to work with IPv6 only. IPv6 enablement always means to support IPv6 and IPv4.

An IPv6-enabled client application should take an IPv4 address, IPv6 address, or a host name as input and detect at run time whether to allocate an IPv4 socket or an IPv6 socket to connect to the remote server.

An IPv6-enabled server application should accept IPv4 clients and IPv6 clients concurrently. Server applications might have an option to restrict clients to IPv4 or IPv6 only; however, in general, a server should handle both.

Because old (IPv4 only) TCP/IP versions might still be used at your site, your application must still work with an IPv4 stack that does not support the new IPv6 specific API functions (NTOP, PTON, GETNAMEINFO, GETADDRINFO, and FREEADDRINFO). If you call an IPv6-specific function, you receive an error code even though the stack your application is using does not support IPv6. Your code must have a fallback code path to use only IPv4-specific functions.

Consider the following situations:

- ▶ IPv4-only stack (old CSI or old BSI stack): No support of IPv6 and no support of IPv6 specific functions. Use fallback code path by using old IPv4 specific functions only.
- ▶ IPv6 only stack (BSI IPv6/VSE without a coupled IPv4 stack): All functions are available (including old IPv4 specific functions), but you cannot allocate an IPv4 socket (AF_INET). IPv6 sockets only (AF_INET6). Mapped IPv4 addresses are not supported.
- ▶ True dual-mode stack (Linux Fast Path, BSI IPv6/VSE coupled with full IPv4 stack): All functions are available. You can allocate IPv4 (AF_INET) and IPv6 (AF_INET6) sockets. Mapped IPv4 addresses can also be used.

To make IPv6 enabling easier, we describe best practices for how to IPv6 enabling client and server applications in the following sections. Sample code (LE/C) also is available to show the call flow.

Enabling client applications for IPv6

An IPv6-enabled client application should take an IPv4 address, IPv6 address, or a host name as input and detect at run time whether to allocate an IPv4 socket or an IPv6 socket to connect to the remote server.

The client application also should detect if it is working with an IPv6 capable stack. Depending on the combination of input address family (IPv4, and IPv6) and stack capabilities, different code paths are run.

The following general flow is used:

- ▶ Try to use GETADDRINFO with user-provided destination address (IPv6 address, IPv4 address, or host name)
- ▶ If return code is EAI_NONAME, the error: host not found is received.
- ▶ If return code is not zero, assume for an IPv4 only stack:
 - Use `inet_addr` to convert IPv4 address to binary
 - If there is an error, assume that the host name is specified and use `gethostbyname` to resolve the host name issue
 - Set the address family to `AF_INET`
- ▶ Else (return code is zero), use first result.
- ▶ Allocate a socket with wanted address family (from ADDRINFO result)
- ▶ If the error is EAFNOSUPPORT, assume for IPv6 only stack:
 - Build mapped IPv4 address
 - Allocate `AF_INET_6` socket
- ▶ Connect to wanted address
- ▶ Proceed with other processing on the socket

Figure 1-2 shows this flow of control.

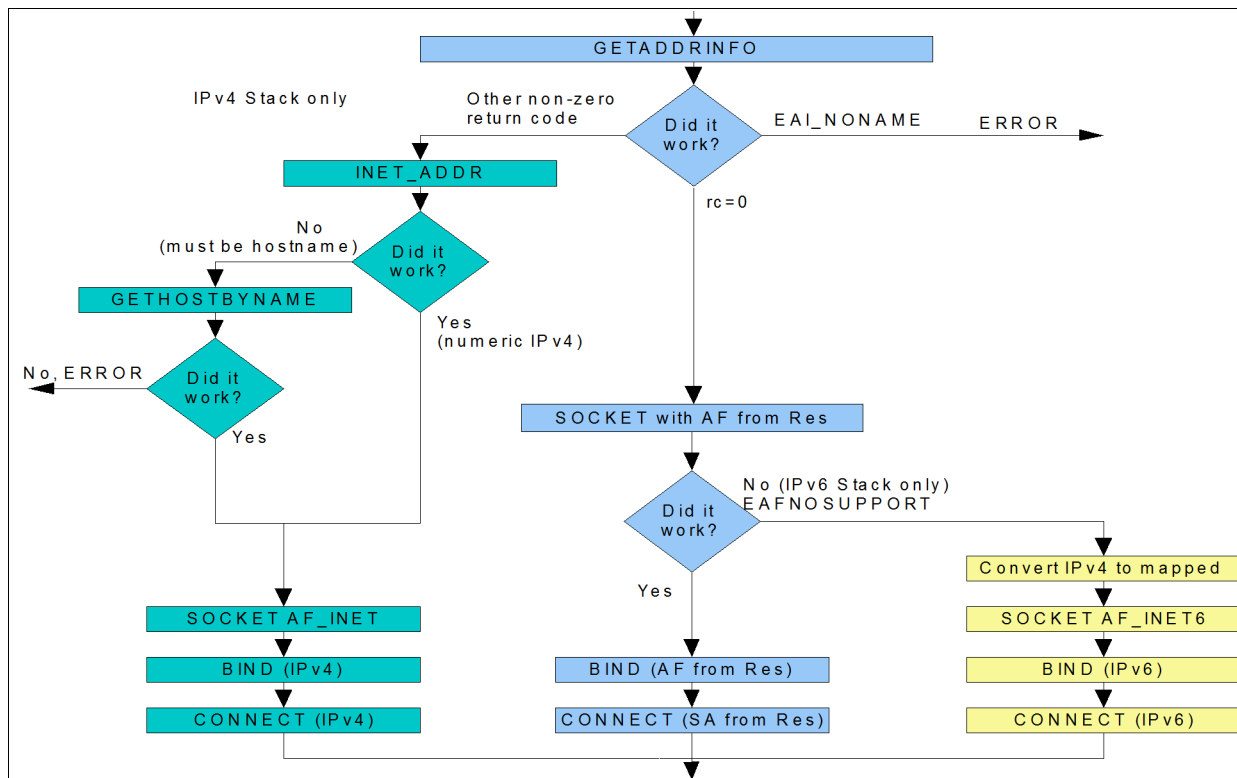


Figure 1-2 Enabling client applications for IPv6

Enabling server applications for IPv6

An IPv6-enabled server application should accept IPv4 clients and IPv6 clients concurrently. Server applications might have an option to restrict clients to IPv4 or IPv6 only; however, in general, they should handle both.

The server application should detect if a client is working with an IPv6 capable stack. Depending on the combination of the stack capabilities and the client's address family (IPv4 or IPv6), different code paths are run.

The general flow is as follows:

- ▶ Try to allocate a listening socket with AF_INET6.
- ▶ If an error is received, assume IPv4 stack only: Allocate an AF_INET socket instead.
- ▶ Bind the socket to ANY address (IPv4 or IPv6) plus listening port.
- ▶ Set the socket into listening mode.
- ▶ Accept new clients (use select to wait for listening socket ready, if needed, then call accept).
- ▶ Inspect the address family of the newly accepted client:
 - If AF_INET6 (IPv6):
 - If mapped IPv4 address, convert to IPv4 address and display IPv4 address (use inet_ntop or inet_ntoa to convert into presentation format).
 - Else display IPv6 address (use inet_ntop to convert into presentation format).

- If AF_INET (IPv4): Display IPv4 address (use `inet_ntoa (!)` to convert into presentation format).
- Proceed with other processing on the socket and return to accept new clients.

Figure 1-3 shows this flow of control.

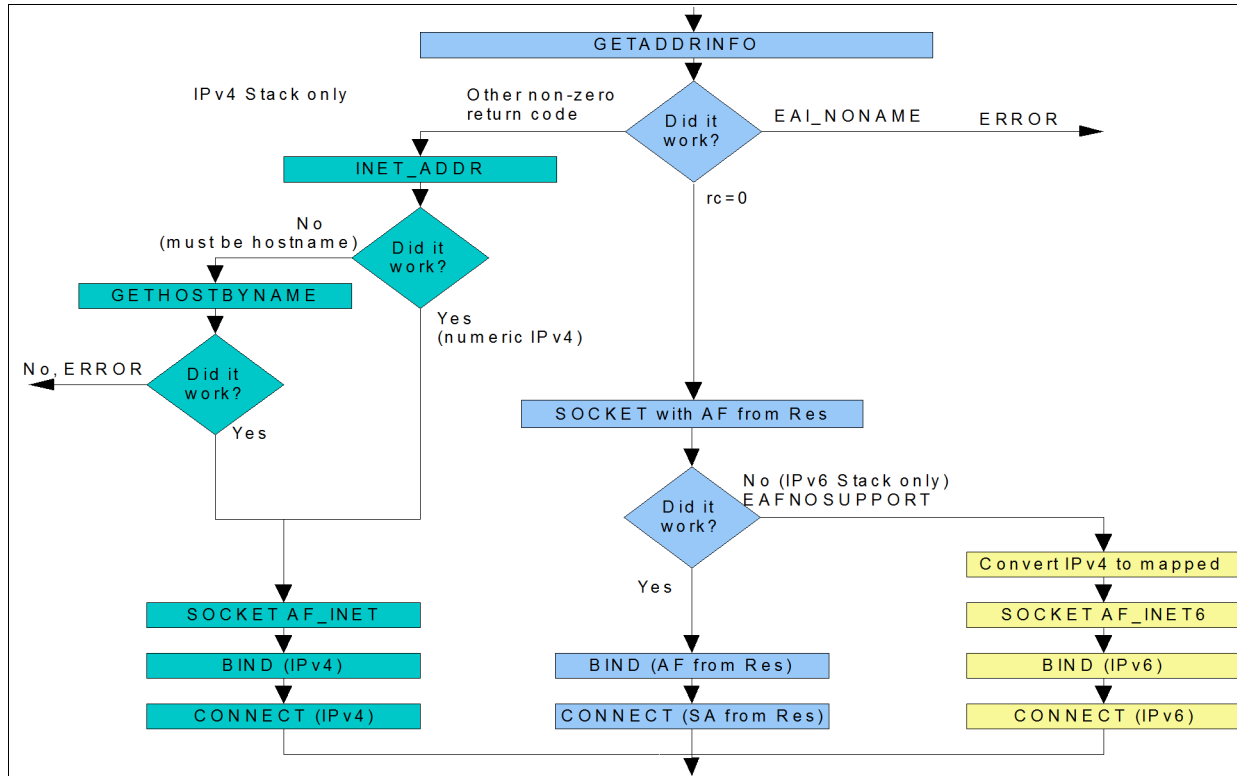


Figure 1-3 Enabling server applications for IPv6

Detecting the TCP/IP stack on which you are running

Normally, an application should not need to identify the TCP/IP stack it is using. However, you can use the `getibmopt` socket call (with LE/C and EZA socket APIs) to perform this detection.

On a stack, the `getibmopt` call might be supported. If the call fails (bad return code), assume that you are running with a CSI stack.

If it works, you can inspect the returned information, as shown in the following example:

```

TCP-IMAGE-NAME is 'SOCKETnn'
TCP-IMAGE-STATUS:
X'80vv': Active
X'40vv': Terminating
X'20vv': Down
X'10vv': Stopped or stopping
  
```

In this example, vv indicates with which stack you are running, as shown in the following example:

```

00: CSI
01: BSI IPv4
02: BSI IPv6
03: IBM LFP
  
```

1.4 Known problems

In this section, we describe some of the issues and insights we had in our test setup.

1.4.1 ERROR DURING OSA EXPRESS PROCESSING,REASON=002C CUU=nnnn,RETCODE=E00A

The following problem occurs when starting an OSAX link.

Symptom

These messages are issued when starting an OSAX link.

```
Y1 0114 0S39I ERROR DURING OSA EXPRESS PROCESSING,REASON=002C
      CUU=nnnn,RETCODE=E00A
```

```
Y1 0112 0005: IPL605E Unable to Initialize IJBOSA, return code: 121
```

```
Y1 0112 0005: IPL609E Unable to initialize OSA Express, Link: CHPIDC0
```

Possible reason

Return code E00A says that the IP address of this OSAX adapter is already registered with another adapter.

Possibly you defined two OSAX links with the same IP address for failover reasons. The IP address remains registered on the second OSAX adapter even when the second link is stopped. The STOP LINKID command does not remove the IP address from the IJBOSA device driver.

The only solution is a DELETE LINK / DEFINE LINK.



TCP/IP for VSE/ESA

TCP/IP for VSE was developed by Connectivity Systems International, Inc. (CSI) and is part of z/VSE as TCP/IP for VSE/ESA. The product can be licensed directly from CSI or from IBM as part of z/VSE.

For more information about configuring TCP/IP for VSE/ESA, setting up FTP, TN3270, and SSL, see *Security on IBM z/VSE*, SG24-7691, which is available at this website:

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

This chapter provides an overview of TCP/IP for VSE/ESA 1.5F and the latest information that is available for z/VSE 5.1. We also describe special networking options.

This chapter includes the following topics:

- ▶ Overview
- ▶ Standard features
- ▶ Other optional features
- ▶ Applications that are provided with TCP/IP for VSE
- ▶ Application programming interfaces
- ▶ Setting up and running TCP/IP for VSE
- ▶ FTP hints
- ▶ Partition priorities
- ▶ Security
- ▶ Remote running with REXX
- ▶ Version checking
- ▶ Datagram analysis
- ▶ Known problems

2.1 Overview

The CSI TCP/IP for VSE product was the first solution on the market for connecting a production z/VSE system to today's Internet world. It provides a comprehensive and flexible set of interfaces and applications to participate fully in the global business arena. This includes many optional and third-party products that bring the full power and reliability of mainframe computing to the e-commerce world. For more information about CSI, see this website:

<http://www.csi-international.com/>

2.2 Standard features

TCP/IP for VSE runs on VSE/ESA 2.1 through to the latest z/VSE versions. It provides its own pseudo task manager within a single partition to support a virtually unlimited number of connections without the use of up standard VSE partitions and subtasks.

TCP/IP for VSE also includes the following features:

- ▶ Support for almost any hardware network connection.
- ▶ Extensive monitoring and tuning capabilities that can be extended to any level of granularity, including specific destination addresses.
- ▶ Customizable security features to protect network access and data encryption.
- ▶ Built-in connection management provides queuing and automatic detection and clean-up of dead connections.
- ▶ A complete set of servers, including File Transfer Protocol (FTP), LPD, HTTPd, and TELNETD for TN3270 access.
- ▶ Other support software, such as PDF creation.

2.3 Other optional features

TCP/IP for VSE provides the following optional features:

- ▶ General Print Services (GPS) enables CICS and IBM VTAM 3270 applications to print directly to TCP/IP based printers.
- ▶ SSL and Transport Layer Security (TLS) provides support for secure connections the use modern cryptography by using RSA with x509v3 certificates for authentication, triple-DES and AES for bulk data encryption, and SHA for secure hashes and digital signatures.
- ▶ SecureFTP provides user authentication, confidentiality, and data integrity by using digitally signed certificates, data encryption, and secure hash functions for VSE-based FTP servers and clients.
- ▶ SeeTCPIP is a system and network monitor that helps identify and correct system bottlenecks to improve system performance.

2.4 Applications that are provided with TCP/IP for VSE

TCP/IP for VSE provides the following applications:

- ▶ FTP
- ▶ Telnet TN3270 Server
- ▶ Email
- ▶ LPR client for printing
- ▶ Web server
- ▶ PDF conversion
- ▶ Automation daemon for automatically sending Power LST queue entries by using FTP, email, or LPR
- ▶ REXEC client for sending commands to a remote REXEC daemon
- ▶ Ping, Traceroute, Discover, and other tools to analyze the network from VSE

A structured file system is also provided that allows access to SAM (disk and tape), VSAM (ESDS and KSDS), Power (RDR PUN and LST queues), Librarian, VTOC, HFS, ICCF, Epic managed, and data spaces, from all these applications.

2.5 Application programming interfaces

You can create your own TCP/IP applications with the APIs that best suit your company's programming expertise. When possible, use the following APIs that are provided by IBM:

- ▶ IBM EZASMI macro and the EZASOKET call interfaces can also be used seamlessly with TCP/IP for VSE
- ▶ IBM LE/VSE C Run-Time Socket API
- ▶ SOCKET MACRO interface for Assembler programs

In addition to these APIs, you can use the following CSI interfaces:

- ▶ Basic socket interface (BSD Sockets) for C, Cobol, Assembler, or any other language that supports standard call and save linkage.
- ▶ Enhanced Pre-Processor is a language-independent API that is similar to the IBM CICS EXEC interface.
- ▶ Socket interface for REXX programming.
- ▶ SSL/TLS callable interface with which you can create secure socket client and server applications on VSE.
- ▶ Cryptographic algorithms, such as RSA, DES, AES, or SHA-1 can be called by using the CryptoVSE API for usage in any VSE application.
- ▶ The Common Encryption Cipher Interface with which you can encrypt and securely store data on any disk or tape device that is attached to VSE.

2.6 Setting up and running TCP/IP for VSE

Setting up and running TCP/IP is a relatively simple process. A single partition must be dedicated with at least 20 megabytes of virtual storage. The JCL for starting a sample TCP/IP partition is shown in Example 2-1.

Example 2-1 JCL to start the TCP/IP stack

```
* $$ JOB JNM=TCPIP15F,DISP=K,CLASS=8
* $$ LST CLASS=F,DEST=(,AFTPTODS),DISP=D RBS=40
// JOB TCP15F00 TCP/IP STARTUP JOB FOR ZVSE51DS
// OPTION LOG,NOSYSDDMP
// LIBDEF PHASE,SEARCH=(CSILIB.TCPT15F,CSILIB.TCPIP15F)
// LIBDEF SOURCE,SEARCH=(CSILIB.ZVSE51DS)
// SETPFIX LIMIT=(512K,2048K)
// EXEC IPNET,SIZE=IPNET,PARM='ID=00,INIT=TCP15F00',DSPACE=8M
/*
/&
```

This example should run in the F8 partition. The * \$\$ LST card allows SYSLST output to be automatically transferred by using FTP based on commands that are contained in a AFTPTODS.L library member.

TCP/IP stores all configuration files and command scripts in a library.sublibrary with the .L member type.

The // EXEC card contains the following parameters that control running the TCP/IP:

► PARM='ID=00'

This sets the stack identifier. Stack ID 00 is the default and should be used by most installations. Multiple stacks can be run by specifying an ID= 00 - 99. Therefore, the maximum number of stacks that can be run on a single VSE image is 100.

► INIT=TCP15F00

The INIT= tells IPNET to read the librarian member TCP15F00.L and run the commands that are contained in it (TCP15F00.L). A job similar to Example 2-2 can be used to create this librarian member.

Example 2-2 JCL to catalog a TCP/IP startup member

```
// JOB TCP15F00 CATALOG TCP/IP 1.5F STARTUP FOR ZVSE51DS
// EXEC LIBR,PARM='MSHP'
ACCESS SUBLIB=CSILIB.ZVSE51DS
CATALOG TCP15F00.L REP=YES
*
* * EXECUTING TCP/IP COMMANDS FROM TCP15F00.L
SET IPADDR = 192.168.0.188
EXEC TCPCOMMN
EXEC TCPOSAX1
EXEC TCPOSAX2
EXEC TCPROUTE
EXEC TCPFILES
EXEC TCPUSERS
EXEC TCPFTPD
EXEC TCPNAMES
EXEC TCPELSTF
```



```

EXEC TCPDNS
EXEC TCPDIAGN
* EXEC TCPMESSG
* EXEC TCPTLND
* EXEC TCPLPD
* EXEC TCPGPSD
/+
/*
/&

```

Note: The lib.su1ib that is used in Example 2-2 on page 36 is first in the LIBDEF SOURCE search chain of the // EXEC IPNET job that is used to start TCP/IP. This is important because we want to ensure that we pick up the correct initial configuration member.

Rather than placing all the commands in a single member, it can be useful to break the configuration commands into separate librarian members, and then run them in the base TCP15F00.L, as shown in Example 2-2. The commands perform the following tasks:

► **SET IPADDR = 192.168.0.188**

This sets the IP address of this stack and must be unique in the network.

► **EXEC TCPCOMMN**

TCPCOMMN.L contains common settings that are used by all stacks in this VSE environment, as shown in Example 2-3.

Example 2-3 TCP/IP startup configuration member

```

// JOB TCPCOMMN CATALOG TCP/IP STARTUP CONFIGURATION MEMBER
// EXEC LIBR,PARM='MSHP'
ACCESS SUBLIB=CSILIB.CONFIG
CATALOG TCPCOMMN.L REP=YES
*
* * EXECUTING TCP/IP COMMANDS FROM TCPCOMMN.L
SET MASK = 255.255.255.0
SECURITY ON,BATCH=ON,AUTO=ON
STEALTH ON
ASECURITY ICMP=NO
CHECKSUM HARDWARE
VERIFY_MEMORY ON
DOWNCHECK OFF
DYNAMIC_ROUTE OFF
DEFINE TRANSLATION,TYPE=SINGLE,MEMBER=IPXLATE,DEFAULT=OS_02
/+
/*

```

For more information about what these commands do, see the *CSI TCP/IP Commands Reference* manual.

2.7 FTP hints

Most companies want to set up and run an FTP server on the VSE system. The FTPDAEMN is the program that services FTP clients that are connecting to the FTP server on VSE. The client automation daemon that is created with a `DEFINE EVENT` command with `ACTION=FTP` connects to an FTP server, and other applications might also automatically establish FTP connections. You want to be sure to have enough FTP server sessions available for all clients to be serviced efficiently.

2.7.1 Internal FTP server suggestions

An internal FTP server is defined with a command similar to what is shown in the following example:

```
DEFINE FTPD,ID=FDAA0021,PORT=21,COUNT=3,ZEROERR=NO, -  
        WELCOME=TCPWELCM,IDLETIME=36000,UNIX=BIN,SENDFAST=YES, -  
        DYNFILE=NO,ALLOWABORT=YES,EXTTYPES=YES,TIMEOUT=9000, -  
        EXTRADATA=ACCEPT
```

With the `COUNT=3` settings, a maximum of three clients can be connected into the VSE FTP server simultaneously. However, it is a good idea to have more than one `DEFINE FTPD` because there is a small window of time (about 1/30th of a second) when a new client is connecting that the single FTP server is not in a listen state. This window can cause a client connection to be rejected. Most clients automatically retry the connection; to avoid this, you can issue a second `DEFINE FTPD` with the same keyword values (especially the same port number) as the first `DEFINE FTPD`. The ID= must be different on the second `DEFINE FTPD`, as shown in the following example:

```
DEFINE FTPD,ID=FDBB0021,PORT=21,COUNT=3,ZEROERR=NO, -  
        WELCOME=TCPWELCM,IDLETIME=36000,UNIX=BIN,SENDFAST=YES, -  
        DYNFILE=NO,ALLOWABORT=YES,EXTTYPES=YES,TIMEOUT=9000, -  
        EXTRADATA=ACCEPT
```

The other parameters of the `DEFINE FTPD` should also be reviewed and set based on your installation requirements.

2.7.2 Using external FTPBATCH servers

Another alternative is to set up and use external FTPBATCH servers in place of the internally defined FTP servers or in addition to them. This configuration provides the following advantages:

- ▶ Uses multiple processors. The z/VSE turbo dispatcher does not run multiple subtasks from a single partition on multiple processors at the same time. However, separate partitions can run on different processors simultaneously.
- ▶ The internal FTP servers share a single file I/O subtask. External FTPBATCH servers use a dedicated file I/O server for each FTP session.
- ▶ Virtual storage usage is segregated for the external FTPBATCH server.
- ▶ Recoverability can be expedient by releasing a new FTPBATCH server.

- Shows the JCL to run an external FTPBATCH server, as shown in Example 2-4.

Example 2-4 JCL for an FTPBATCH server

```
// JOB FTPB2121
// OPTION LOG
// OPTION SYSPARM='00'
// EXEC FTPBATCH,SIZE=FTPBATCH,PARM='FTPDPORT=2121,UNIX=BIN,MAXACT=12'
SET IDLETIME 36000 / 300= 120 SECONDS TO TERMINATE IDLE SESSIONS
/*
```

Any files that are defined in the TCP/IP file system must have assigns and DLBLs so the sample job that is shown in Example 2-4 can access these files.

The external FTP server that is shown in Example 2-4 is set to listen on port 2121. Another trick that can be used is to have your AUTOFTP events use the external FTP server by using a script similar to what is shown in Example 2-5.

Example 2-5 Using an auto-FTP script

```
CATALOG AFTPTEST.L    REP=YES
LOPEN 127.0.0.1 2121
LUSER xxxxxxxx
LPASS xxxxxxxx
OPEN  ip-addr
USER  yyyyyyyy
PASS  yyyyyyyy
LCD   /POWER/LST/F
CD    /ZVSE52DS
SETVAR &LFN = &PWRNAME + "." + &PWRNUMB + "." + &PWRSUFF
SETVAR &PCNAME = &LFN + ".LST"
LSITE CC OFF
LSITE STRIP ON
PUT &LFN &PCNAME
CLOSE
LCLOSE
/+
```

This causes the internal automation daemon (CLIENTD) to use the external FTPBATCH server that is listening on port 2121. Thus, the automation daemon is the FTP client that is sending FTP commands that are contained in the script file to the external FTPBATCH server.

2.8 Partition priorities

You should use weighted balanced partition priorities, with TCP/IP having a higher weight than the socket application partitions. The following example shows priorities with TCP/IP running in the F8 partition:

```
PRTY BG=FB=FA=F7=F6=F5=F4=F3=F2=C=P=R=W=S=Y=Z=T=F8,F9,F1
PRTY SHARE,BG=100,F2=100,F3=100,F4=100,F8=400
```

The workload of applications and TCP/IP should be viewed as a single unit of work. This is from stress testing that is performed on our system by using 72 simultaneous FTPBATCH jobs.

2.9 Security

Automatic Security is available and can be used to eliminate maintaining home grown security programs. It provides the following features:

- ▶ ICMP=N0 to turn off ping sweep hackers
- ▶ BLOCKIP=YES can be used to block after multiple violations; for example:
 - IPN497I ANONYMOUS denied access to SXTYPASS Password-check from 192.187.96.94
 - IPI107I All traffic with 192.187.96.94 is prevented
- ▶ Works with DEFINE USER for resources

For more information about setting up and using automatic security, see this website:

http://www.tcpip4vse.com/doc/z-VSE/TCP-IP4VSE/Release_1.5F/TCP-IP4VSE_1.5F_Command_Reference_rev_1_20100319.pdf

Consider the following security issues as you proceed:

- ▶ The **DEFINE USER** command can be used to restrict user IDs to specific directories in the file system, as shown in the following example:
DEFINE USER, ID=GSEUSER1, ROOT='POWER.LST.A'
Limits GSEUSER1 to just Power LST queue class A.
- ▶ The **DIAG CONNREJ** command can be used to display rejected connection attempts.
- ▶ **STEALTH ON** causes no responses to rejected hack attempts.
- ▶ **DIAG GETVIS** monitors TCP/IP and system GETVIS usage.

2.10 Remote running with REXX

Here is a tip about how to start a remote run client from REXX.

The original problem is that when the address link is used to call the TCP/IP CLIENT phase from REXX, this works only once. However, it might fail because of missing internal usage during other calls. Example 2-6 shows an extract from a REXX procedure that uses address link.

Example 2-6 Using address link to start the TCP/IP CLIENT phase

```
t = 0
t=t+1;SYSIPT.t = "SET DEBUG=ON"
t=t+1;SYSIPT.t = "SET HOST="hostname
t=t+1;SYSIPT.t = "SET USERID="userid
t=t+1;SYSIPT.t = "SET PASSWORD="passwd
t=t+1;SYSIPT.t = "COMMAND ls -l"
t=t+1;SYSIPT.t = "COMMAND ./bin/myscript.sh"
t=t+1;SYSIPT.t = "QUIT"
SYSIPT.0 = t
address link "CLIENT APPL=REXEC"
```

The solution for multiple sequential calls is writing a second REXX procedure to call (instead of loading) CLIENT under REXX. Change the code extract that is shown in Example 2-6 to the lines that are shown in Example 2-7.

Example 2-7 Using a second REXX procedure to call the TCP/IP REXEC client

```
cmd1="ls -l"
cmd2="./bin/myscript.sh"
rc=REXXEXEC(userid,passwd,hostname,cmd1,cmd2)
```

This way, all parameters are passed to a second REXX procedure REXXEXEC, which connects to the internal REXEC client and processes all passed commands.

Example 2-8 shows the relevant parts of the REXXEXEC procedure.

Example 2-8 Sample REXX procedure to start the TCP/IP REXEC client

```
x=ASSGN('STDOUT','SYSLOG')
narg=arg()
if narg < 4 then do
    say 'Invalid number of arguments passed to REXXEXEC.'
    say 'You need at least USERID, PWD, HOST, CMD'
    exit 8
end
```

```
/* Save the passed data */
```

```
ruser=arg(1)
rpass=arg(2)
rhost=arg(3)
cmdno=3
loopcnt=narg-3
```

```
/* Connect to the internal REXEC client */
```

```
call OPEN
x=ASSGN('STDOUT','SYSLSST')
sendbuf='REXEC'
Call send
call receive
sendbuf='SET DEBUG=ON'
Call send
call receive
sendbuf='SET HOST='rhost
Call send
call receive
sendbuf='SET USER='ruser
Call send
call receive
sendbuf='SET PASS='rpass
Call send
call receive
```

```
/* Process all of the commands passed */
```

```
do loopcnt
```

```

    cmndno=cmndno+1
    rcmd=arg(cmndno)
    sendbuf='c 'rcmd
    Call send
    call receive
    end

sendbuf='QUIT'
Call send
call receive
call close
exit 0

```

The following examples show the subroutines open, send, receive, and close. Example 2-9 shows the open routine.

Example 2-9 The open routine

```

open:                                /* Start of routine          */
x = SOCKET('CLIENT','OPEN',,,,SYSID)
if x \= 0 then do
    say 'Unable to connect to internal CLIENT.'
    say 'Check STACK availability or SYSID settings'
    say errmsg
    exit 4
end
return

```

Example 2-10 shows the send routine.

Example 2-10 The send routine

```

send:                                /* Start of routine          */
say sendbuf
x=SOCKET(handle,'SEND',sendbuf)
    if x \= 0 then do                /* If it failed...          */
        say errmsg                  /* Tell us why              */
        call close                  /* Close the connection     */
        exit 8                      /* And terminate the program */
    end                            /* If it worked...          */
return                             /* Return to caller         */

```

Example 2-11 shows the Receive routine.

Example 2-11 Receive routine

```
receive:                                /* Start of routine                */
do forever                              /* Do until we find a match          */
X = SOCKET(HANDLE,'RECEIVE')            /* Receive a response                */
if x > 4 then do                         /* If it failed...                  */
    say errmsg                          /* Tell us what went wrong          */
    call close                          /* Close the connection             */
    exit x                              /* And terminate the program        */
end                                     /* Otherwise...                     */
say buffer                              /* What came back ?                 */
loc=POS('Ready:',buffer)                /* Look for another keyword          */
if loc > 0 then return                  /* And if we found it, it's okay    */
loc=POS('Error:',buffer)                /* Look for another keyword          */
if loc > 0 then do                      /* Oops                             */
    x=ASSGN('STDOUT','SYSLOG')          /* Output to SYSLOG only            */
    say 'REXEC Client command failure. Check SYSLSST'
    exit 4
end
end
```

Example 2-12 shows the close routine.

Example 2-12 The close routine

```
close:                                /* Start of routine                */
x = SOCKET(handle,'CLOSE')              /* Close the SOCKET                 */
return                                  /* Return to caller                 */
```

2.11 Version checking

A new batch utility CIALSHPH can be used to verify that downloaded phases have a matching SHA-1 hash. This can ensure that the bits and bytes of any phase or librarian member are at the correct maintenance level. For more information, see the TCP/IP documentation that is listed in “Online resources” on page 228.

2.12 Datagram analysis

A new utility SVSESRVR captures datagrams from a data space. These datagrams can then be investigated by the SeeTCPIP product on a system that is running Windows. Run the **SVSESRVR** command with a job similar to the JCL shown in Example 2-13.

Example 2-13 JCL to run the SVSESRVR utility

```
// JOB SVSESRVR
// OPTION SYSPARM='00'
// EXEC SVSESRVR,SIZE=SVSESRVR
TRACEDSP 256K default size of dataspace
TRACETIM 60 default sampling time
/*
```

After the datagrams are captured, use *SeeTCPIP* on a system to pull the dataspace of datagrams that were captured by *SVSESRVR*. This creates an *nnn.pcap* file on the system and starts the Wireshark datagram analysis tool against the *.pcap* file.

2.13 Known problems

This section describes the issues that we encountered during our tests.

2.13.1 Routing in a subnet

In this section, we describe a configuration tip that you can use to manage routing problems in a subnet.

Problem

If you notice routing problems in your subnet (perhaps after a switch or router update), check your route configuration. For example, if you have a class C network and your z/VSE has the IP 192.168.0.3, you have a configuration as shown in the following example:

```
SET IPADDR    = 192.168.0.3
SET MASK      = 255.255.255.0
```

Possible solution

If your router has IP 192.168.0.1, you add the following route statements:

```
DEFINE ROUTE, ID=SUBNETR, LINKID=OSA, IPADDR=192.168.0.0
DEFINE ROUTE, ID=ALL, LINKID=OSA, IPADDR=0.0.0.0, GATEWAY=192.168.0.1
```

The first route (ID=SUBNETR) routes all packets for the subnet directly (without the use of the gateway). For all other packets, it uses the gateway, as set in the second route (ID=ALL).

2.13.2 Using SSL ciphers

Some supported cipher suites are no longer recommended and some do not work.

Problem

TCP/IP for VSE supports the SSL ciphers 01, 02, 08, 09, 0A, 2F, and 35. Recent tests showed that 01 and 08 do not work anymore, at least when the client is a Java application, such as VSE Navigator.

Solution

Use the AES cipher suites 2F and 35 whenever possible. For security reasons, DES-related cipher suites should no longer be used.

2.13.3 Secure SSL port

Do not rely on a default port value when 'SSL=CLIENT' in FTPBATCH is used.

Problem

The default port number for FTPBATCH always is 21, whereas the default port for secure FTP often is 990.

Solution

When FTPBATCH is used with 'SSL=CLIENT', the port number must always be explicitly specified on the OPEN command, as shown in the following example:

```
OPEN ipaddr port
```

2.13.4 SSL client does not verify the server certificate

You should be aware that when the GSK API is used, a connection can be established even when a non-existing key ring is referenced.

Symptom

An SSL client does not verify the server certificate. The connection is established even when a non-existing key ring is referenced; for example, for FTPBATCH, as shown in the following example:

```
// EXEC FTPBATCH,SIZE=FTPBATCH,PARM='SSL=CLIENT'  
SET SSL PRIVATE CRYPTO.KEYRING.nnnn
```

In this example, there is no nnnn.PRVK, nnnn.CERT, and nnnn.ROOT members or they can contain keys that are not related to the server certificate.

Reason

This is the intended behavior of the CSI implementation of the GSK API. An SSL client does not read the key ring. Therefore, function gsk_secure_soc_init by default uses handshake type GSK_AS_CLIENT_NO_AUTH.

2.13.5 TLS issue with IBM Personal Communications 6.0.7

Here is a problem that we experienced when we moved from PCOM 6.0.5 to 6.0.7.

Problem

Personal Communications 6.0.7 displays message "The Local Security Authority Cannot Be Contacted".

TCP/IP for VSE/ESA issues the following messages:

```
TEL927I TELNETDS diagnostic: SSLRSRBD RC=00000004 RS=000001A8  
TEL927D TELNETDS diagnostic: SSLHSINI RC=FFFFF3E4 RS=00000186  
TEL917I Daemon Resetting Telnet Termname: nnnn IPaddr: n.n.n.n
```

This setup worked with PCOM 6.0.5.

Reason

PCOM 6.0.7 tries to use TLS 1.1, which is not supported by TCP/IP for VSE/ESA. A fix should be provided by PCOM to allow the use of TLS 1.0.



IPv6/VSE

This chapter provides an overview of Internet Protocol version 6 (IPv6) and its support in z/VSE through the IPv6/VSE product. IPv6/VSE is being developed by Barnard Software, Inc. (BSI).

While the product is named IPv6/VSE, it supports IPv4 and IPv6 communications. IPv6/VSE provides a full-function IPv4 stack and applications and a full-function IPv6 stack and applications. Both TCP/IP stacks (IPv4 and IPv6) can be run together (dual stack mode), individually, or as stand-alone stacks.

In this chapter, we describe how to set up IPv6/VSE for IPv4, IPv6, and dual-stack environments. We then show how to configure and use TCP/IP applications, such as TN3270, FTP, and TN3270E printing. Finally, we provide detailed information about how to set up and use SSL in various scenarios.

This chapter includes the following topics:

- ▶ Overview
- ▶ Obtaining and activating a license key
- ▶ Stack setup
- ▶ Setting up FTP
- ▶ Setting up TN3270
- ▶ Setting up TN3270E printing
- ▶ Setting up SSL
- ▶ Setting up CICS Web Support
- ▶ Known problems

3.1 Overview

IPv6/VSE is part of z/VSE since z/VSE 4.2.2. It is designed to provide an IPv6 solution for z/VSE to perform the following tasks:

- ▶ Allow z/VSE users to participate in an IPv6 network.
- ▶ Bring the benefits of IPv6 functionality to z/VSE users.
- ▶ Help z/VSE users to meet the requirements of the commercial community and governmental agencies. This fulfills the statement of direction in IBM Software Announcement 209-319, dated October 20, 2009.

For more information about BSI, see this website:

<http://www.bsiopti.com>

IPv6/VSE not only provides an IPv6 TCP/IP stack. It also includes a full-function IPv4 TCP/IP stack, which does not require the IPv6 TCP/IP stack to be active.

The IPv6 stack includes the following components and features:

- ▶ Provides support for the IPv6 protocol
- ▶ IPv6 application programming interfaces (APIs)
- ▶ IPv6-enabled applications
- ▶ Supports IPv6 only, not IPv4

The IPv4 stack includes the following components and features:

- ▶ Provides support for the IPv4 protocol
- ▶ IPv4 application programming interfaces (APIs)
- ▶ IPv4-enabled applications
- ▶ Supports IPv4 only, not IPv6

To allow applications to use IPv4 and IPv6 at the same time, the following criteria must be met:

- ▶ Run both stacks (in separate partitions)
- ▶ Run the **COUPLE** command to join the two stacks

The two coupled stacks act as one (dual) stack, which supports IPv6 and IPv4.

3.2 Obtaining and activating a license key

A license key for IPv6/VSE can be obtained by using one of the following methods:

- ▶ When IPv6/VSE is licensed from IBM, you get the key from the IBM Key Center in Copenhagen, Denmark.
- ▶ When IPv6/VSE is licensed directly from Barnard Software, Inc., you get the key from BSI.

A license key is generated from the CPU ID; therefore, it is different for each processor and for each LPAR on a system.

The key data consists of three lines (COMPANY, CPUID, and LICENSE) that must be copied into the BSTTPARM.A member in the IPv6/VSE installation sublibrary (default is PRD2.TCPIPB).

Example 3-1 shows the contents of a sample BSTTPARM member.

Example 3-1 IPv6/VSE license key in BSTTPARM.A

```
* * * * *
*
*      TCP/IP-TOOLS COMMON PARAMETERS
*
* * * * *
*
COMPANY IBM DEUTSCHLAND R&D
CPUID 3B0B82 MODEL 2097
LICENSE IPV6/VSE ABCDEFGHIL6Z 2029365 3736856
*
IPV6/VSE ENABLE
```

Copy the BSTTPARM.A member to PRD2.CONFIG and make the changes there.

3.3 Stack setup

In this section, we describe how to set up IPv6/VSE for IPv4, IPv6, and a mixture of both.

The IPv6/VSE stack requires a minimum 20 MB partition. This amount of storage allows for stack startup and storage for a few users. It is enough for basic testing, but 20 MB is too small for production usage.

The basic formula is 20 MB + 108 KB per TCP connection. For example, a system with 100 TCP (TN3270E) connections plus 10 FTP (client and server) connections requires 20 MB + 108 x 110 (11880 KB) = ~32 MB.

The maximum number of TCP connections that is supported is 4095. To support the maximum number of connections, the stack must to run in a 460 MB partition.

Note: Many z/VSE users still have small batch partition sizes. IPv6/VSE applications all require an 8 MB minimum partition size, unless there is a specific formula in the manual.

The following sections show the basic steps to set up an IPv4 stack, an IPv6 stack, and a mixed environment with a gateway between IPv4 and IPv6. We then describe how to couple IPv4 and IPv6 to get a dual stack environment.

3.3.1 IPv4 stack setup

Having both sides in an IPv4 network, the setup is simple, as shown in Figure 3-1.

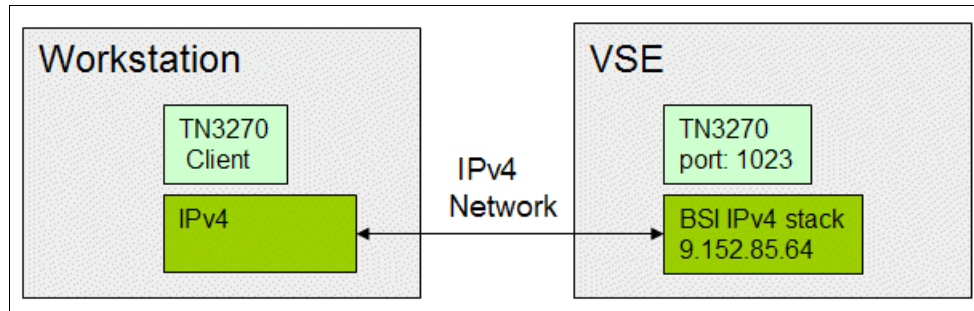


Figure 3-1 IPv4 stack setup

In the job in the following section, the BSI IPv4 stack has SYSID=44.

IPv4 stack startup job

The JCL that is shown in Example 3-2 was used in our test environment to start the IPv4 stack.

Example 3-2 JCL for starting an IPv4 stack

```
* $$ JOB JNM=BSI44,CLASS=S,DISP=L
* $$ LST CLASS=S,DISP=D
// JOB BSI44
// OPTION LOG,PARTDUMP,NOSYSDDUMP,SADUMP=1
// LIBDEF *,SEARCH=(PRD2.CONFIG)
// ASSGN SYS000,SYSLST
// SETPFIX LIMIT=(2M,1M)
// EXEC BSTTINET,SIZE=BSTTINET,OS390,TASKS=ANY
ID 44
INTERVAL 120
*
DEVICE OSAXD10 OSAX D10 IPV4TST D12
*
LINK OSAXD10 0 9.152.85.64 255.255.252.0 1492
*
ROUTE OSAXD10 9.152.85.0 255.255.252.0 0.0.0.0 0
ROUTE OSAXD10 0.0.0.0 0.0.0.0 9.152.84.1 1
*
DNS 9.152.120.241
DNS 9.152.64.172
*
HOST LOCALHOST 127.0.0.1
HOST VSEC01 9.152.85.64
HOST VSEC03 9.152.85.67
HOST VSEC09 9.152.86.16
HOST VSEP15 9.152.85.126
HOST VSEP16 9.152.85.165
*
ATTACH TCP/IP
/*
```

3.3.2 IPv6 stack setup

The network setup of a plain IPv6 network looks similar to plain IPv4, as shown in Figure 3-2.

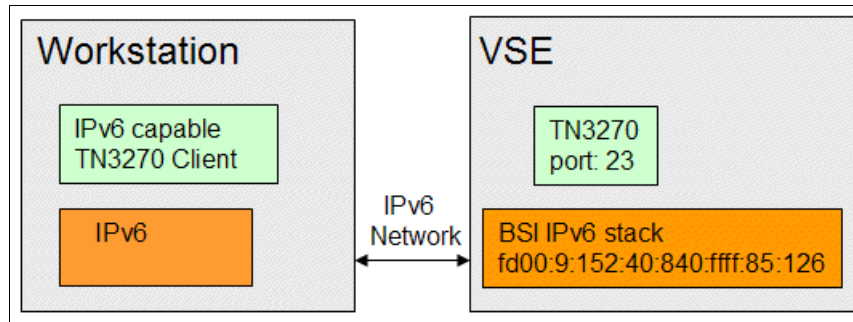


Figure 3-2 IPv6 stack setup

Many Telnet clients (for example, IBM Personal Communications V5.9) are not IPv6-capable. Therefore, we used Open Text Exceed as the TN3270 client with IPv6. The full setup for IPv6 requires some configuration on the Windows side before we can establish a sample TN3270 session.

Setting up IPv6 in Windows XP

In this section, we provide a brief overview of configuring IPv6 for Windows XP. IPv6 is not preinstalled on Windows XP; therefore, you must install and configure the IPv6 protocol manually. IPv6 cannot be fully installed and configured by using a GUI. Instead, the **netsh** command must be used here. On Windows Vista and Windows 7, all steps are supported in the GUI.

Installing IPv6

Complete the following steps to install the Windows IPv6 protocol:

1. Open the Windows Control Panel.
2. Open the Windows Network Connections.
3. Open the properties window of your LAN connection and click **Install**.

4. On the Network Component Type window, select **Protocol** and click **Add**, as shown in Figure 3-3.

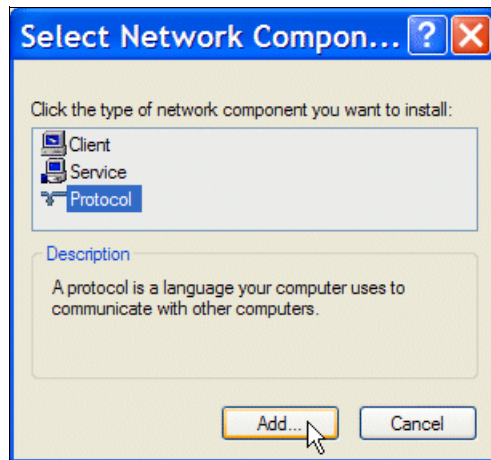


Figure 3-3 Network Component Type

5. In the Select Network Protocol window, select the IPv6 protocol and click **OK**, as shown in Figure 3-4.

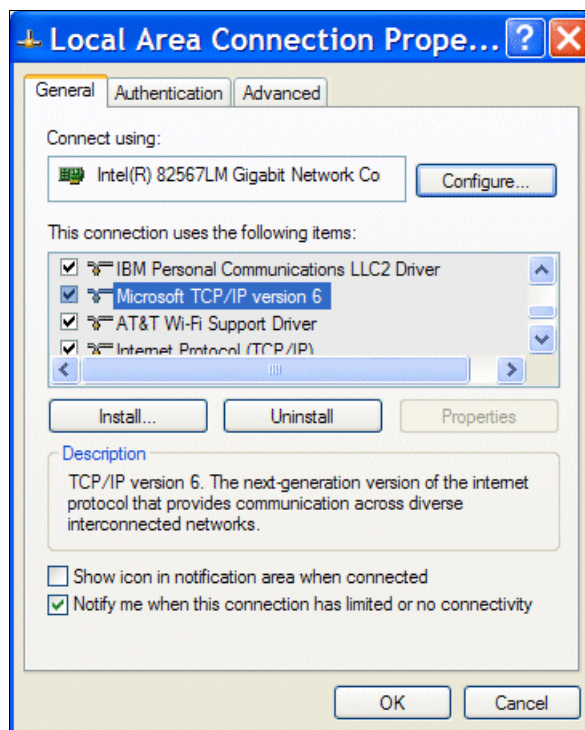


Figure 3-4 Local Area Connection Properties window in MS Windows

The Microsoft TCP/IP version 6 item should be visible in the LAN Connection Properties window.

Note: Figure 3-4 was taken from a Windows XP SP3, so the **Properties** button is not available. Instead, the **netsh** command was used for configuring IPv6.

Configuring IPv6

First, you must determine the index of your IPv6 interface, as shown in Example 3-3.

Example 3-3 Using netsh to display the network interfaces

```
D:\>netsh interface ipv6 show interface
Querying active state...
```

Idx	Met	MTU	State	Name
---	----	-----	-----	-----
6	2	1280	Disconnected	Teredo Tunneling Pseudo-Interface
5	0	1500	Connected	Local Area Connection
4	0	1500	Disconnected	Wireless Network Connection
3	1	1280	Connected	6to4 Tunneling Pseudo-Interface
2	1	1280	Connected	Automatic Tunneling Pseudo-Interface
1	0	1500	Connected	Loopback Pseudo-Interface

In this example, it is index 5 (Local Area Connection).

Then, use the following **netsh** commands to configure a static IPv6 address. Make sure to use the correct interface number from the output that is shown in Example 3-3 on page 53:

```
netsh interface IPv6 add address interface=5 address=fd00:9:152:40:840::1001
netsh interface IPv6 add prefixpolicy prefix=fd00:9:152:40:840::/80 5 6
netsh interface IPv6 add route fd00:9:152::/48 interface=5 nexthop=fd00:9:152:40:840::1
```

You can now test the connection to VSE.

Testing the connection

On Windows, you can use `ping -6` or `ping6` to test if a computer is reachable in an IPv6 network. Example 3-4 shows the output from our test setup.

Example 3-4 Testing the IPv6 connection

```
D:\>ping6 fd00:9:152:40:840:ffff:85:126
Pinging fd00:9:152:40:840:ffff:85:126
from fd00:9:152:40:840::1001 with 32 bytes of data:

Reply from fd00:9:152:40:840:ffff:85:126: bytes=32 time=4ms
Reply from fd00:9:152:40:840:ffff:85:126: bytes=32 time<1ms
Reply from fd00:9:152:40:840:ffff:85:126: bytes=32 time<1ms
Reply from fd00:9:152:40:840:ffff:85:126: bytes=32 time<1ms

Ping statistics for fd00:9:152:40:840:ffff:85:126:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 4ms, Average = 1ms
```

You can use the **netsh** command to display more information; for example, showing neighbors (VSE is shown in **bold**), as shown in Example 3-5.

Example 3-5 Using netsh to display IPv6 interface information

```
D:\>netsh interface ipv6 show neighbors
```

Interface 5: Local Area Connection

Internet Address	Physical Address	Type
-----	-----	-----
fd00:9:152:40:840::1	00-0a-8b-bf-71-80	Stale (router)
fd00:9:152:40:840::1001	00-24-7e-e0-72-4a	Permanent
fe80::224:7eff:fee0:724a	00-24-7e-e0-72-4a	Permanent
fe80::20a:8bff:febf:7180	00-0a-8b-bf-71-80	Stale
fd00:9:152:40:840:ffff:85:126	00-11-25-bd-98-c2	Stale
...		

If you cannot reach VSE, see 3.9.1, “VSE cannot be reached” on page 119.

IPv6 stack startup job

The JCL that is shown in Example 3-6 is used in our test environment to start the IPv6 stack. The BSI IPv6 stack has SYSID=66.

Example 3-6 IPv6 stack startup job

```

* $$ JOB JNM=BSI66,CLASS=S,DISP=L
* $$ LST CLASS=S,DISP=D
// JOB BSI66
// OPTION LOG,PARTDUMP,NOSYSDUMP,SADUMP=1
/. LIBDEF *,SEARCH=(PRD2.CONFIG,PRD2.BSI)
// LIBDEF *,SEARCH=(PRD2.CONFIG)
// ASSGN SYS000,SYSLST
/. SETPFIX LIMIT=(3M,32M)
// SETPFIX LIMIT=(2M,1M)
// EXEC BSTT6NET,SIZE=BSTT6NET,OS390,TASKS=ANY
ID 66
INTERVAL 120
*
DEVICE OSAXD00 OSAX D20 IPV6TST D22
*
LINK          OSAXD00 0 FD00:9:152:40:840:FFFF:85:64 /80 1492
*
ROUTE         OSAXD00 FD00:9:152:40:840::1 /80 ::0 0
ROUTE         OSAXD00 ::0 ::0 FD00:9:152:40:840::1 1
DNS           FD00:9:152:41:2422:FFFF:4:230 PRI
*
HOST          LOCALHOST  ::1
HOST          IP6GW       fd00:9:152:40:840::1
HOST          IP6DNS1     fd00:9:152:41:2422:ffff:4:230
HOST          IP6DNS2     fd00:9:152:48:2522:ffff:5:231
HOST          VSEC01      fd00:9:152:40:840:ffff:85:64
HOST          VSEC03      fd00:9:152:40:840:ffff:85:67
HOST          LINXLFP     fd00:9:152:40:840:ffff:86:237
HOST          LINR01      fd00:9:152:40:840:ffff:84:13
HOST          LINR02      fd00:9:152:40:840:ffff:84:171
*
ATTACH TCP/IP
/*
// EXEC LISTLOG
/&
* $$ EOJ

```

Starting a TN3270 server

You can use the same JCL for starting the TN3270 server with IPv6 as is used with IPv4 with one change: specify the SYSID of the IPv6 stack, as shown in the following example:

```
// EXEC BSTTVNET,SIZE=BSTTVNET,DSPACE=3M,TASKS=ANY,OS390
ID 66
OPEN VSEC01 1023
*
```

The daemon startup is shown on the VSE console, as shown in Example 3-7.

Example 3-7 BSTTVNET startup on VSE console

```
S1 0045 // JOB BSITELR
      DATE 11/09/2010, CLOCK 13/23/16
S1 0045 BSTT000I INITIATED  BSTTWAIT Ver 2.46 04/01/09 18.08    EP=00520078
S1 0045 BSTT003I COPYRIGHT (C) 1998-2010 BARNARD SOFTWARE, INC.
S1 0045 BSTT001I TERMINATED BSTTWAIT
S1 0045 BSTT000I INITIATED  BSTTVNET Build248 02/15/10 16.24    EP=00520078
S1 0045 BSTT003I COPYRIGHT (C) 1998-2010 BARNARD SOFTWARE, INC.
S1 0045 BSTT004I CB=TTLA A=00569000 L=000013FC
S1 0045 BSTT019I VSE 8.20 MODE 31-BIT
S1 0045 BSTT004I CB=COMR A=0033D4F0 L=00000108
S1 0110 BSTT000I INITIATED  BSTTXVNC Build249 04/29/10 08.27    EP=005FB000
S1 0110 BSTT671I BSTTVNET VTAM FEATURE IS AVAILABLE
S1 0110 BSTT701I IPv6/VSE    BUILD 249
S1 0110 BSTT695I CONNECTING TO PORT 23 IP fd00:9:152:40:840:ffff:85:126
S1 0111 BSTT000I INITIATED  BSTTVSRV Build248 03/22/10 20.41    EP=00668000
S1 0110 BSTT042I ATTACH OF BSTTVSRV COMPLETED
```

Connecting with Open Text Exceed

As shown in Figure 3-5, enter the IPv6 address and TN3270 port of the VSE system in the Edit Host Info window to connect directly to VSE by using IPv6.

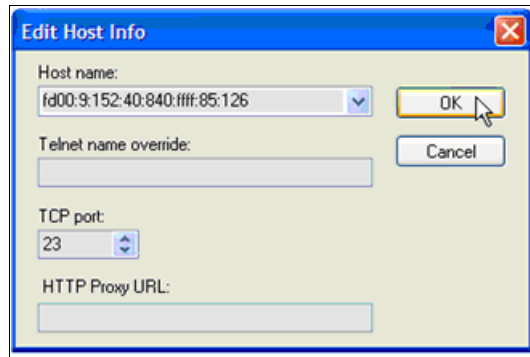


Figure 3-5 Edit Host Info window in Open Text Exceed

3.3.3 Mixed IPv4 and IPv6 network setup

Having the TN3270 client and server in different network types requires a gateway that can handle IPv4 and IPv6 traffic, as shown in Figure 3-6.

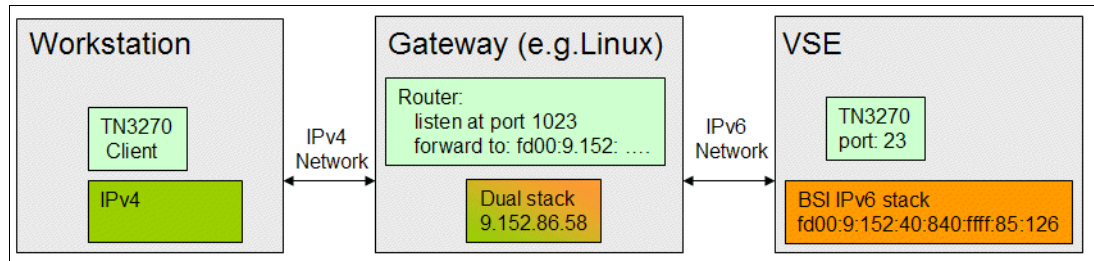


Figure 3-6 Mixed IPv4 and IPv6 network setup

In our test setup, a C program was used as the gateway. Here, the middle-tier platform is Linux, but it might be any platform with a dual stack, including Windows. For an example of a Transmission Control Protocol (TCP) forwarding program, see this website:

http://www.kernel.org/doc/man-pages/online/pages/man2/select_tut.2.html

Note: z/VSE 4.3 and later can be used as the gateway when the EZA socket interface is used. With EZA, you can access IPv4 and IPv6 sockets in the same program. However, a C interface is not yet available.

Connecting with a TN3270 client

As with the IPv4 setup, we can use IBM Personal Communications as the TN3270 client. For Host Name or IP Address, enter the IP address of the middle-tier platform: 9.152.86.58 with port number 1023, as shown in Figure 3-7. The gateway program forwards traffic to the VSE system in the IPv6 network, where the TN3270 daemon listens on port 23.

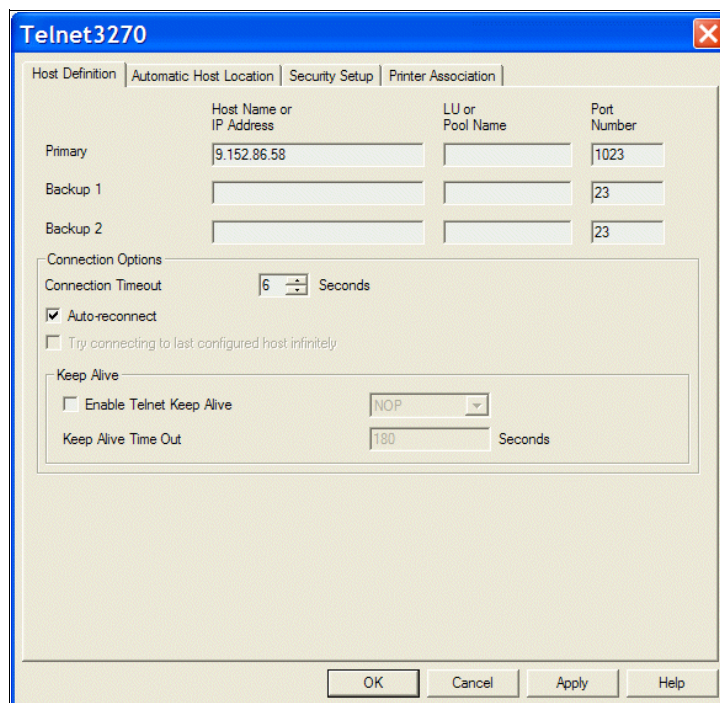


Figure 3-7 Telnet3270 window in IBM Personal Communications

After a connection is made to VSE, the IPv6/VSE VTAM selection window that is shown in Figure 3-8 opens.

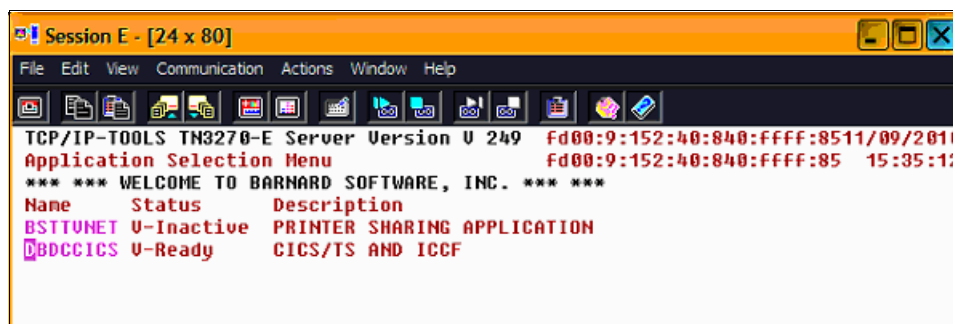


Figure 3-8 VTAM selection window for IPv6/VSE

Select **DBDCCICS** and press Enter to open the VSE sign-on window. You can use the same setup with any other TN3270 client, such as Open Text Host Explorer.

3.3.4 Setting up a dual-stacked system

In this section, we describe the setup of a dual-stacked system with an IPv4 stack coupled to an IPv6 stack. This setup has the advantage that your applications work transparently on IPv4 and IPv6 networks, as shown in Figure 3-9.

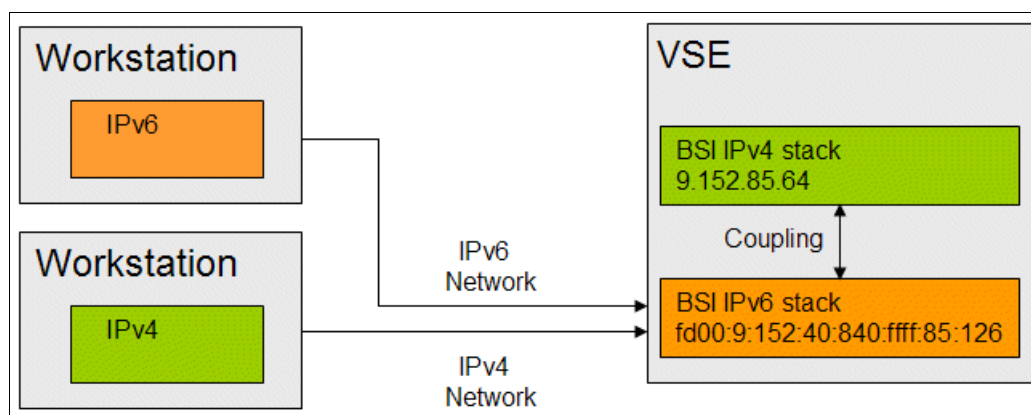


Figure 3-9 Dual-stacked system setup

All types of requests are first routed to the IPv6 stack, which recognizes IPv4 packets and sends them to the IPv4 stack for processing.

For more information about the COUPLED command, see *IPv6/VSE User's Guide*, SC34-2618.

For more information about configuring a dual-stacked system, see *IPv6/VSE Installation Guide*, SC34-2616.

Example 3-8 shows an example of how to start a dual-stacked system.

Example 3-8 Sample job for a dual-stack system

```
* $$ JOB JNM=BSI02V6,CLASS=S,DISP=D
* $$ LST CLASS=S,DISP=D
// JOB      BSI02V6
```

```
// OPTION LOG,PARTDUMP,NOSYS_DUMP,SADUMP=1
// LIBDEF PHASE,SEARCH=(PRD2.BSI)
// LIBDEF SOURCE,SEARCH=(PRD2.CONFIG,PRD2.BSI)
// SETPFIX LIMIT=(2M,1M)
// ASSGN SYS000,SYSLST
// EXEC BSTT6NET,SIZE=BSTT6NET,OS390,TASKS=ANY
ID      02
COUPLE  03
*
INTERVAL 120
DEVICE  OSA OSAX D60 TRLHF9 D62 LAYER2 020000000006
*
LINK    OSA 0 FD00:9:152:40:840:FFFF:85:126 /80 8192
*
ROUTE  OSA  FD00:9:152:40:840::1 /80 ::0 0
ROUTE  OSA  ::0 ::0 FD00:9:152:40:840::1 1
*
ATTACH TCP/IP
/*
/&
* $$ EOJ
```

3.3.5 UDPv4 in a coupled stack environment

While the IPv6/VSE stack supports “dual-listens” for TCP sockets in a coupled stack environment, it does not support “dual-binds” for UDP sockets. This means that if you use an application that uses UDP for communication (such as, the SNMP Monitoring Agent), the application is reachable only with IPv6. The SNMP Monitoring Agent answers requests from IPv6 only.

If you want to use your application for IPv4 and IPv6, you must start two instances of your application and bind one instance to the IPv6 stack and the other to the IPv4 stack. You do this by using SYSPARM OPTION in the startup job of your application. A sample SNMP Monitoring Agent startup job is shown in Example 3-9.

Example 3-9 SNMP Monitoring Agent startup job

```
* $$ JOB JNM=STARTMAS,DISP=L,CLASS=R
// JOB STARTMAS STARTS THE SNMP MONITORING AGENT
* *****
* This Job starts the SNMP MONITORING AGENT.
* Please change the ID and the SYSPARM card if necessary
* *****
// ID USER=VCSRV,PWD=VCSRV
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD1.BASE,PRD2.SCEEBASE)
// OPTION SYSPARM='02'
// EXEC IESMASNM,PARM='DD:PRD2.CONFIG(IESMASCF.Z)'
/*
/&
* $$ EOJ
```

Sample JCL is provided in skeleton SKSTMAS in ICCF library 59.

3.3.6 Connectivity considerations

In this section, we provide some tips for resolving connectivity issues by using a printer as an example. Consider the situation where the printer no longer responds.

The first issue that we must consider is the difference between pinging an IP address and connecting to an IP address. Pinging an IP address means only that there is a host of some kind that is responding at that IP address. It does not mean that there is a printer at that IP address or that there are any applications that are running at that IP address.

When you receive an RC=4 on an OPEN, this indicates that the application did not establish a connection on the specified port. The next issue to consider is how to determine what is happening.

One way to tell if the printer is accepting connections is to use telnet (rather than ping) to attempt to connect to the printer. In a Windows or Linux command prompt, enter the following command:

```
telnet ip-address 9100
```

You might see output that is similar to the following three cases:

- Telnet did not find the host at all, as shown in the following example:

```
jcb@dv9500t:~> telnet 10.1.1.16 9100
Trying 10.1.1.16...
telnet: connect to address 10.1.1.16: No route to host
```

- The host exists, but is not accepting connections on port 9100, as shown in the following example:

```
jcb@dv9500t:~> telnet 10.1.1.2 9100
Trying 10.1.1.2...
telnet: connect to address 10.1.1.2: Connection refused
```

- An application (for example, a printer) is answering the attempt to connect, as shown in the following example:

```
jcb@dv9500t:~> telnet 192.168.1.103 9100
Trying 192.168.1.103...
Connected to 192.168.1.103.
Escape character is '^]'.
^]
telnet> quit
Connection closed.
```

3.4 Setting up FTP

In this section, we describe how to set up FTP with z/VSE acting as server or client.

3.4.1 Security

The following methods are available for controlling FTP security:

- Set up the BSTTSCTY.T security member.

A default security member is provided with the stack. However, the default member provides open access to the FTP server. Any user ID and password are accepted.

- Use the IBM provided security exit BSSTISX.

The BSI FTP server security exit routine BSTTFTS1.PHASE calls the BSSTISX security exit to verify user ID and password. All other security is controlled by using the BSTTSCTY.T member.

To enable the BSTTFTS1.PHASE security exit, complete the following steps:

- Copy the BSTTFTS1.PHASE to a configuration lib.slib as BSTTFTSX.PHASE.
- In a LIBDEF statement, reference this configuration lib.sublib in the BSTTFTPS PHASE,SEARCH chain.
- Add the following BSSTISX command to your BSTTFTPS startup commands:

```
BSSTISX <data>
<data> is [anonym_uid] [, [anonym_pwd] [, [preproc] [, [postproc] [, [mode]]]]]
```

For more information about configuring security for IPv6/VSE, see the *IPv6/VSE IPv4 User's Guide*, SC34-2618, and *IPv6/VSE Migration Guide*, SC34-2624.

3.4.2 VSE as a server

You can start an FTP server by using the BSTTFTPS utility. The FTP server partition must be 24 MB or larger. BSI recommends running BSTTFTPS in the largest possible partition with no 31-bit storage.

The JCL that is shown in Example 3-10 starts an FTP server on VSE that is listening on port 21.

Example 3-10 FTP server JCL

```
* $$ JOB JNM=FTPDIPV6,CLASS=S,DISP=D
// JOB FTPDIPV6 - FTP SERVER FOR BSI STACK
// LIBDEF PHASE,SEARCH=(PRD2.CONFIG,PRD2.TCPIPB)
// LIBDEF SOURCE,SEARCH=(PRD2.TCPIPB)
// EXEC BSTTFTPS,SIZE=BSTTFTPS,OS390
ID 02
*
OPEN 9.152.86.91 21
*
* USE THE IBM PROVIDED BSM SECURITY EXIT
BSSTISX
*
* DEFINE THE DEFAULT FILE SYSTEM TO USE
SMNT POWER
*
ATTACH SERVER-1
ATTACH SERVER-2
ATTACH SERVER-3
*
/*
/&
* $$ E0J
```

Notes: Up to four **ATTACH** commands can be used in each FTP Server partition to attach FTP Server subtasks.

When the IBM provided security exit BSSTISX is used, make sure that you copy phase BSTTFTS1 as phase BSTTFTSX into a configuration `lib.sublib`.

BSTTFTPS uses the SMNT command to mount a particular file system. The following supported file systems are available:

- ▶ SMNT POWER (mounts VSE/POWER)
- ▶ SMNT SAM (mounts SAM file system)
- ▶ SMNT VSAM <catalog.name> (mounts a VSAM catalog)
- ▶ SMNT LIBRARY <libname> (mounts a library)
- ▶ SMNT NULL (mounts a null filesystem)

The following example is for using a Windows FTP script to retrieve an LST queue member. For more information about scripting with the MS command line FTP client, see this website:

<http://support.microsoft.com/kb/96269>

In a Windows command prompt, you start an FTP script, as shown in the following example:

```
ftp -s:ftptest.txt
```

The `ftptest.txt` file contains the commands that are shown in Figure 3-11.

Example 3-11 MS Windows FTP script

```
open 9.152.86.91
userid
password
cd smnt-power
cd lst
ascii
get class.number.name
quit
```

Note: BSTTFTPS allows specifying generic values, such as the use of wildcards. For example, if you want to download the first queue entry with a name in class A, you write `A.0.name`.

For more information about setting up FTP clients for use with BSTTFTPS, see 6.2.2, “FTP clients” on page 199.

3.4.3 VSE as a client

An FTP client is provided by the BSTTFTPC utility. Example 3-12 shows typical JCL that uses BSTTFTPC.

Example 3-12 FTP client JCL

```
// JOB BSTTFTPC - FTP CLIENT FOR BSI STACK
// LIBDEF PHASE,SEARCH=(PRD2.CONFIG,PRD2.TCPIPB)
// LIBDEF SOURCE,SEARCH=(PRD2.TCPIPB)
// EXEC BSTTFTPC,SIZE=BSTTFTPC,OS390
ID 02
```

```
OPEN 9.152.222.71
USER myuser
PASS mypasswd
*
QUIT
/*
/ &
```

While FTP of ICCF members is not supported by using the BSTTFTPS server, you can do this with BSTTFTPC. The technique is to read from SYSIPT and use SLI to include the ICCF member with JCL, as shown in Figure 3-13.

Example 3-13 FTP client JCL showing how to FTP an ICCF member

```
// EXEC BSTTFTPC,SIZE=BSTTFTPC
ID ..
USER ...
PASS ...
*
INPUT SYSIPT
TYPE A
STOR file.name
*
QUIT
/*
* $$ SLI ICCF=...,LIB=...
/*
```

3.5 Setting up TN3270

In this section, we describe how to set up TN3270, which involves configuring VTAM and setting up the JCL for the BSI Telnet server. We then provide examples of using various Telnet clients, such as, IBM Personal Communications, Open Text Exceed, and wc3270.

3.5.1 Setting up VTAM

In this section, we describe the VTAM setup for a TN3270 server. The following steps must be performed:

- ▶ Add VTAM applications to ATCCON00.B
- ▶ Catalog the new VNETAPPL B-book
- ▶ Activate the modified setup

Changing the ATCCON00.B book

You must add the following VTAM application to the ATCCON00.B member:

```
VNETAPPL
```

Cataloging the VNETAPPL.B book

You can use a job that is similar to what is shown in Example 3-14 to catalog the VNETAPPL B-book.

Example 3-14 JCL to catalog a VNETAPPL B-book

```
* $$ JOB JNM=VNETAPPL,CLASS=0,DISP=D
* $$ LST CLASS=0,DISP=D
// JOB VNETAPPL
// EXEC LIBR,SIZE=256K,PARM='MSHP'
ACCESS S=PRD2.CONFIG
CATALOG VNETAPPL.B REPLACE=YES
VNETAPPL VBUILD TYPE=APPL
BSTTVNET APPL
BSTTUSST APPL AUTH=(PASS,ACQ)
VNETTRM GROUP MODETAB=IESINCLM,DLOGMOD=SP3272QN
T001 APPL AUTH=(ACQ),EAS=1
T002 APPL AUTH=(ACQ),EAS=1
T003 APPL AUTH=(ACQ),EAS=1
T004 APPL AUTH=(ACQ),EAS=1
T005 APPL AUTH=(ACQ),EAS=1
T006 APPL AUTH=(ACQ),EAS=1
T007 APPL AUTH=(ACQ),EAS=1
T008 APPL AUTH=(ACQ),EAS=1
T009 APPL AUTH=(ACQ),EAS=1
T010 APPL AUTH=(ACQ),EAS=1
/+
/*
/&
* $$ E0J
```

The APPL names must match with the TERMINAL names in the BSTTVNET JCL, as shown in Example 3-15.

Activating the changed VTAM book

To activate the changed VTAM book, enter the following command:

```
V NET,ACT,ID=VNETAPPL
```

3.5.2 Starting a TN3270 server

The JCL that is shown in Example 3-15 starts the TN3270 server that uses the IPv4 stack.

Example 3-15 JCL to start a TN3270 server in a IPv4 stack

```
* $$ JOB JNM=BSITN44,CLASS=S,DISP=D
* $$ LST CLASS=S,DISP=D
// JOB BSITN44
// OPTION LOG,PARTDUMP,NOSYSDDUMP
// LIBDEF *,SEARCH=(PRD2.CONFIG)
// OPTION SYSPARM='44'
// EXEC BSTTWAIT,SIZE=BSTTWAIT
// EXEC BSTTVNET,SIZE=BSTTVNET,DSPACE=3M,TASKS=ANY,OS390
ID 44
OPEN VSEC01 1023
*
```

```

APPLID DBDCCICS CICS/TS AND ICCF
APPLID BSTTVNET PRINTER SHARING APPLICATION
*
TITLE *** ** Welcome to vse C01 test system *** **
*
TERMINAL T001 GENERIC
TERMINAL T002 GENERIC
TERMINAL T003 GENERIC
TERMINAL T004 GENERIC
TERMINAL T005 GENERIC
TERMINAL T006 GENERIC
TERMINAL T007 GENERIC
TERMINAL T008 GENERIC
TERMINAL T009 GENERIC
TERMINAL T010 GENERIC
*
ATTACH TN3270E
/*
/&
* $$ E0J

```

When you have many terminals, it is more convenient to include the list of `TERMINAL` statements by using `* $$ SLI`. This way, the actual BSTTVNET commands are stored in ICCF or a z/VSE library member. You can change the BSTTVNET startup command without changing the POWER JCL. You can use an editor to duplicate the `TERMINAL` commands and modify only the LUNAME for each line.

Note: BSTTVNET supports up to 2000 sessions per BSTTVNET partition. BSI recommends defining printer sessions in a separate BSTTVNET partition.

Generally, BSI suggests dividing many sessions between multiple BSTTVNET partitions. After you define over 1,000 sessions in one BSTTVNET partition, it is suggested that a new BSTTVNET partition is used for more new sessions.

3.5.3 Controlling the terminal type

The terminal type is controlled by when a terminal session is opened. The session is installed automatically or the session (terminal) is defined in RDO (DFHCSD file).

If the session is installed automatically (this is often the case), the auto-install process selects the `TYPETERM` from a CICS table. If the session is defined in RDO, the CICS `TYPETERM` that is used is specified in the RDO entries for the terminal.

For more information about the CICS `TYPETERMs` that are used in the `DFHTERM` group for auto installation (`TYPETERM` definitions in group `DFHTYPE`), see *CICS Resource Definition Guide*, SC33-1653.

The BSTTVNET server passes the VTAM logmode to CICS. This logmode is used to select a `TYPETERM` in the CICS auto installation routines. Because all BSTTVNET `TERMINAL` sessions are non-SNA, the `DFH3270` `TYPETERM` is normally used.

If the `VSESPG` RDO group is used for automatic installation, the `VSE3278Q` `TYPETERM` often is used. This is the appropriate `TYPETERM` because it queries the `TN3270(E)` client to determine the correct configuration.

Note: Unless you have a specific need for a true TN3270E session, BSI recommends the use of TERMINAL sessions. There is less processor usage for TERMINAL sessions when compared to TN3270E.

3.5.4 Connecting with IBM Personal Communications

In our plain IPv4 setup, we used IBM Personal Communications as the TN3270 client. For Host Name or IP Address, enter the IP address of the VSE system (9.152.85.64) with port number 1023, as shown in Figure 3-10.

	Host Name or IP Address	LU or Pool Name	Port Number
Primary	9.152.86.58		1023
Backup 1			23
Backup 2			23

Connection Options

Connection Timeout: 6 Seconds

☒ Auto-reconnect

☐ Try connecting to last configured host infinitely

Keep Alive

☐ Enable Telnet Keep Alive: NOP

Keep Alive Time Out: 180 Seconds

Figure 3-10 Telnet 3270 window in IBM Personal Communications

After you connect to VSE, the VTAM selection window opens, as shown in Figure 3-11.

```
TCP/IP-TOOLS TN3270-E Server Version U 249 9.152.85.64 11/11/2010
Application Selection Menu 9.152.222.135 14:21:19
*** Welcome to use C01 test system ***
Name      Status      Description
BSTTUNET  U-Inactive  PRINTER SHARING APPLICATION
DBDCCICS  U-Ready    CICS/TS AND ICCF
```

Figure 3-11 VTAM selection window of IPv6/VSE

Move the cursor to DBDCCICS and press Enter to open the VSE sign-on window.

3.5.5 Connecting with Open Text Exceed

The Open Text Exceed product was formerly known as Hummingbird Exceed and provides various connectivity tools. In the Hummingbird Neighborhood folder, double-click a Default 3270 icon and configure the session to VSE.

In the 3270 session window, select **Options** → **Session Properties**, then specify the VSE IP address and TN3270 port, as shown in Figure 3-12.

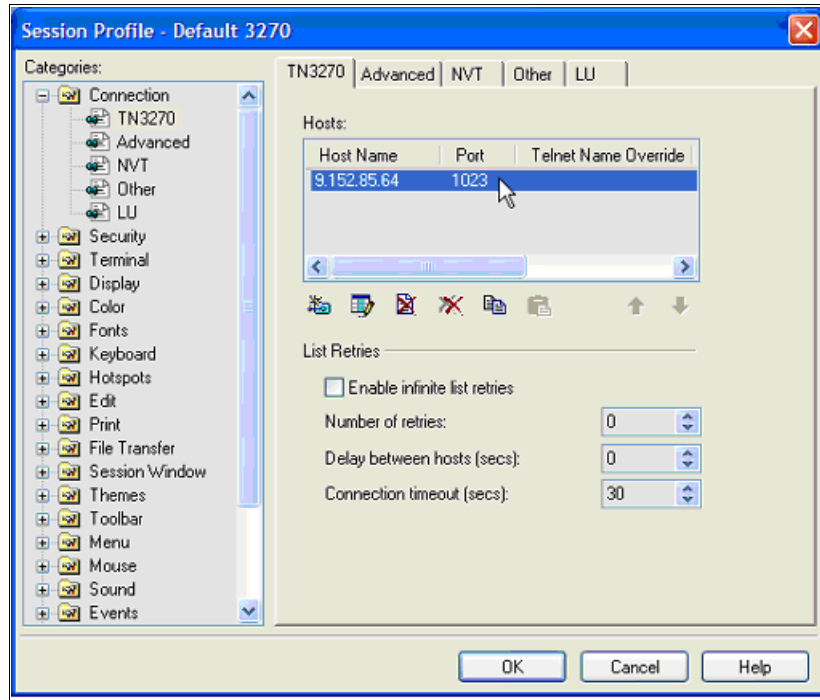


Figure 3-12 Configuring the Session properties in Open Text Exceed

Select **File** → **Connect** to open the VTAM selection window, as shown in Figure 3-13.

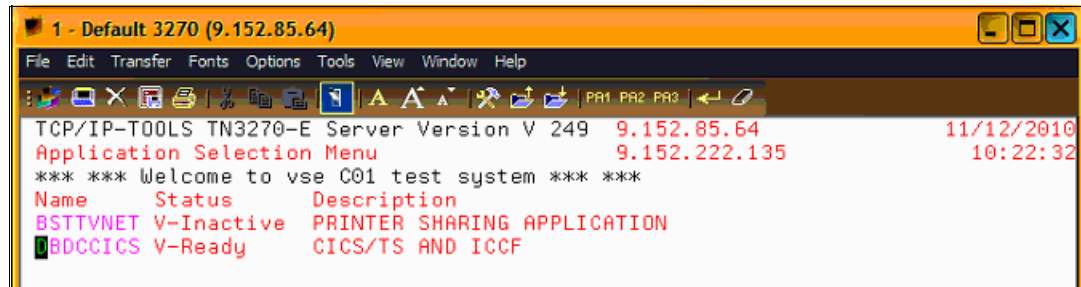


Figure 3-13 VTAM selection window in IPv6/VSE

3.5.6 Connecting with wc3270

The wc3270 emulator is a IBM 3270 terminal emulator for the X Window System and Windows available at no extra cost. It is available at this website:

<http://x3270.bgp.nu>

By using the wc3270 session wizard, you can create wc3270 sessions. Because we are not using SSL, choose SSL Tunnel = NO when you are creating your wc3270 session, as shown in Example 3-16.

Example 3-16 Setting up a wc3270 session

1. Host	: vsessl
2. Logical Unit Name	: (none)
3. TCP Port	: 23
4. Model Number	: 4 (43 rows x 80 columns)
5. Oversize	: (none)
6. Character Set	: bracket (CP 37*)
7. SSL Tunnel	: No
8. Proxy	: (none)
11. wpr3287 Printer Session	: No
15. Keymaps	: (none)
17. Font Size	: 12
18. Background Color	: black
19. Menu Bar	: Yes

Double-click the session icon to start the TN3270 session, as shown in Figure 3-14.

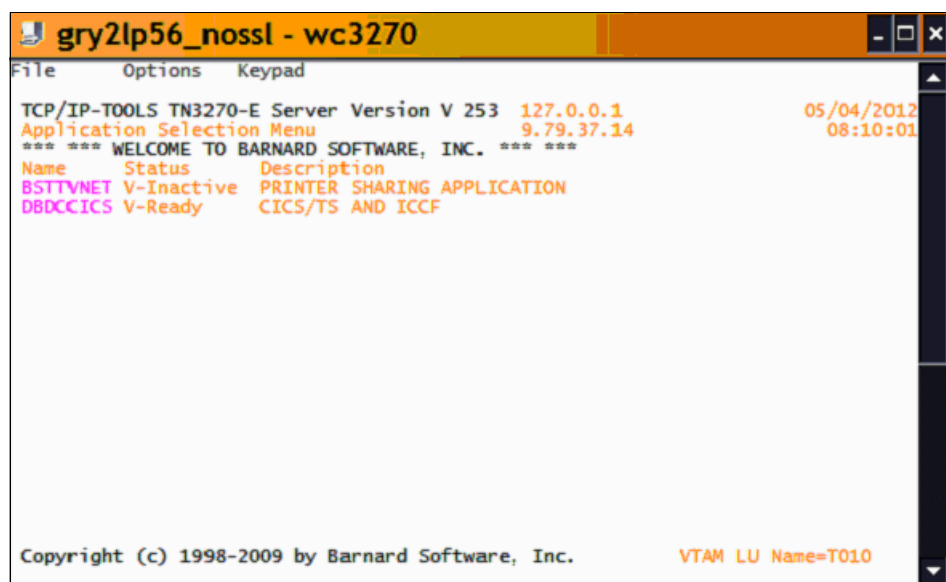


Figure 3-14 VTAM selection panel that uses the wc3270 terminal emulator

3.5.7 Recovering from broken connections

IPv6/VSE provides two commands for recovering from interrupted connections.

- ▶ REACT *luname*

The REACT command forces a termination by closing the VTAM ACB for the *luname* and then issues a VTAM OPEN again. Effectively, this forces the application connected to the ACB to release it and the TCP socket is also closed.

- ▶ KILL *luname*

Terminates a session normally. The KILL command posts a session termination ECB. On the VTAM side, a TERMSESS is issued. On the TCP side, the socket is closed.

Closing the VTAM ACB for the *luname* being used forces CICS to clean up the session and release the resource. A proper DFHZNEP phase is also helpful here. IPv6/VSE provides three samples (source) for a ZNEP phase.

3.6 Setting up TN3270E printing

In this section, we describe how to set up TN3270E printing.

3.6.1 TN3270E setup

Example 3-17 shows a simple BSTTVNET startup for printers. BSTTVNET accepts connections that use port 5023 for the three defined TN3270E printer sessions. The DIRECT/LPR/FTP/FTPP sessions are non-SNA. For more information, see the chapter that is titled “Printing without using a TN3270E Client” in *IPv6/VSE User's Guide*, SC34-2618. The DIRECT and LPR type printer session are widely used.

Example 3-17 BSTTVNET JCL for printers

```
// OPTION SYSPARM='10'
// EXEC BSTTWAIT,SIZE=BSTTWAIT
/*
// EXEC BSTTVNET,SIZE=BSTTVNET,DSPACE=3M
ID 10
OPEN 127.0.0.1 5023
*
APPLID BSTTVNET PRINTER SHARING APPLICATION
*
PRINTER P001 BSTTVNET
PRINTER P002 BSTTVNET
PRINTER P003 BSTTVNET SCS
*
DIRECT DIR BSTTVNET 192.168.1.101 9100 *
LPR LPR BSTTVNET 192.168.1.60 * * hp2505
FTP FTP BSTTVNET 192.168.1.60 * * tstuser tstpswd
FTPP FTTP BSTTVNET 192.168.1.60 * * tstuser tstpswd
*
ATTACH TN3270E
/*
```

BSI recommends running a separate BSTTVNET partition for TERMINAL and PRINTER/DIRECT/LPR sessions.

Table 3-1 shows the CICS typeterms and VTAM log modes to be used for particular printers.

Table 3-1 CICS Typeterms and VTAM Logmodes for printers

Printer	CICS Typeterm	VTAM Logmode
CICS printer	VSEDSCP	SPDSCPRT
SNA LU1 SCS printer	VSESCSPA	SPSCSPRT

For more information, see the chapter that is titled “Using the TN3270E Server” in the *IPv6/VSE User's Guide*, SC34-2618.

3.6.2 BSTTVNET JCL

The JCL that is shown in Example 3-18 starts an IPv6/VSE printer server to use the sample printers. This server setup is only for printers and should be connected through the IP4 stack.

Example 3-18 JCL for a IPv6/VSE printer server

```
* $$ JOB JNM=BSITELNP,CLASS=H,DISP=D
* $$ LST DISP=H,CLASS=Q
// JOB BSITELNP Printer Server for TELNET ACCESS FROM PCOMM V6
// LIBDEF *,SEARCH=(PRD2.CONFIG,BARNARD.IPV6)
// OPTION SYSPARM='44'
/*
// EXEC BSTTVNET,SIZE=BSTTVNET,DSPACE=3M,OS390,TASKS=ANY
ID 44
OPEN 192.168.1.61 2023
SBCS DN_03
*
APPLID BSTTVNET PRINTER SHARING APPLICATION
*
PRINTER TADV3 BSTTVNET
PRINTER TADV4 BSTTVNET SCS
PRINTER TADV5 BSTTVNET SCS
PRINTER TADV6 BSTTVNET SCS
PRINTER TADV7 BSTTVNET
PRINTER TADV8 BSTTVNET
PRINTER TADV9 BSTTVNET
ATTACH TN3270E
/*
/&
* $$ EOJ
```

3.6.3 Node error program

Example 3-19 shows a sample node error program. In our setup, the ZNEP example from BSI caused a loop in IESX (highlighted in bold text).

Example 3-19 Sample node error program

```
// JOB IESZNEP ASSEMBLE
// LIBDEF *,CATALOG=PRD2.CONFIG
// LIBDEF SOURCE,SEARCH=(PRD1.BASE,PRD1.MACLIB)
// OPTION ERRS,SXREF,SYM,CATAL,NODECK
  PHASE DFHZNEP,*
  INCLUDE DFHEAI
// EXEC ASMA90,SIZE=(ASMA90,64K),PARM='EXIT(LIBEXIT(EDECKXIT)),SIZE(MAXC
    -200K,ABOVE)'
* $$ END
// ON $CANCEL OR $ABEND GOTO ENDJ2
// OPTION NOLIST,NODUMP,DECK
// EXEC DFHEAP1A,SIZE=512K
*
      PUNCH ' CATALOG IESZNEP.OBJ REP=YES'
      DFHSNEP TYPE=INITIAL,NAME=DFHZNEP
*
```

```

NEPBASE EQU 10                                ERROR PROCESSOR BASE REGISTER
*
      DFHSNEP TYPE=ERRPROC,                                X
      CODE=(10,11,D1,A7,61,57,49),GROUP=01
      USING NEPROC01,NEPBASE  DEFINE BASE
      LR NEPBASE,EPBAR      LOAD BASE REGISTER
      ST 14,NEPEPRS        SAVE RETURN REGISTER
*
* CATCH POWER OFF AT SNA TERMINALS
  CLI TWAEC,X'61'          IS ERRORCODE=61 ?
  BNE NOT61                ..NO
  CLC TWAENSR(2),=X'0831'  IS SENSECODE=0831 ?
  BNE RETURN              ..NO
  OI TWAOPT3,TWAONCN      SET CLSDST BIT
  B ACTION
*
NOT61 DS OH
ACTION DS OH
*
      EXEC CICS LINK PROGRAM('IESCLEAN') COMMAREA(NEPCABEG) ,
*
* The two lines below from the Barnard example caused a loop in IESX.
* Removing these lines solved the problem.
*      CLI TWAEC,X'10'          EC=10?
*      BE YESD1                NO.
      CLI TWAEC,X'11'          EC=11?
      BE YESD1                NO.
      CLI TWAEC,X'D1'          EC=D1?
      BNE NOTD1                NO.
*
YESD1 DS OH
      OI TWAOPT3,TWAOINT      SET CREATE
      NI TWAOPT3,X'FF'-TWAONINT RESET NOCREATE
NOTD1 DS OH
*
* Link to a vendor Znep
      EXEC CICS LINK PROGRAM('MENUNEP')
      COMMAREA(DFHNEPCA) LENGTH(=AL2(NEPCALEN))
*
RETURN DS OH
      L 14,NEPEPRS          RESTORE REGISTER
      BR CSVTBAR            RETURN TO NEP
      LTORG
      DFHSNEP TYPE=FINAL
      END DFHNEPNA          NEP ENTRY POINT
/*
/. ENDJ2
// EXEC IESINSRT
/*
// IF $MRC GT 4 THEN
// GOTO NOLNK
// EXEC LNKEDT,SIZE=256K,PARM='MSHP'
/. NOLNK
/&
* $$ EOJ

```

* \$\$ END
/&

3.7 Setting up SSL

In this section, we describe how to set up SSL for the IPv6/VSE stack. SSL requires service level 253pre03-ibm or later and OpenSSL support in z/VSE 5.1 (z/VSE Cryptographic Services, 5686-CF9-17, CLC=51S). Check for phase IJBSSL in PRD1.BASE.

The full support is provided in the following PTFs:

- ▶ DY47414 / UD53863: VSE/AF update for HW crypto support
- ▶ DY47499 / UD53983: OpenSSL, upgrade to version 1.0.1e
- ▶ PM98875 / UK98397: IPV6/VSE, support of TLSV1.2 parameter

3.7.1 Installing the prerequisite programs

You can create the OpenSSL keystore by using the Keyman/VSE tool or OpenSSL on a workstation platform. Install both programs on your system so that they are available.

OpenSSL-Light

OpenSSL-Light for Windows is available at this website:

<http://www.slproweb.com/products/Win32OpenSSL.html>

This website features the following links:

- ▶ Win32 OpenSSL v1.0.0d Light
- ▶ Visual C++ 2008 Redistributable package (you need these runtime components for OpenSSL)

First download and install the Visual C++ Redistributable package, then install OpenSSL Light.

Keyman/VSE

Download and install the latest version of the Keyman/VSE tool from the VSE, which is available at this website:

<http://www.ibm.com/systems/z/os/zvse/downloads/#vkeyman>

Note: The Keyman/VSE tool needs the VSE Connector Client that can be downloaded and installed from this website:

<http://www.ibm.com/systems/z/os/zvse/downloads/#vsecon>

3.7.2 Creating the keystore

With the latest version of Keyman/VSE (build date January 2013 or later), you can create a .pem file and upload it to VSE with a few mouse clicks.

In a test environment, you might want to use self-signed certificates. In a production environment, you should use certificates that are issued by a certificate authority (CA).

Complete the following tasks to create a .pem file that is used on VSE by OpenSSL by using Keyman/VSE and OpenSSL on a workstation.

Creating the keystore using Keyman/VSE

Complete the following steps to create and upload your .pem file by using Keyman/VSE:

1. Create an RSA key, SSL, and SSL root certificate, as shown in Figure 3-15.

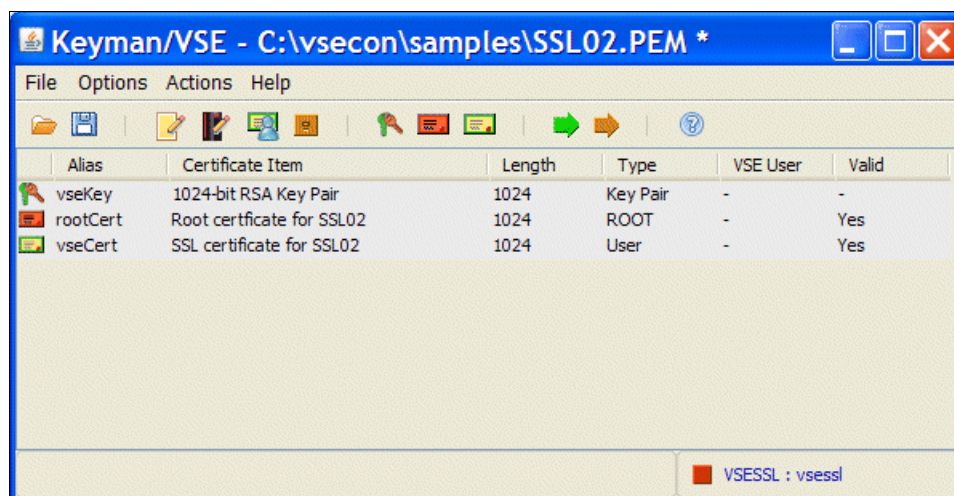


Figure 3-15 Keyman/VSE main window

2. Click **Local Properties** and specify your .pem file name, as shown in Figure 3-16.

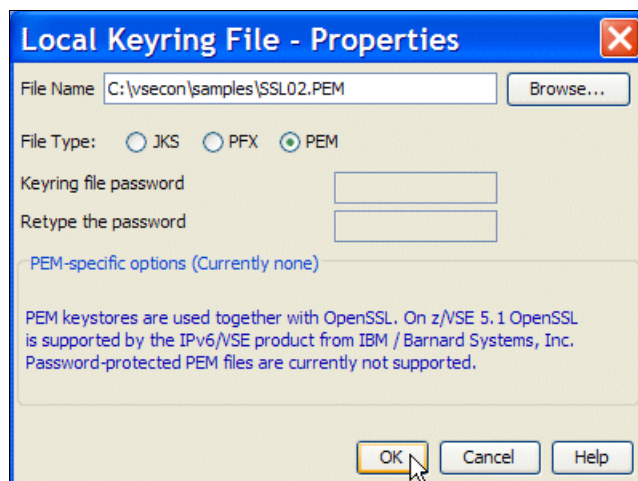


Figure 3-16 Keyman/VSE local keyring file properties window

3. Click **Save as PEM file on VSE...** to upload your .pem file to VSE, as shown in Figure 3-17.

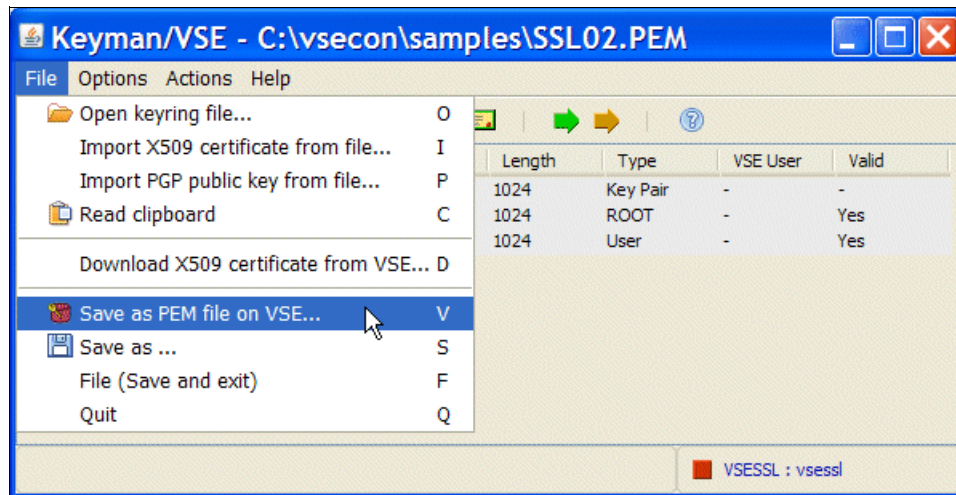


Figure 3-17 Keyman/VSE upload .pem file to VSE

4. Click **File** → **Save as PEM file on VSE**, the click **Upload**, as shown in Figure 3-18.

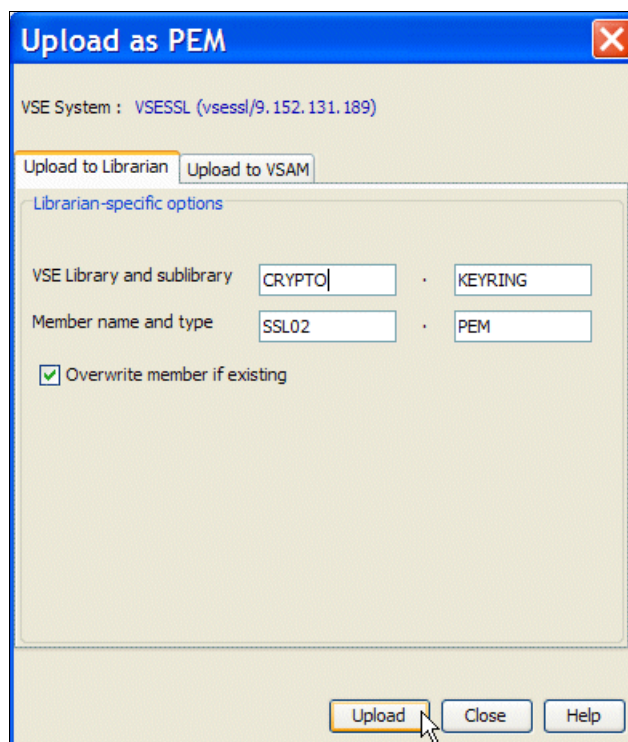


Figure 3-18 Keyman/VSE upload as .pem window

In the next section, we describe another method that can be used to create an OpenSSL keystore by using OpenSSL on your system. Other keystore-related issues also are described.

Creating the keystore using OpenSSL

In this section, we describe how to create your .pem file by using OpenSSL that is installed on MS Windows. This process is more complicated than the use of Keyman/VSE; however, it might give you more flexibility. In this case, we use the Keyman/VSE tool as a CA for signing the certificate request that was created by using OpenSSL.

Complete the following steps:

1. Create an RSA key and certificate request by using OpenSSL-Light on your system.

The following command can be used to create a .pem file that contains an RSA key pair with OpenSSL-Light on Windows:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ssl01.pem  
-out myreq.pem
```

The **-nodes** parameter stands for “no DES encryption” and causes the .pem file to be created without a password. Password-protected .pem keystores are not supported by the IPv6/VSE stack.

2. Create a self-signed root certificate, as shown in Figure 3-19.

Open the Keyman/VSE tool and create a self-signed root certificate. We use this certificate later for signing the certificate request that was created in the previous step.

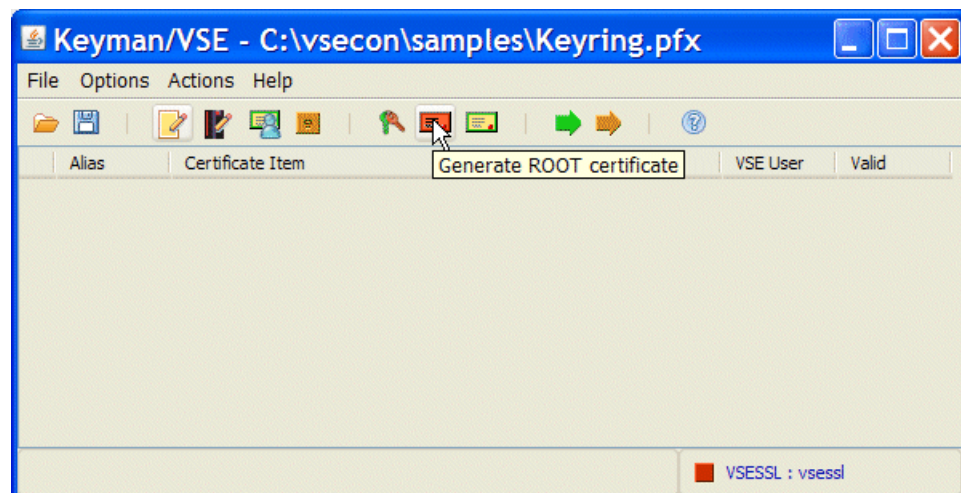


Figure 3-19 Keyman/VSE generate self-signed root certificate

3. Sign the certificate request.

Open the previously created .pem file that contains the certificate request (myreq.pem) with a text editor and copy the request to the clipboard. Copy the entire text, including the BEGIN/END delimiter lines, as shown in Figure 3-20.

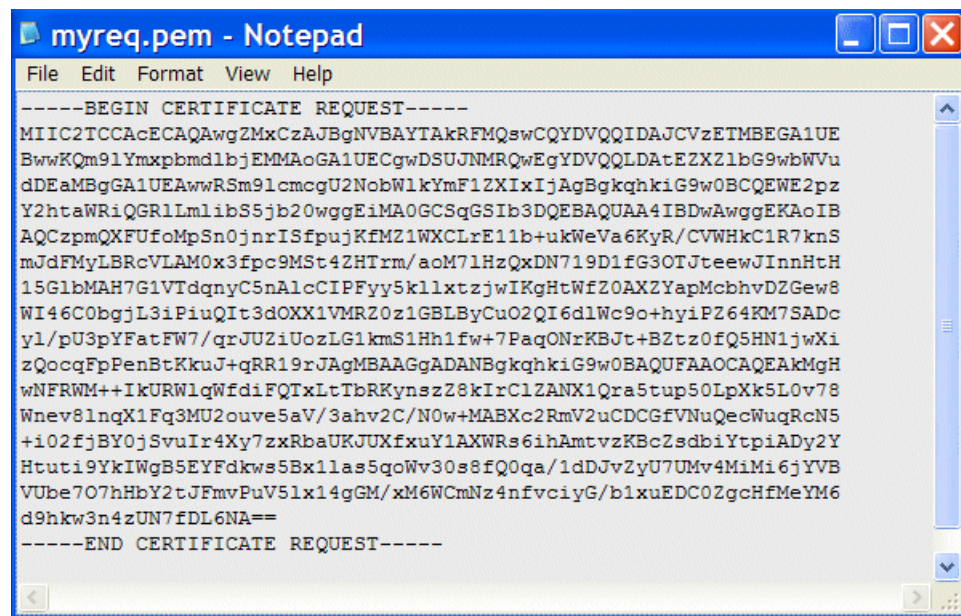


Figure 3-20 Editor window that shows the contents of myreq.pem

In the Keyman/VSE window, right-click the root certificate and select **Sign certificate request**. Paste the certificate request into the text area in the Sign certificate request window, as shown in Figure 3-21.

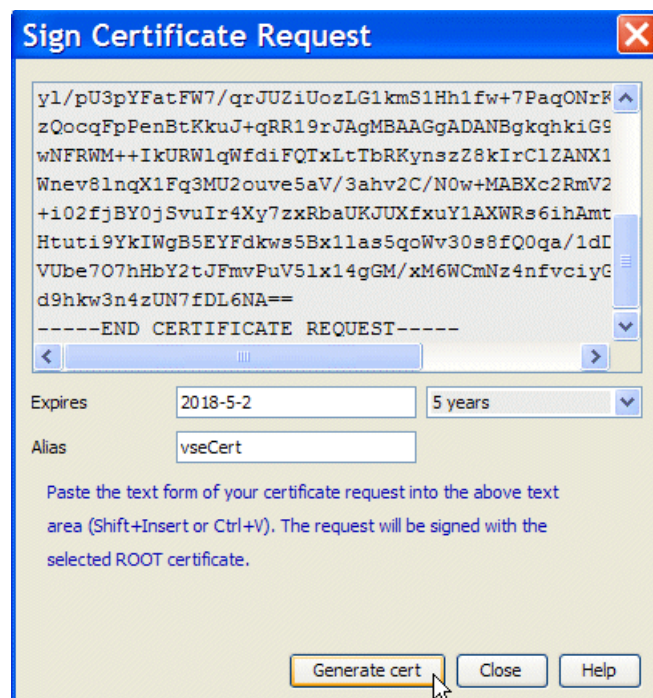


Figure 3-21 Sign certificate request window in Keyman/VSE

- Click **Generate cert.** You now have two certificates in the Keyman/VSE main window, as shown in Figure 3-22.

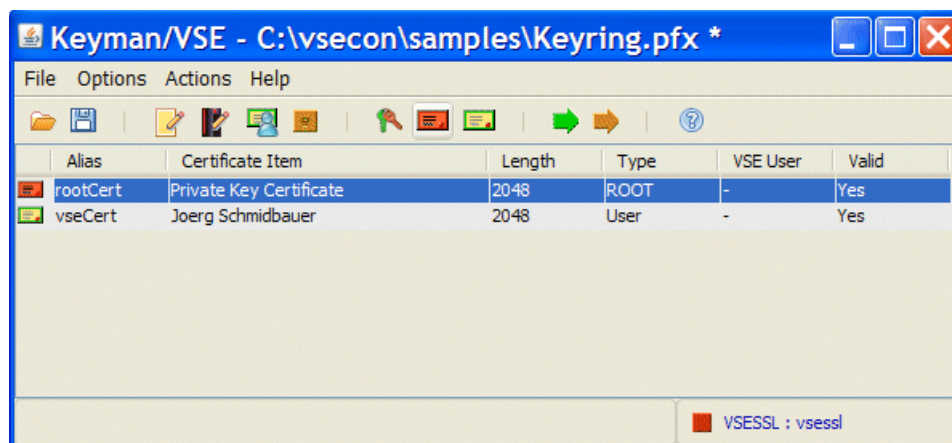


Figure 3-22 Keyman/VSE main window

- Copy certificates into the .pem file, as shown in Figure 3-23.

In the Keyman/VSE main window, copy and paste both certificates into the .pem file by right-clicking a certificate and selecting **Copy to clipboard**.

Important: The VSE certificate must be placed before the root certificate in the .pem file so that the order is: **key - vsecert - rootcert**.

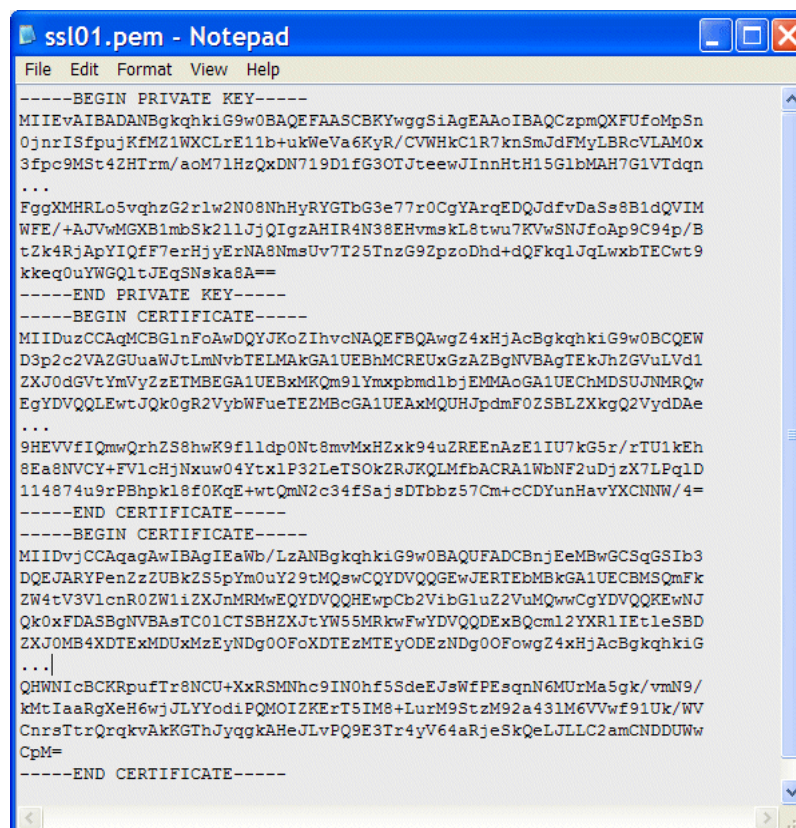


Figure 3-23 Editor window that shows a .pem file

The .pem file now contains the key and two certificates.

6. Save the .pfx and .pem files. The .pfx file is used on the client side; the .pem file must be uploaded to VSE and then used by BSTTATLS.
7. Upload the .pem file to VSE.

After the .pem file is created, it must be uploaded to VSE with ASCII to EBCDIC translation, either as a VSE library member or as a VSAM file.

When the .pem file is created on Windows, there is a problem with the line-end characters. The openssl.exe creates UNIX style line breaks with only "line-feed" (hex 0A). On Windows, we usually have "carriage-return/line-feed" CRLF (hex 0D0A).

Without having CRLF line endings, FTP or VSE Connector Server cannot transfer the file to VSE with ASCII to EBCDIC translation. Therefore, this issue must be fixed before the .pem file is uploaded to VSE.

On Windows, line endings can be converted by using the WordPad editor.

Note: On Windows XP, the notepad.exe cannot read UNIX style text files correctly. To convert the line endings with WordPad, edit the .pem file and save it back to disk.

After the .pem file is uploaded (see Figure 3-24) as a VSE library member SSL01.pem in CRYPTO.KEYRING, you can configure the SSL proxy server.

```
Session E - [43 x 80]
File Edit View Communication Actions Window Help
Process View Options Help

DITTO/ESA for USE          LE - Library Member Edit

Member SSL01.PEM          Library CRYPTO.KEYRING      Col 1      Format CHAR
                               SYSIPT data NO

1...5...10...5...20...5...30...5...40...5...50...5...60...5...70...
00000 ***** Top of data *****
00001 -----BEGIN PRIVATE KEY-----
00002 MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBAQwggSiAgEAAoIBAQCzpmQXFUfoMpSn
00003 0jnRISfpujKfMZ1WXLRE11b+ukWeUa6KyR/CUWHkC1R7knSmJdFMyLBRcULAM0x
00004 3fpc9MSt4ZHTrm/aoM71HzQxDN719D1fG30TJteewJInnHtH15GlbMAH7G1Utdqn
00005 uC5nAlcCIPFuy5k11xtzjwIKgHtWfZ0AXZVapMcbhvDZGw8WI46C0bgjL3iPiuQ
00006 It3d0XX1UMRZ0z1GBLByCu02QI6d1Wc9o+hyiPZ64KM7SA0cy1/pU3pVFatFW7/q
00007 rJUZIuozLG1kmS1Hh1fw+7Paq0NrKBjt+Bztz0fQ5HN1jwXizQocqFpPenBtKkuJ
00008 +qRR19rJAgMBAAECggEAY+DaPMskE0ArzbHfaY4hdpyCkGcxJ1ZLQ46c1QrG1CwZ
00009 v59ExywUUS0fRTHJ8T/MujhX1kRIEA7+Bf93tj6PKm0CekG9BjupSxEyHyMcwn1r
00010 tXi5pNIvhp9hoowpLiP3VZU4ni8gChEiw61Tuw1Z/mD6W+w91NxR8s1LRTNxXu/A
00011 wIA60x6krSSsP7BZ4syJ0jiggaUukxwXsVa4uUD1qfaxR+01JA569s7IJZ59kRDi
00012 U09ycJHPdcBptYTFHI/o8GPXe8WUMDTwTfqouuTKN/L0Z5HuGTFZjbCkdHALWHwF
00013 xI/8Ipt8HyQKn3PtrnmWuPmBUEZn2caHj+khYyM4QKKBgQDcTDuT2oEBaiFELWv1
00014 p/J0OMOR0fQ6SKsGiCP0xnjJiczXPAUN2g520x5T7getdU0uCMgEDtIREYx7aoF4
00015 U0e6aoDk+KNgQpEP8FTZ79R0stp4LNQip3rW99zBzdW1rEW66J6wm5yiWC/pkMjt
00016 JIwrxM2/yfD21+nUk6JA7uam5QKBgQDQw8B0qrRI6ZFwPdpNyud9DRg/nKucbYKL
00017 JmqDw19b1a/2bgI4AjH2LGBtU32mpEFLIhk10F0+THCiWdJ5Aujx2jw8EmSC6L5
00018 BypaIwHGbmcrBqps03Tk+cWb/HJGertB8uGww1q2ycUFb3fFrMG3YsqGYSGfHC0
00019 wFuF897iFQKBgCBuNOJa7F1587Xm/hsoA/wdr2Wbh5shMdT/XESIwGZL3JXf52QD
00020 89c41CFFTW0TeSOJsrD0Qc1ziPxuu0srjQac0af015FHm7X+C3Ae6GfaiWJKL11+
00021 zQ4z2hQ+quUzjNatAsg7gP9B/qPjxj2YBfo50YR9BFV4fZwNZXkr81e1AoGAXkx9
00022 y3UsoF8go1U0bMNBa1AU6u3z+vJQbFbu/mSJfx+uGbubkE66SKN0Hot9su3Aaracs
00023 +NWaK3U0kAS/13I1+9XgRi0T319ZP0mIT0zRKXP1M3reFFeEYRRe5tisC52ka2sB
00024 FggXMHRL05vqhG2r1w2N08NhHyRYGTbG3e77r0CgYArqEDQJdFuDaSs8B1dQUIM
00025 WFE/+AJUwMGXB1mbSk211JjQIgZAHIR4N38EHvmskL8twu7KUwSNJfoAp9C94p/B
00026 tZk4RjapYIQf7erHjyErNA8NmsUu7T25TnzG9ZpzoDhd+dQFk1JLqLxbTECwt9
00027 kkeq0uYwGQ1tJEqSnska8A==
00028 -----END PRIVATE KEY-----
00029 -----BEGIN CERTIFICATE-----
00030 MIIDuzCCAqMGBGlnFoAwDQYJKoZIhvcNAQEFBQAwgZ4xHjAcBgkqhkiG9w0BCQEW
00031 D3p2c2UAZGuaWJtLmNubTELMakGA1UEBhMCREUeGzAZBGNuBAGTEKJhZGUUdUd1

Command ==>
F1=Help F2=SplitJoin F3=Save+Exit F4=Left F5=Right F6=RFind F7=Bkwd
F8=Fwd F10=RChange F11=CRetrieve F12=Quit

41/015
Connected to remote server/host vsssl using port 23 Print to Disk - Append
```

Figure 3-24 z/VSE DITTO display of a .pem file in VSAM

When the file is uploaded correctly, its contents are readable Base64-encoded text.

Verifying the keystore

OpenSSL provides a verify function to check a .pem file for correctness. When we verified our .pem file we received the following error:

```
C:\vsecon\samples>openssl verify test.pem
test.pem: emailAddress = zvse@de.ibm.com, C = DE, ST = Baden-Wuerttemberg, L = B
oeblingen, O = IBM, OU = Development, CN = 9.152.131.189
error 20 at 0 depth lookup:unable to get local issuer certificate
```

The error 20 is issued because we are using a self-signed root certificate and OpenSSL cannot decide whether to trust it. In our case, we can ignore this error.

If the order of the two certificates is not correct, that is, the SSL certificate does not appear before the root certificate, you receive the following error:

```
C:\vsecon\samples>openssl verify test.pem
test.pem: emailAddress = zvse@de.ibm.com, C = DE, ST = Baden-Wuerttemberg, L = B
oeblingen, O = IBM, OU = IBM Germany, CN = Test CA Certificate
error 18 at 0 depth lookup:self signed certificate
OK
```

OpenSSL now finds the self-signed root certificate first, which is incorrect. By using this .pem file, connections fail.

3.7.3 Removing the private CA key from the client keyring file

There is an important point regarding how the client certificate is selected during the SSL session establishment.

With IBM Personal Communications 5.9 and 6.0, we observed that the first certificate in the keyring file with a private key is selected as the client certificate. In many cases, this is the CA root certificate and not the client certificate we created for this purpose.

The reason for this is that when the SSL certificate is obtained from an external CA, you never receive a private key from the CA. Instead, you receive a “public” CA certificate that contains only a public key. No one ever distributes a private key.

Based on our environment with Keyman/VSE, this means that we must store our self-signed CA root certificate in a safe place, remove the private key from the certificate, and then save the .pfx file.

Removing the private key from a certificate can be easily done by copying the certificate to the clipboard as Base64 test form. This process removes the private key. Reading the clipboard again pastes the root certificate in its “public form” without a private key into Keyman/VSE.

For example, double-clicking the root certificate in Keyman/VSE opens the Properties window, which shows a private and public key pair, as shown in Figure 3-25.

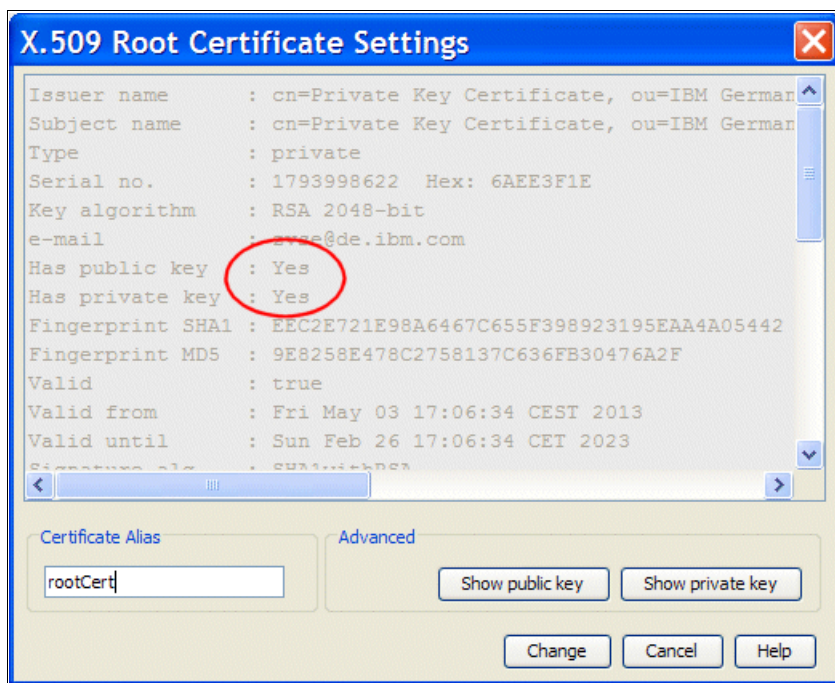


Figure 3-25 Root certificate settings in Keyman/VSE

After the certificate is copied to the clipboard and it is read back, the private key is removed, as shown in Figure 3-26.

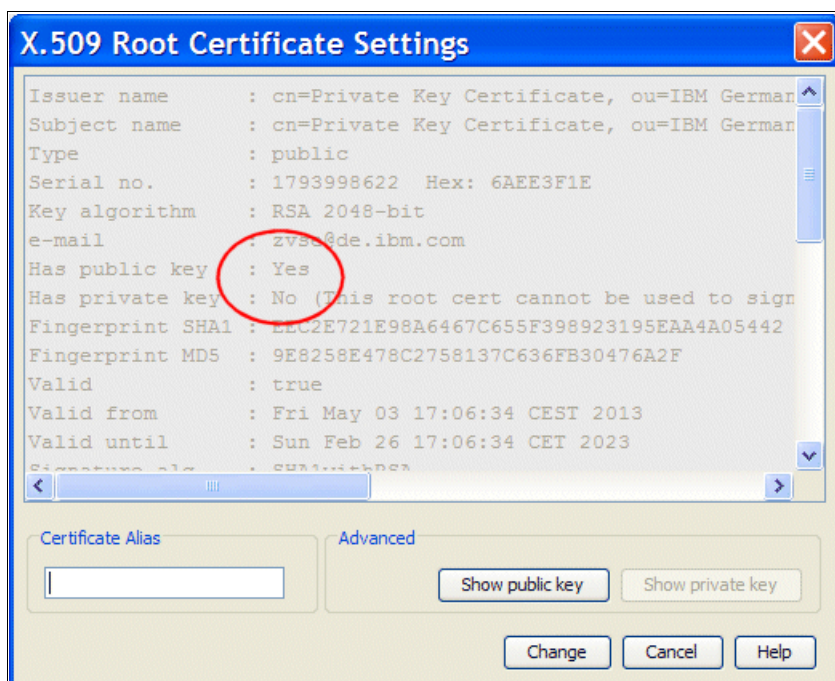


Figure 3-26 Root certificate settings in Keyman/VSE

Save the original root certificate that contains a private key in the Keyman/VSE window in another .pfx file, change the file name to SSL01.PFX by using the Local File properties window, and then save the file, as shown in Figure 3-27.

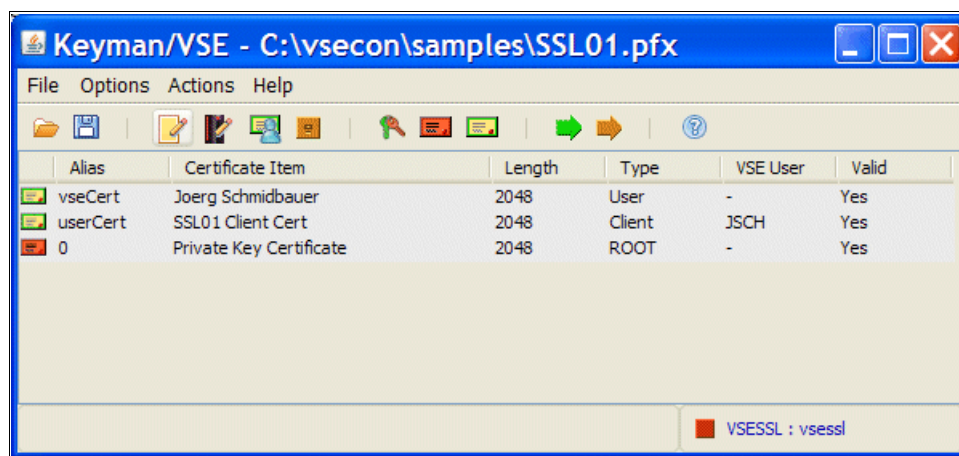


Figure 3-27 Keyring file that contains an SSL client certificate

The file now contains a public root certificate (without a private key), an SSL certificate that is signed by this root certificate, and an SSL client certificate, which also is signed by this root certificate.

The next steps involve configuring a BSI SSL proxy server (BSTTATLS or BSTTPRXY).

3.7.4 Deciding whether to use the SSL proxy server or AT-TLS

The IPv6/VSE stack provides the following two servers that provide SSL for VSE server and client applications. Both servers provide SSL/TLS transparently to the application. They support batch and CICS applications that are written in any supported API, including applications that use the ASM SOCKET macro, EZASMI, EZASOKET, and LE/C APIs:

- ▶ **SSL proxy server (BSTTPRXY)**
BSTTPRXY is a simple proxy server. It allows only a single PROXY command. To proxy multiple connections, you must run multiple BSTTPRXY partitions. BSTTPRXY performs IPv4-to-IPv6 or IPv6-to-IPv4 translation.
- ▶ **AT-TLS server (BSTTATLS)**
Automatic Transport Layer Security is a facility that is similar to the z/OS Application Transparent - Transport Layer Security (AT-TLS) facility. BSTTATLS does much more than BSTTPRXY. It allows many ATTLS definitions and monitors incoming and outgoing connections and intercepts and converts sockets from clear text to SSL or vice versa, as necessary. However, BSTTATLS does not perform IPv4-to-IPv6 or IPv6-to-IPv4 translation.

There are advantages to each application. BSTTPRXY is a proxy server application whereas BSTTATLS is an extension of the APIs to handle intercepting and converting of sockets to or from SSL.

The BSTTPRXY or BSTTATLS servers must be started before any of the applications they are servicing. The serviced applications should be shut down before the proxy or AT-TLS servers are terminated.

Note: BSTTPRXY requires a minimum 20 MB partition.

BSTTATLS requires a minimum 20 MB partition and 72K for each possible socket to be handled.

Ensure that the BSTTATLS/BSTTPRXY partition has a higher priority than the application partition. Otherwise, application hangs might occur.

3.7.5 Specifying parameters

When parameters are specified for BSTTPRXY and BSTTATLS, be aware that some parameters are global (specified only once), while others can be specified separately for specific connections. This is especially important for BSTTATLS, which handles multiple connections.

Consider the following parameter characteristics and tips:

- ▶ The SECTYPE, KEYRING, and KEYFILE are global parameters that set the default values. These parameters are specified only once.
- ▶ The SECTYPE and KEYRING must always be specified.
- ▶ The KEYFILE command can be omitted; however, if it is omitted, you must specify the keyring file on each ATTLS command. On the ATTLS command, KEYFILE is a positional parameter. That is, you do not specify the keyword; only the file name as the last parameter is specified, as shown in the following example:

```
ATTLS 23 AS 992 SSL TEST51
```

- ▶ Use the KEYFILE command for specifying the default value. If you want to change the KEYFILE for an ATTLS connection, you specify the KEYFILE on the ATTLS command.

Note: The KEYFILE parameter on the ATTLS command was not working correctly before APAR PM98875/PTF UK98397.

For more information about parameters, see *IPv6/VSE SSL Installation, Programming, and User's Guide*, SC34-2676.

3.7.6 Configuring the SSL proxy server

The SSL proxy server BSTTPRXY can be started with a JCL similar what is shown in Example 3-20. This example proxies a TN3270 server connection. Secure clients that are connecting to port 992 are forwarded to clear port 23.

We assume that you uploaded your .pem file as VSE library member SSL01.pem in PRD2.CONFIG.

Example 3-20 JCL for a BSTTPRXY server

```
* $$ JOB JNM=BSTTPRXY,CLASS=S,DISP=D
// JOB BSTTPRXY - BSI SSL PROXY SERVER
// OPTION PARTDUMP,NOSYSDUMP,NOLOG
// OPTION SYSPARM='02'
// SETPARM IPTRACE='YYNNNNNN'
/* SETPARM IPTRACE='NNNNNNNN'
// SETPARM SSL$DBG='YES' OR 'NO' (DEFAULT = NO)
// SETPARM SSL$ICA='NO' OR 'NO' (DEFAULT = YES)
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD2.TCIPB,PRIMARY.JSCH,PRD2.SCEEBASE)
// EXEC BSTTPRXY,SIZE=BSTTPRXY,PARM='TRAP(OFF)/'
ID 02
*
KEYRING CRYPTO.KEYRING
KEYFILE SSL01
SECTYPE TLSV1
*
OPTION SERVER
PROXY TCP V4 992 SSL * TO V4 23 TXT * 127.0.0.1
/*
// EXEC LISTLOG
/*
/&
* $$ E0J
```

The PROXY command is different if VSE is the client; for example, when an FTP client is used on VSE, as shown in the following example:

```
OPTION CLIENT
OPTION FTP
PROXY TCP V4 21 TXT * TO V4 990 SSL * 9.152.131.28
```

Here, the VSE FTP client connects to localhost (127.0.0.1), where BSTTPRXY converts the clear connection to SSL and forwards the traffic to the server at 9.152.131.28.

The parameters SSL\$ICA and SSL\$DBG are evaluated by the IJBSSL phase that is part of z/VSE 5.1 Advanced Functions and provides an SSL implementation that is based on OpenSSL.

When SSL\$ICA is YES, cryptographic hardware (crypto cards and CPACF) is used to perform cryptographic operations. SSL\$DBG controls debugging of the IJBSSL phase.

For more information about how cryptographic hardware is supported by z/VSE, see *Security on IBM z/VSE*, SG24-7691, which is available at this website:

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

3.7.7 Configuring the AT-TLS server

The AT-TLS server BSTATLS can be started with JCL, as shown in Example 3-21.

Example 3-21 JCL for a BSTATLS server

```
* $$ JOB JNM=BSTATLS,CLASS=S,DISP=D
// JOB BSTATLS - AUTOMATIC TRANSPORT LAYER SECURITY
// OPTION PARTDUMP,NOSYS_DUMP,NOLOG
// OPTION SYSPARM='02'
// SETPARM IPTRACE='YYNNNNNN'
/* SETPARM IPTRACE='NNNNNNNN'
// SETPARM SSL$DBG='YES' OR 'NO' (DEFAULT = NO)
// SETPARM SSL$ICA='YES' OR 'NO' (DEFAULT = YES)
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD2.TCIPB,PRIMARY.JSCH,PRD2.SCEEBASE)
// EXEC BSTATLS,SIZE=BSTATLS,PARM='TRAP(OFF)/'
ID 02
*

KEYRING CRYPTO.KEYRING
KEYFILE SSL01
SECTYPE TLSV1
*

OPTION SERVER
ATTLS 23 AS 992 SSL
*

OPTION CLIENT
OPTION FTP
ATTLS 21 TO 9.152.222.71 AS 990 SSL
/*
// EXEC LISTLOG
/*
/&
* $$ EOJ
```

Note: The KEYRING parameter can be specified only once, assuming that customers want to have all .pem files at the same place. KEYFILE and SECTYPE can be specified for each connection separately.

3.7.8 Considerations on SSL performance

BSTATLS and BSTTPRXY provide a new performance-related parameter to use multiple subtasks for the SSL handshake process.

When you add

```
// SETPARM SUBTASK=nn
```

to your BSTATLS/BSTTPRXY JCL, multitasking is enabled for the gsk_secure_soc_init API function (refer to “gsk_secure_soc_init” on page 222). This provides better performance handling multiple concurrent SSL sockets.

You can specify up to 16 (nn=01, 02, ... 16) subtasks be used.

3.7.9 Considerations on blocking clear ports

In the previous sections of this chapter, we described how to proxy clear connections to secure connections and vice versa. The question remains how to effectively block the clear port against connections from outside.

Note: A first try might be to block the clear port by using the following PORT SUSPEND command:

```
MSG BSTTINET,D=PORT 21 SUSPEND
```

This command blocks port 21 until a PORT 21 RESUME is issued. While the port is suspended, no connections are accepted. However, the use of the PORT command also stops BSTTATLS from connecting on port 21 and this result is not what we want.

The BSTTSCTY.T FTP server security member provides IP address security. The use of the following FTP-IP/FTP-IP6 commands ALLOW access from the BSTTATLS or BSTTPRXY facility that uses the loopback address and DENY access for any other IP address:

```
*
FTP-IP  ALLOW 127.0.0.1      255.255.255.255
FTP-IP6 ALLOW ::1           /128
FTP-IP  DENY  0.0.0.0        0.0.0.0
FTP-IP6 DENY  ::0           ::0
*
```

Because the BSTTATLS and BSTTPRXY facilities can be set up to use the loopback address, this provides the access protection that you want. Complete the following steps:

1. Configure and start BSTTPRXY or BSTTATLS by using loopback.
2. Define FTP-IP/FTP-IP6 IP address security in BSTTSCTY.T.
3. Start the BSTTFTPS FTP server on port 21 (standard).

At this point, the only access to BSTTFTPS is through the loopback address on port 21. Local (or remote) users that attempt to connect directly to port 21 have their connection ended.

When a local (or remote) FTP client attempts to connect to the FTP server on port 21, they see the following messages:

```
jcb@z930-bt9300:~> ftp vse51b
Connected to vse51b.
421 Service not available, remote server has closed connection.
ftp> quit
jcb@z930-bt9300:~>
```

While on the z/VSE console, you see the following message:

```
S2 0491 BSTT053I USER DEFAULT LOGOFF RC=      4 03/26/2013-15:30:38-01EB
```

The USER DEFAULT indicates that the FTP client never logged on and RC=4 indicates a security issue.

You can still connect to the BSTTFTPS FTP server by using an SSL connection on port 990 (or whatever port you chose to use).

Note: Check for the latest IPv6/VSE version. It provides a firewall mechanism that allows enabling and blocking single ports. For example:

```
IN TCP DENY PORT 21
IN TCP DENY PORT 23
IN TCP ALLOW PORT 0
```

The following sections show how to set up different Telnet clients for SSL:

- ▶ wc3270 for Windows
- ▶ IBM Personal Communications

3.7.10 Configuring wc3270 for SSL

There are two versions of each of the Windows emulators, one with SSL support and one without. An option in the setup program defines which set is installed.

The secure versions of wc3270, s3270, and wpr3287 do not function without OpenSSL DLLs installed on your workstation. These DLLs are not part of the wc3270 installation; installing them is a separate process.

For more information, see the wc3270 documentation, which is available at this website:

<http://x3270.bgp.nu/documentation-ssl.html>

Adding your root certificate to wc3270

You must add your root certificate to the wc3270 root_certs.txt file in the wc3270 ApplicationData folder. On Windows XP, this folder is in the following directory:

C:\Documents and Settings\Administrator\Application Data\wc3270

Edit your .pem file and paste the ASCII text form of your root certificate into the root_certs.txt file, as shown in Figure 3-28.



Figure 3-28 Adding your root certificate to the .pem file

Creating a secure wc3270 session

In the wc3270 session wizard, choose SSL Tunnel = YES. Also, change the port number to the secure TN3270 port as defined in the BSTTPRXY JCL, as shown in Example 3-22.

Example 3-22 Configuring a secure wc3270 session

1. Host	: vsessl
2. Logical Unit Name	: (none)
3. TCP Port	: 992
4. Model Number	: 4 (43 rows x 80 columns)
5. Oversize	: (none)
6. Character Set	: bracket (CP 37*)
7. SSL Tunnel	: Yes
8. Proxy	: (none)
11. wpr3287 Printer Session	: No
15. Keymaps	: (none)
17. Font Size	: 12
18. Background Color	: black
19. Menu Bar	: Yes

Starting BSTTPRXY and BSTTVNET

You can now start BSTTPRXY and BSTTVNET. When SSL is used, BSTTPRXY or BSTTATLS must always be started before the application that they are servicing, as shown in Example 3-23.

Example 3-23 Starting BSTTPRXY

```
S1 0045 // JOB BSTTPRXY - BSI SSL PROXY SERVER
          DATE 05/03/2013, CLOCK 15/53/58
S1 0045 * // OPTION PARTDUMP,NOSYSDDUMP,NOLOG
S1 0045 BSTT000I INITIATED BSTTPXY Build253 07/05/12 16.30 EP=004201B0
S1 0045 BSTT003I COPYRIGHT (C) 1998-2012 BARNARD SOFTWARE, INC.
S1 0045 BSTT700I IPv6/VSE BUILD 253PRE11
S1 0045 BSTT719I      2:      1 Listening on port      992 SSL
```

After BSTTPRXY is started and listening on the secure port, start BSTTVNET, as shown in Example 3-24.

Example 3-24 Starting BSTTVNET

```
R1 0046 // JOB BSTTVNET - TN3270 WITH BSI STACK
          DATE 05/03/2013, CLOCK 15/58/21
R1 0046 BSTT000I INITIATED BSTTWAIT Ver 2.46 04/01/09 18.08 EP=00420078
R1 0046 BSTT003I COPYRIGHT (C) 1998-2012 BARNARD SOFTWARE, INC.
R1 0046 BSTT001I TERMINATED BSTTWAIT
R1 0046 BSTT000I INITIATED BSTTVNET Build248 02/15/10 16.24 EP=00420078
R1 0046 BSTT003I COPYRIGHT (C) 1998-2012 BARNARD SOFTWARE, INC.
R1 0046 BSTT004I CB=TTLA A=00469000 L=000013FC
R1 0046 BSTT019I VSE 0.10 MODE 31-BIT
R1 0046 BSTT004I CB=COMR A=002D0518 L=00000108
R1 0121 BSTT000I INITIATED BSTTXVNC Build249 04/29/10 08.27 EP=004FB000
R1 0121 BSTT671I BSTTVNET VTAM FEATURE IS AVAILABLE
R1 0121 BSTT701I TCP/IP-TOOLS BUILD 253
R1 0121 BSTT695I CONNECTING TO PORT 23 IP 9.152.131.189
```

R1 0113 BSTT000I INITIATED BSTTVSRV Buil250 12/14/10 17.49 EP=00569000
R1 0121 BSTT042I ATTACH OF BSTTVSRV COMPLETED

Connecting by using SSL

Double-clicking the SSL session icon connects to BSTTPRXY by using SSL, as shown in Figure 3-29.

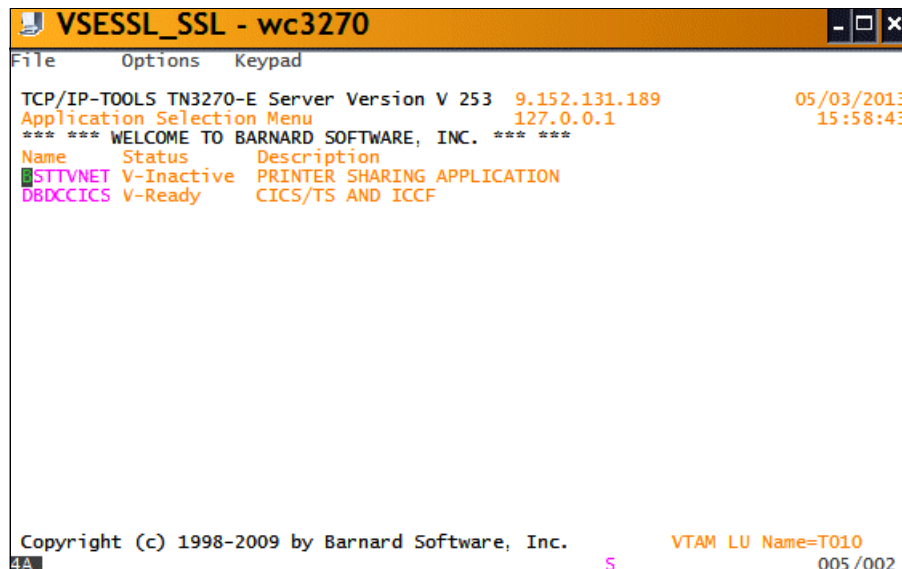


Figure 3-29 VTAM selection panel in wc3270 that uses SSL

The green S at the bottom of the wc3270 window indicates that this is a secure session. You might also check the SSL session properties (as shown in Example 3-25) by clicking **File** → **Status**.

Example 3-25 wc3270 session properties

[wc3270]

```
wc3270 v3.3.12ga7 Wed Aug 24 09:36:34 CDT 2011 pdm
Model 3279-4-E: 80 columns x 43 rows, color, extended data stream
Terminal name: IBM-3279-4-E
EBCDIC character set: bracket (SBCS)
Host code page: 37+
Host SBCS CGCSGID: GCSGID 697, CPGID 37
Windows code page: OEM 1252 ANSI 1252
Connected to: vsessl
  Port: 992
  using TLS/SSL
  3270 mode, 7 minutes 35 seconds
Sent 58 bytes, 0 records
Received 2007 bytes, 1 record
Press <Enter> to resume session.
wc3270>
```

3.7.11 Configuring IBM Personal Communications for SSL

For more information about configuring IBM Personal Communications for SSL with the IP stack from Connectivity Systems, Inc, see the following resources:

- ▶ White paper that is available at this website:

<http://www.ibm.com/servers/eserver/zseries/zvse/documentation/documents.html>

- ▶ *Security on IBM z/VSE*, SG24-7691, which is available at this website:

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

The following sections are based on PCOM 5.9/6.0 for Windows. There are two major changes with PCOM 5.9 and later:

- ▶ The Microsoft Windows CryptoAPI (MSCAPI) can now be used. This allows importing your certificate (or certificates) into the Windows certificate store (click **Control Panel** → **Internet Options**).
- ▶ PCOM 5.9 and later now supports the so-called “Telnet-negotiated security”, which is not supported by the TLS.D.

Adding your root certificate to the MSCAPI on Windows

Using the Microsoft Windows CryptoAPI is an alternative method to the use of the PCOM key database.

Note: A previously created PCOM key database can be migrated to MSCAPI by using the Certificate Migration utility. This utility can be accessed by clicking **Start** → **All Programs** → **IBM Personal Communications** → **Administrative and PD Aids**.

Adding your root certificate to PCOM

PCOM wants the certificates in PFX form, so you must convert your previously created .pem file into the PFX format. This can be done, for example with OpenSSL on Windows, as shown in the following example:

```
openssl pkcs12 -export -out ssl01.pfx -in ssl01.pem
```

During this process, you are prompted to specify an export password. This password also must be entered later when the PFX is added to the PCOM key database.

For more information about this process, see *Security on IBM z/VSE*, SG24-7691, which is available at this website:

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

Creating a secure PCOM session

On the Host Definition tab, specify the secure Telnet port, as shown in Figure 3-30.

	Host Name or IP Address	LU or Pool Name	Port Number
Primary	vsessl		992
Backup 1			23
Backup 2			23

Connection Options

Connection Timeout: 6 Seconds

☒ Auto-reconnect

☐ Try connecting to last configured host infinitely

Keep Alive

☐ Enable Telnet Keep Alive

Keep Alive Time Out: 180 Seconds

Buttons: OK, Cancel, Apply, Help

Figure 3-30 Telnet3270 window in IBM Personal Communications

Note: As shown in Figure 3-31, there is a new option in the Security Setup window that is call **Telnet-negotiated**. Do not select this option because BSTTATLS/BSTTPRXY do not support Telnet-negotiated security.

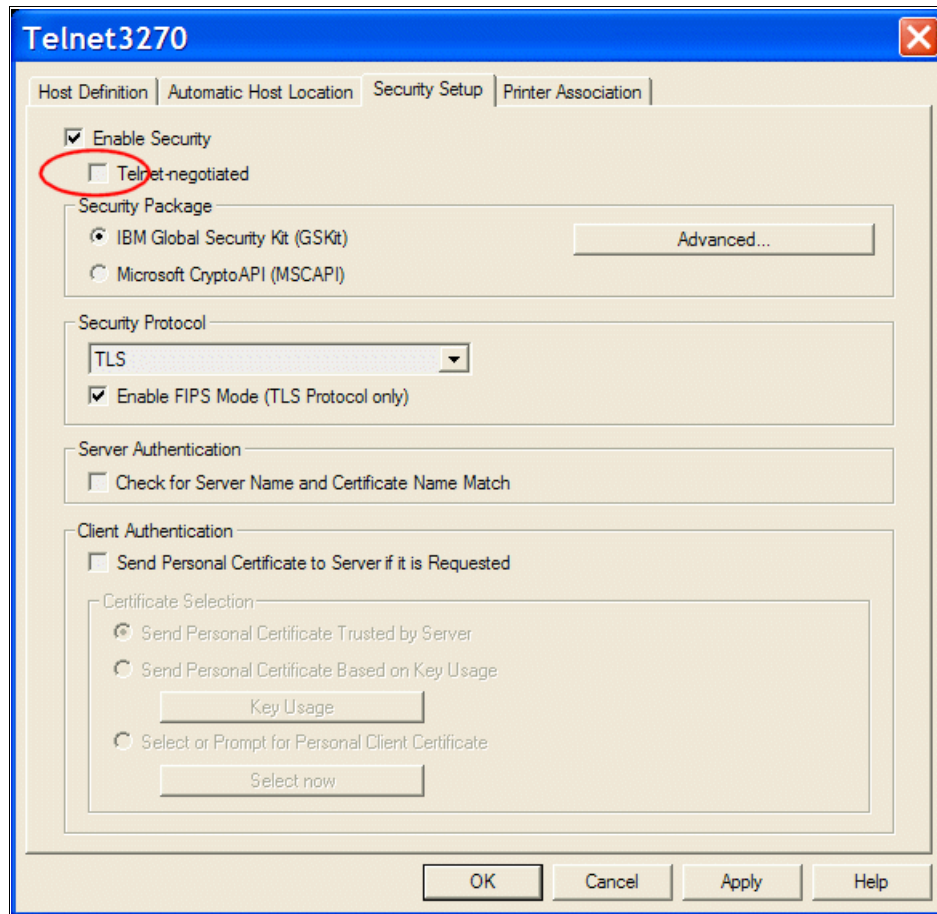


Figure 3-31 Telnet-negotiated option in IBM Personal Communications

Depending on whether you imported your certificates into the PCOM key database or the Windows certificate store, select the related Security Package.

Connecting using SSL

You can directly connect to the BSTTATLS or BSTTPRXY application on VSE, as shown in Figure 3-32.

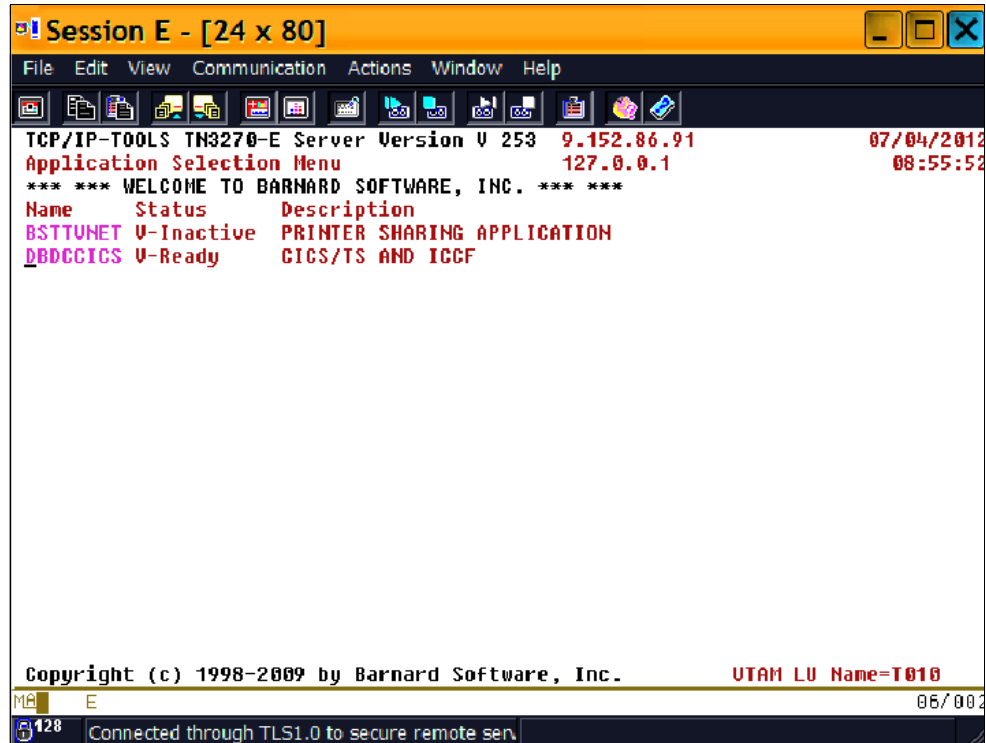


Figure 3-32 Secure PCOM session

The lower left corner of the PCOM window shows the encryption grade. In this example, the connection was established by using AES-128.

3.7.12 Configuring secure FTP

In this topic, we describe the setup of secure FTP by using the BSTTATLS server. We also provide an overview of different secure FTP connection types. These types are relevant when FTP clients are configured for use with BSTTATLS.

File transfer connection types overview

Secured FTP connections are one important part of every company network. There are several kinds of secure FTP connection types, but only a subset is supported on z/VSE. Unfortunately, their naming is not consistent through various FTP products.

In general, all terms that are used by the different products can be mapped to the following four types of secured FTP connections:

- FTPS Explicit

An extension to the FTP standard that allows clients to request that the FTP session is encrypted. The login and the data are encrypted. Explicit FTPS currently is not supported on z/VSE.

► **FTPS Implicit**

A standard for FTP that requires the use of an SSL or TLS connection. It was specified to use different ports than FTP. This is the supported connection type by TCP/IP for VSE/ESA and IPv6/VSE. As with Explicit, the login and the data are encrypted.

► **SSH File Transfer Protocol (SFTP)**

This type uses a different protocol. Standard FTP clients cannot be used to talk to an SFTP server, nor can one connect to an FTP server with a client that supports only SFTP. SFTP currently is not supported on z/VSE.

► **FTP over SSH**

This type refers to the practice of tunneling a normal FTP session over an SSH connection. The login is encrypted, but data is not. This connection type also is not supported on z/VSE.

Table 3-2 shows some examples of how these connection types are named by some FTP clients that we used in our tests.

Table 3-2 Examples of secure FTP connection types

Connection type	FileZilla	Core FTP Language Environment®	Total Commander
FTPS explicit	Explicit FTP over TLS	AUTH_SSL AUTH_TLS	N/A
FTPS implicit	Implicit FTP over TLS	FTPS (SSL DIRECT)	SSL/TLS

VSE as a server

The JCL that is shown in Example 3-26 starts an FTP server on VSE that is listening on port 21.

Example 3-26 JCL to start an FTP server

```
* $$ JOB JNM=FTPDIPV6,CLASS=S,DISP=D
// JOB FTPDIPV6 - FTP SERVER FOR BSI STACK
// LIBDEF PHASE,SEARCH=(PRD2.CONFIG,PRD2.TCPIPB)
// LIBDEF SOURCE,SEARCH=(PRD2.TCPIPB)
// EXEC BSTTFTPS,SIZE=BSTTFTPS,OS390
ID 02
*
OPEN 9.152.86.91 21
*
* USE THE IBM PROVIDED BSM SECURITY EXIT
BSSTISX
*
* DEFINE THE DEFAULT FILE SYSTEM TO USE
SMNT POWER
*
ATTACH SERVER-1
ATTACH SERVER-2
ATTACH SERVER-3
*
/*
/&
* $$ E0J
```


BSTTATLS now listens on the default secure FTP port 990 with the following AT-TLS definitions:

```
KEYFILE SSL01
SECTYPE TLSV1
OPTION SERVER
OPTION FTP
ATTLS 21 AS 990 SSL
```

The following topics provide examples of how to use different FTP clients for connecting to BSTTPRXY/BSTTATLS. In our tests, we used some popular freeware clients, such as FileZilla, Core FTP Language Environment, and Total Commander.

Note: The FTP clients do not need a client keyring file if you do not use SSL client authentication. When you are connecting the first time, a client usually displays a window in which the user must accept or reject the SSL server certificate. Some clients allow importing the SSL server certificate into their client keystore, which makes this check obsolete when you are reconnecting.

The following sections show how to set up different FTP clients for SSL:

- ▶ FileZilla client
- ▶ Core FTP Language Environment
- ▶ Total Commander

Using FileZilla client

When FileZilla client is used, you configure a secure connection by using the Site Manager. On the FileZilla client main window, click the **Site Manager** toolbar button, as shown in Figure 3-33.

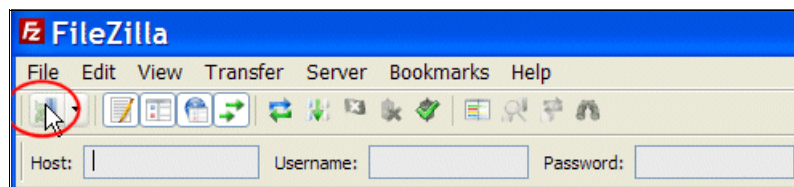


Figure 3-33 FileZilla client main window

For more information about FileZilla, see this website:

<http://filezilla-project.org/download.php>

In the Site Manager window (see Figure 3-34), configure a new site. For Encryption, you must select **Require implicit FTP over TLS**. For the user and password, enter a valid VSE user ID and its password. Port 990 is the default value for the secure FTP port and does not need to be specified.

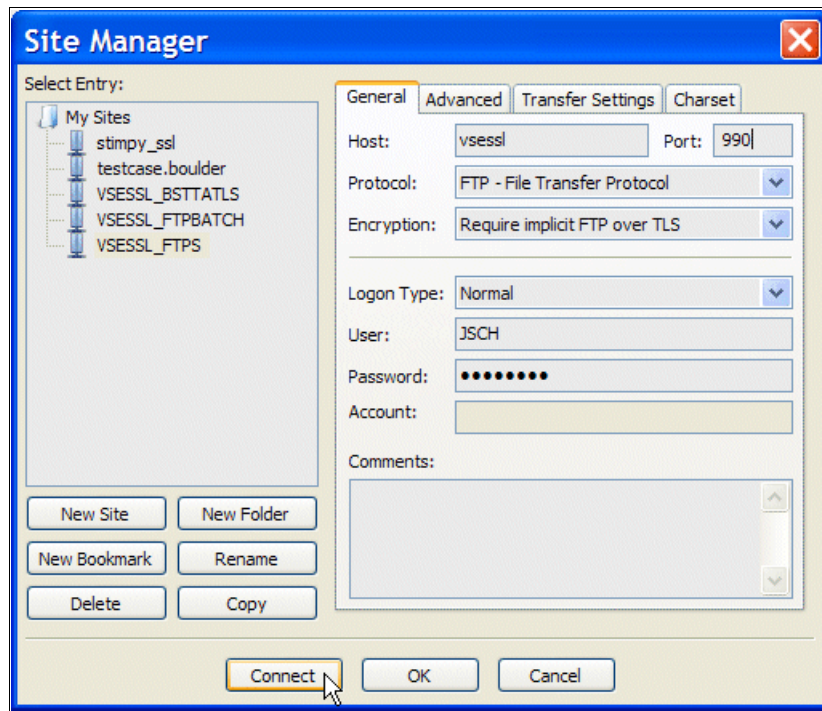


Figure 3-34 FileZilla client Site Manager window

If you are using a firewall, you should switch to passive mode. Select **Transfer Settings** and then click **Passive**, as shown in Figure 3-35.

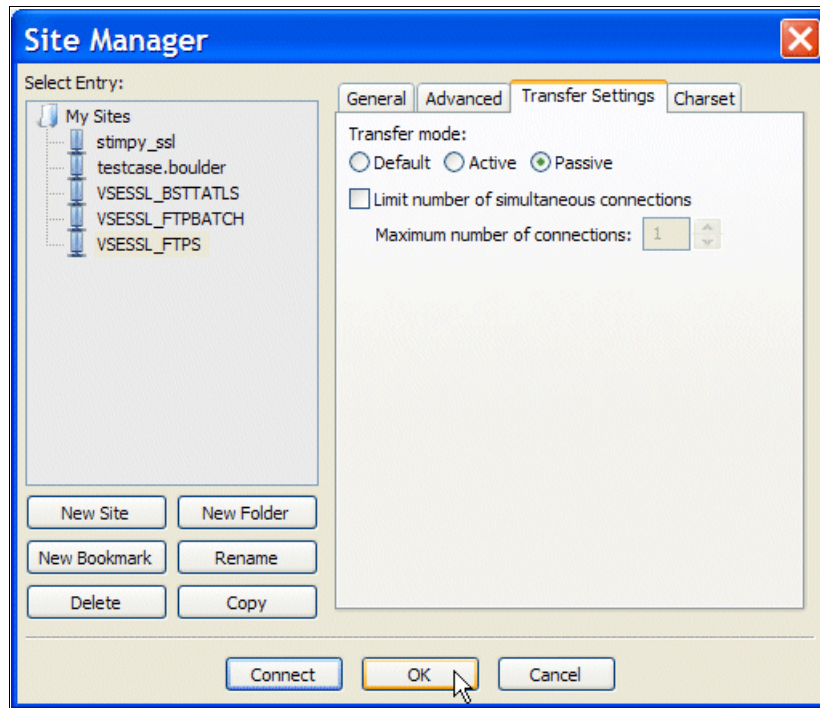


Figure 3-35 Selecting passive mode in FileZilla client.

In the next section, we describe how different FTP clients sometimes require different SSL settings.

Using Core FTP Language Environment

The freeware FTP client Core FTP Language Environment has a similar concept of defining connections as does FileZilla.

For more information about Core FTP Language Environment, see this website:

<http://www.coreftp.com/>

Open the Site Manager and specify the credentials for your VSE system. For Connection, you must select **FTPS (SSL DIRECT)**, as shown in Figure 3-36.

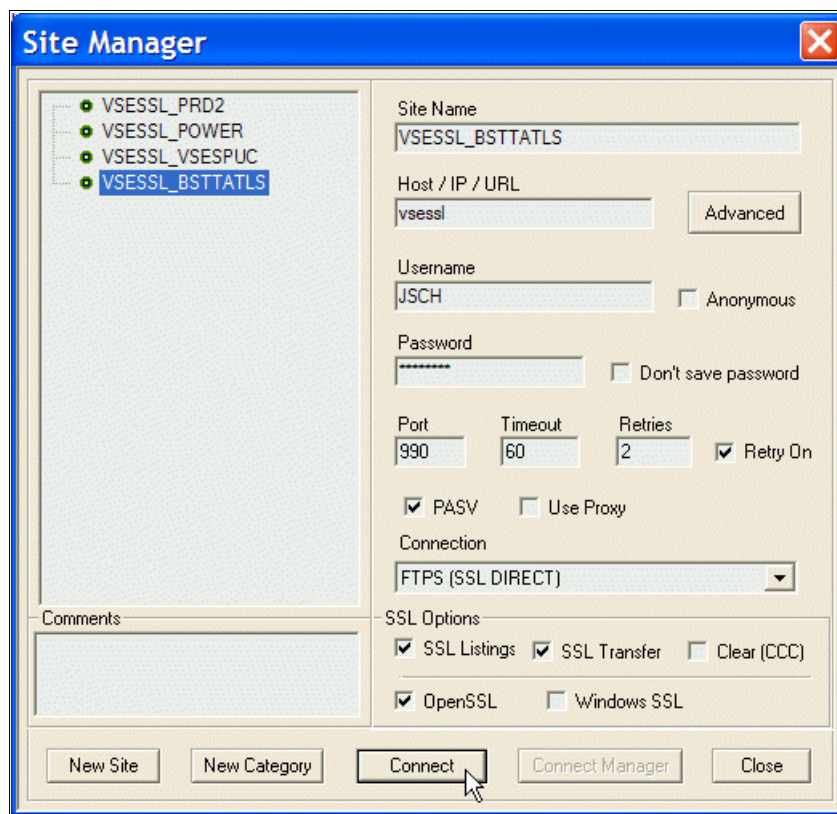


Figure 3-36 CoreFTP Language Environment Site Manager window

You can now connect immediately to VSE.

Note: For BSTTATLS/BSTTPRXY, you must select **FTPS (SSL DIRECT)**. In contrast, when TCP/IP for VSE is used, you must use AUTH SSL or AUTH TLS.

Using Total Commander

Total Commander is a Shareware file manager for Windows. You can download a test version from this website:

<http://www.ghisler.com/>

For SSL support, Total Commander must have access to the OpenSSL DLLs on Windows. In our tests, we copied the following three DLLs into the Total Commander installation directory:

- ▶ LIBEAY32.DLL
- ▶ LIBSSL32.DLL
- ▶ SSLEAY32.DLL

By using Total Commander, you can configure SSL by clicking **Net** → **FTP Connect**. In the Connect to ftp server window, create a connection to your host, as shown in Figure 3-37.

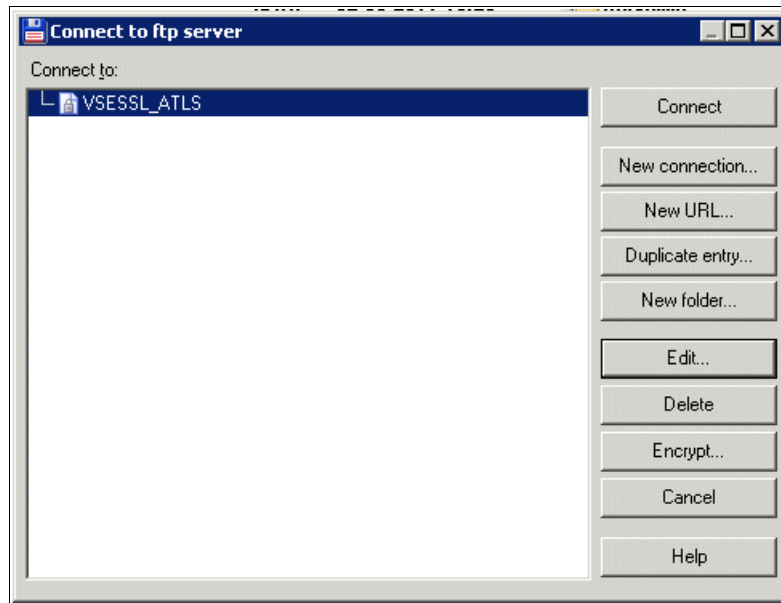


Figure 3-37 Total Commander Connect to ftp server window

In the FTP: connection details window, select **SSL/TLS**. Also, enter `ftps://VSESSL:990` in the **Host name** field, as shown in Figure 3-38.

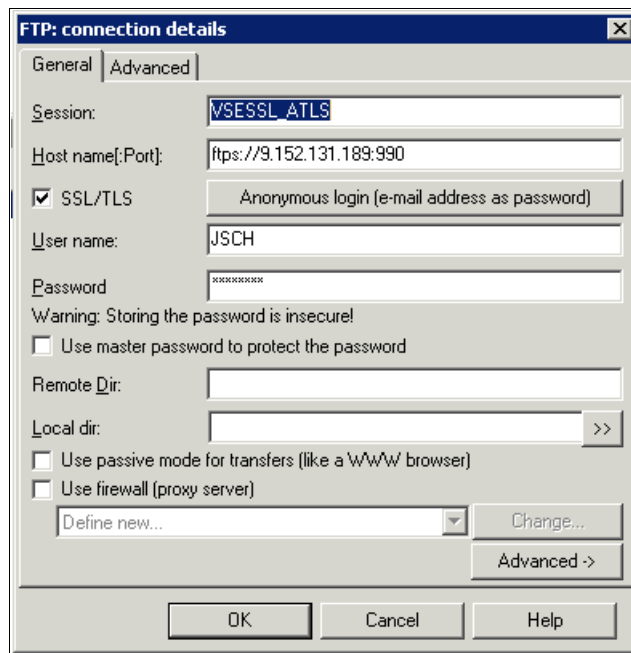


Figure 3-38 Total Commander FTP: connection details window

There are no other options to distinguish between the different SSL connection types.

In our tests, it was important not to use passive mode. However, this might be different in your network.

The next section describes examples when VSE acts as the FTP client.

VSE as a client

The BSTTFTPC utility can be used with BSTTATLS and BSTTPRXY for connecting to a remote FTP server by using SSL. However, you cannot use the same JCL because BSTTATLS and BSTTPRXY work differently. BSTTATLS intercepts outbound connections and converts them to SSL. BSTTPRXY creates a proxy connection to the remote FTP server address with the SSL port.

These examples are described next.

Using BSTTATLS

Example 3-27 shows an example for an FTP client that is connecting to a remote FTP server address. This connection is intercepted by BSTTATLS and converted to SSL.

Note: The user ID and password might be case-sensitive, depending on which FTP server is used. PBSZ 0 and PROT P are necessary to encrypt any transferred data.

Example 3-27 JCL for BSTTFTPC using BSTTATLS for SSL

```
// EXEC BSTTFTPC,SIZE=BSTTFTPC,OS390
ID 02
OPEN 9.152.222.71
PBSZ 0
PROT P
USER myuser
PASS mypasswd
QUIT
/*
/ &
```

With the following AT-TLS definitions, BSTTFTPC connects to the secure port 990 of the FTP server with the IP address, as shown in Example 3-28.

Example 3-28 BSTTATLS JCL

```
// EXEC BSTTATLS,SIZE=BSTTATLS,PARM='TRAP(OFF)/'
ID 02
KEYRING PRD2.CONFIG
KEYFILE SSL01
SECTYPE TLSV1
*
OPTION CLIENT
HANDSHAKE_CLIENT 0
OPTION FTP
ATTLS 21 TO 9.152.222.71 AS 990 SSL
*
/*
/ &
```

The HANDSHAKE_CLIENT parameter controls whether the server certificate is verified by the client. Specifying 0 causes the server certificate to be verified by using the public key in the keyring file. This means that the CA root certificate must be in the KEYFILE. The default value is 3, which means that no checking is made.

Using BSTTPRXY

Example 3-29 shows an example for BSTTFTPC that uses BSTTPRXY for SSL. BSTTFTPC connects to port 21 on localhost, where BSTTPRXY creates a proxy connection to the remote FTP server.

Note: You can run BSTTPRXY on another z/VSE system and specify this system's IP address instead of localhost.

Example 3-29 JCL for BSTTFTPC that uses BSTTPRXY for SSL

```
// EXEC BSTTFTPC,SIZE=BSTTFTPC,OS390,TASKS=ANY
ID 02
OPEN 127.0.0.1
PBSZ 0
PROT P
USER myuser
PASS mypasswd
QUIT
/*
/ &
```

With the following BSTTPRXY definitions, BSTTFTPC connects to the remote FTP server by using SSL, as shown in Example 3-30.

Example 3-30 BSTTPRXY JCL

```
// EXEC BSTTPRXY,SIZE=BSTTPRXY,PARM='TRAP(OFF)/'
ID 02
*
KEYRING PRD2.CONFIG
KEYFILE SSL01
SECTYPE TLSV1
*
OPTION CLIENT
HANDSHAKE_CLIENT 0
OPTION FTP
PROXY TCP V4 21 TXT * TO V4 990 SSL * 9.152.222.71
```

The following examples show FTP servers that are running on MS Windows and Linux platforms.

Using FileZilla server

When the FileZilla FTP server is used, complete the following steps to configure SSL:

1. In the FileZilla Server interface window, select **Edit** → **Settings**, as shown in Figure 3-39.

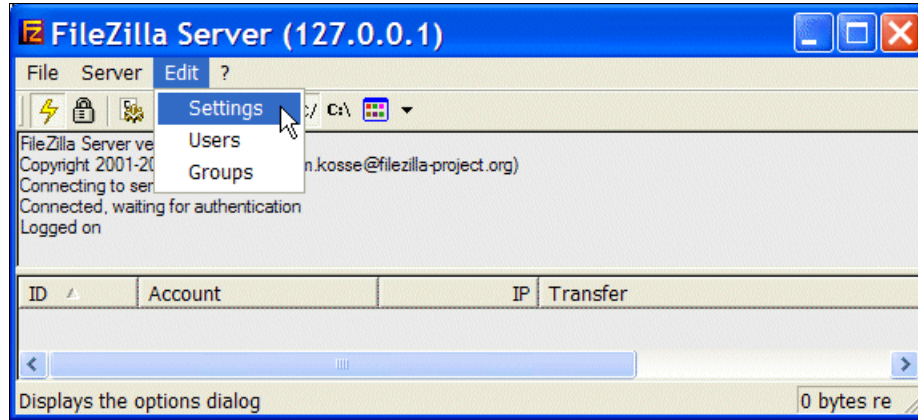


Figure 3-39 FileZilla Server interface window

2. In the FileZilla Server Options window, select **SSL/TLS settings**, as shown in Figure 3-40.

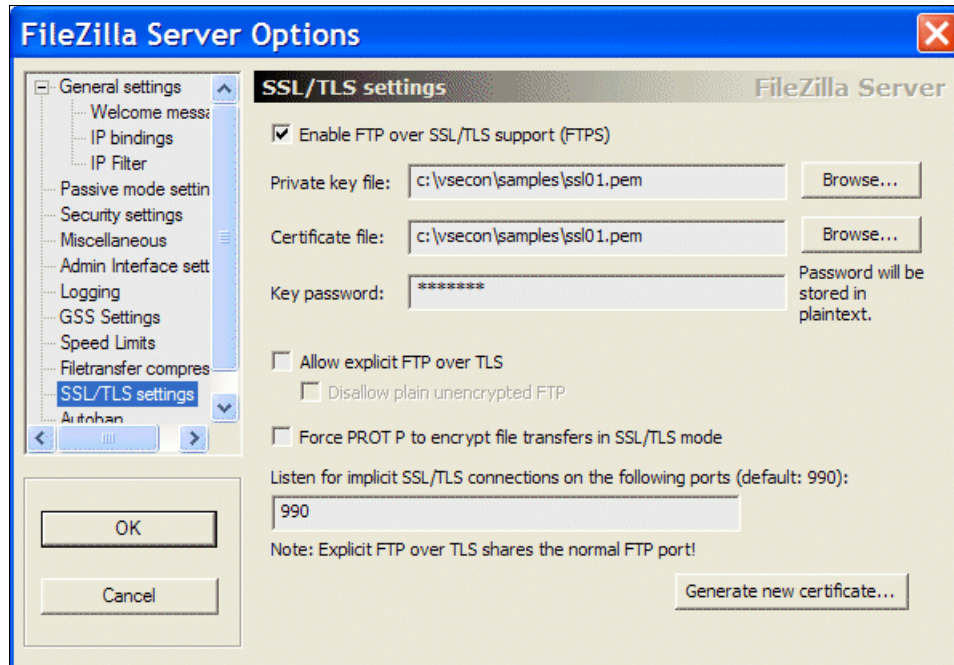


Figure 3-40 FileZilla server options window

3. Select the **Enable FTP over SSL/TLS support** option and specify your .pem file in the fields for Private key file and Certificate file.

Also, ensure that the SSL/TLS listening port matches the number that is specified in the BSTATLS JCL.

Note: By clicking **Generate new certificate**, you can create a .pem file with a private key and a second .pem file with a self-signed certificate. You also can create a .pem file with both items. However, this .pem file is not usable because the server certificate is missing. You must complete the steps that are described in 3.7.2, “Creating the keystore” on page 71 to create a working .pem file.

When you are installing FileZilla server on Windows 7, the server interface usually starts automatically when Windows is started. However, while the server interface can be configured for manual startup by using the Windows Services on Windows XP, this configuration is not possible on Windows 7. Instead, you must open the registry editor and browse to the following key:

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run

Then, you delete the FileZilla Server Interface entry.

Using VSFTPD server

VSFTPD is a GPL licensed FTP server for UNIX like systems, including Linux. It is reported to be secure and fast.

In our tests, we used VSFTPD 3.0.2. Example 3-31 shows the relevant parameters from the vsftpd.conf file.

Example 3-31 VSFTPD configuration file

```
# Limit passive ports to this range to assist firewalling
pasv_min_port=30000
pasv_max_port=30100
#
anon_mkdir_write_enable=NO
anon_root=/srv/ftp
anon_upload_enable=YES
chroot_local_user=NO
ftpd_banner=Welcome message
idle_session_timeout=900
local_enable=YES
log_ftp_protocol=NO
max_clients=10
max_per_ip=3
pasv_enable=YES
ssl_sslv2=NO
ssl_sslv3=YES
ssl_tlsv1=YES
write_enable=YES
ssl_request_cert=NO
rsa_cert_file=/etc/vsftpd/FILEZ01.PEM
listen_port=990
implicit_ssl=YES
ssl_ciphers=AES256-SHA256
```

Parameter `ssl_ciphers` specifies the SSL cipher suites to be used. Because VSFTPD uses OpenSSL internally, the parameter values are the same as those values that are used by OpenSSL.

For more information about how to use VSFTPD in a z/VSE environment, see *Security on IBM z/VSE*, SG24-7691.

3.7.13 Configuring VSE Connector Server

The VSE Connector Server can use OpenSSL with the BSTTPRXY or BSTTATLS utilities that are provided by BSI.

In this topic, we describe the scenario in which BSTTATLS is used to provide SSL for the VSE Connector Server, as shown in Figure 3-41. BSTTPRXY also can be used in the same way.

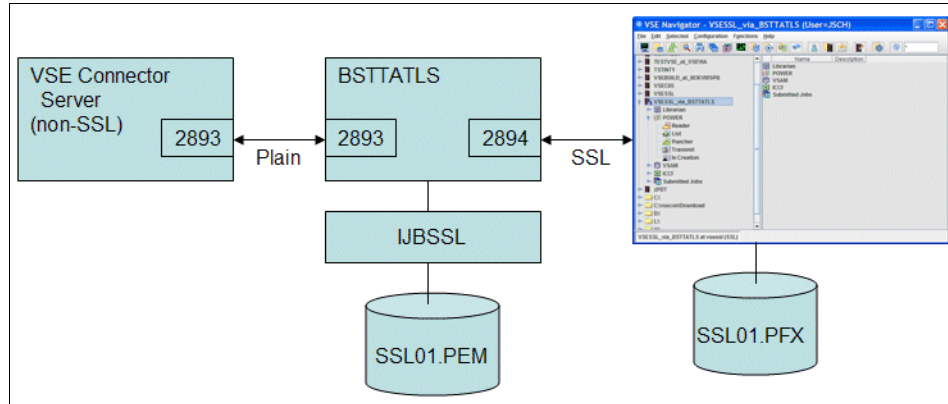


Figure 3-41 Overview of connectivity that uses BSTTATLS

In this case, the VSE Connector Server runs in non-SSL mode. SSL configuration is only required for BSTTATLS and any VSE Connector Client.

Note: The VSE Connector Server partition must have a lower priority than the BSTTATLS partition. Otherwise, your system might hang.

Setting up BSTTATLS

In Example 3-32, the VSE Connector Server listens on its default port 2893. BSTTATLS listens for secure connections on port 2894.

Example 3-32 BSTTATLS setup for VSE Connector Server

```
// EXEC BSTTATLS,SIZE=BSTTATLS
ID 02
*
KEYRING PRD2.CONFIG
KEYFILE SSL01
SECTYPE TLSV1
*
OPTION SERVER
ATTLS 2893 AS 2894 SSL
*
/*
/ &
```

After changing the JCL, start the BSTTATLS application.

Setting Up BSTTPRXY

The JCL that is shown in Example 3-33 shows the same setup that uses BSTTPRXY instead of BSTTATLS.

Example 3-33 BSTTPRXY setup for VSE Connector Server

```
// EXEC BSTTPRXY,SIZE=BSTTPRXY
ID 02
*
KEYRING PRD2.CONFIG
KEYFILE SSL01
SECTYPE TLSV1
*
OPTION SERVER
PROXY TCP V4 2894 SSL * TO V4 2893 TXT * 127.0.0.1
*
/*
/ &
```

Note: BSTTPRXY can proxy one connection only.

Setting up VSE Connector Server

As described in the previous section, the VSE Connector Server runs in non-SSL mode. Therefore, there is no SSL configuration necessary for the VSE Connector Server.

Setting up keyring file

If you did not save the keystore in .pfx format when you were creating the .pem file, the following methods can be used to copy your certificates from the .pem file to a PFX or JKS keyring file that can be used by the VSE Connector Client:

- Edit or browse the .pem file directly on VSE with DITTO (see Figure 3-42 on page 104), VSE Navigator, or VSE Librarian and paste the certificates one after the other into Keyman/VSE. Select and copy each certificate, including the BEGIN CERTIFICATE and END CERTIFICATE lines, as shown in Figure 3-42 on page 104.

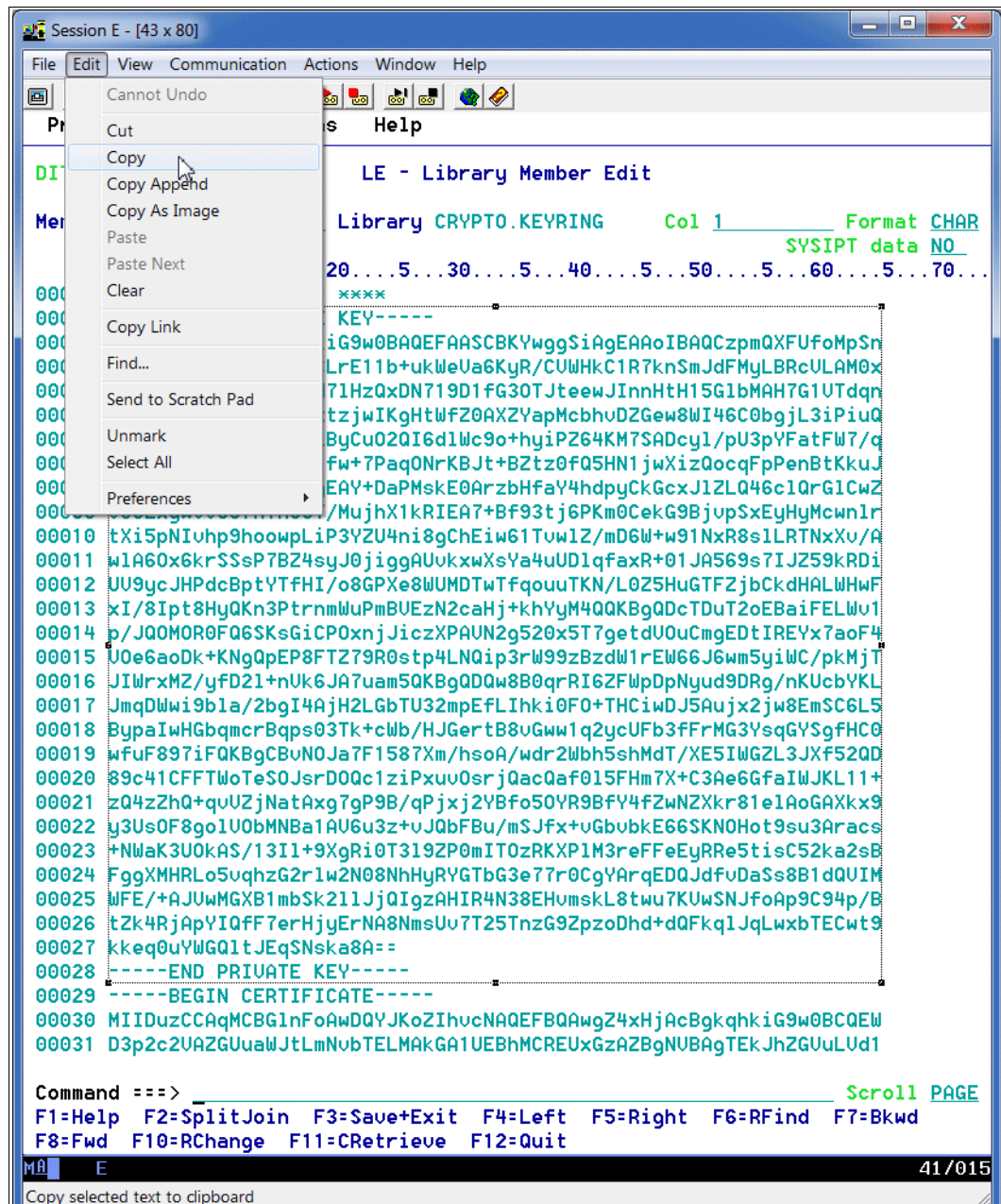


Figure 3-42 z/VSE DITTO display of a .pem file in VSAM

- ▶ You can use the same method by using a local copy of the .pem file.
- ▶ Use Openssl-Light on Windows to convert the .pem file into a PFX. For more information, see 5.4, “Keystore considerations” on page 158.

You do not need the private key on the client side. In Keyman/VSE, click **File** → **Read clipboard**, as shown in Figure 3-43.

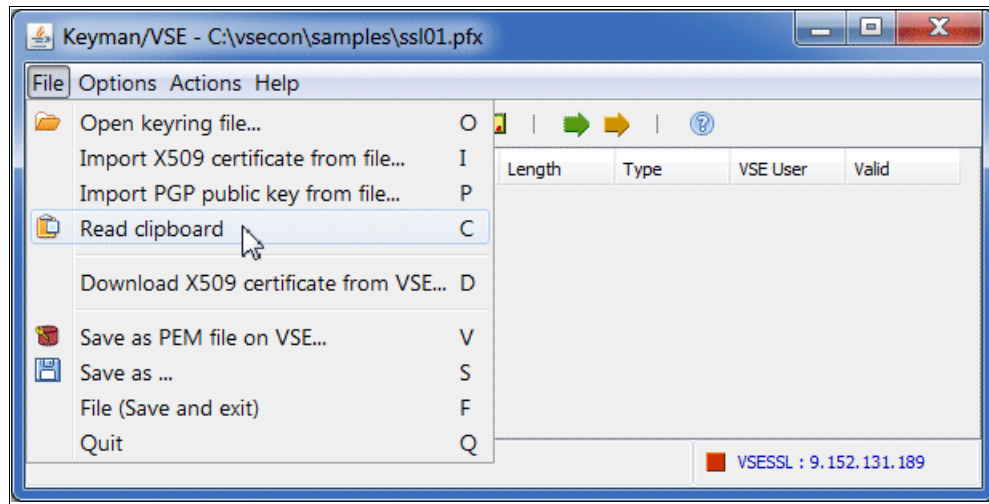


Figure 3-43 Keyman/VSE read clipboard function

After the two certificate.s are copied, save the two items into a .pfx or .jks file, as shown in Figure 3-44

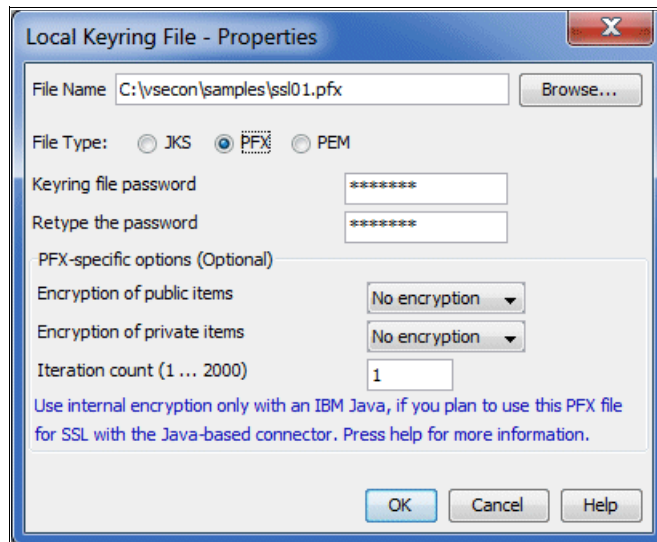


Figure 3-44 Keyman/VSE local keyring file properties window

Click **OK** and save the keyring file.

Setting up VSE Navigator

Modify your existing VSE definition or define a new VSE host with the IP address of your VSE system. Specify the following port number where BSTTATLS forwards the SSL traffic to (see 3.7.6, “Configuring the SSL proxy server” on page 82):

ATTLS 2893 AS 2894 SSL

As shown in Figure 3-45, the VSE Navigator connects to port 2894 on VSE by using SSL. BSTATLS forwards the traffic to port 2893 where the VSE Connector Server listens for non-SSL connections.

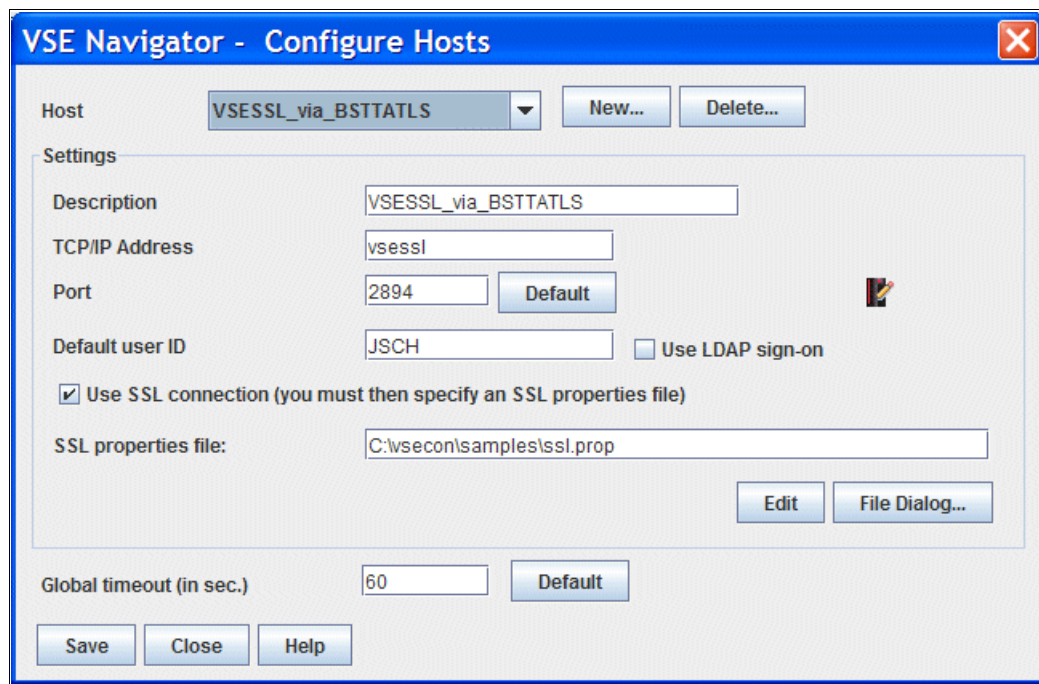


Figure 3-45 VSE Navigator Configure Hosts window

Now start the BSTATLS server.

Click **Edit** to edit the Navigator's SSL properties file and change the SSL parameters as shown in the following example:

```
KEYRINGFILE=c:\\vsecon\\samples\\ss101.pfx
KEYRINGPWD=myspasswd
SSLVERSION=SSL
CIPHERSUITES=TLS_RSA_WITH_AES_128_CBC_SHA,SSL_RSA_WITH_3DES_EDE_CBC_SHA
```

Now start the VSE Connector Server.

Connect to BSTTATLS

You can now directly connect to the VSE Connector Server through BSTTATLS, as shown in Figure 3-46.

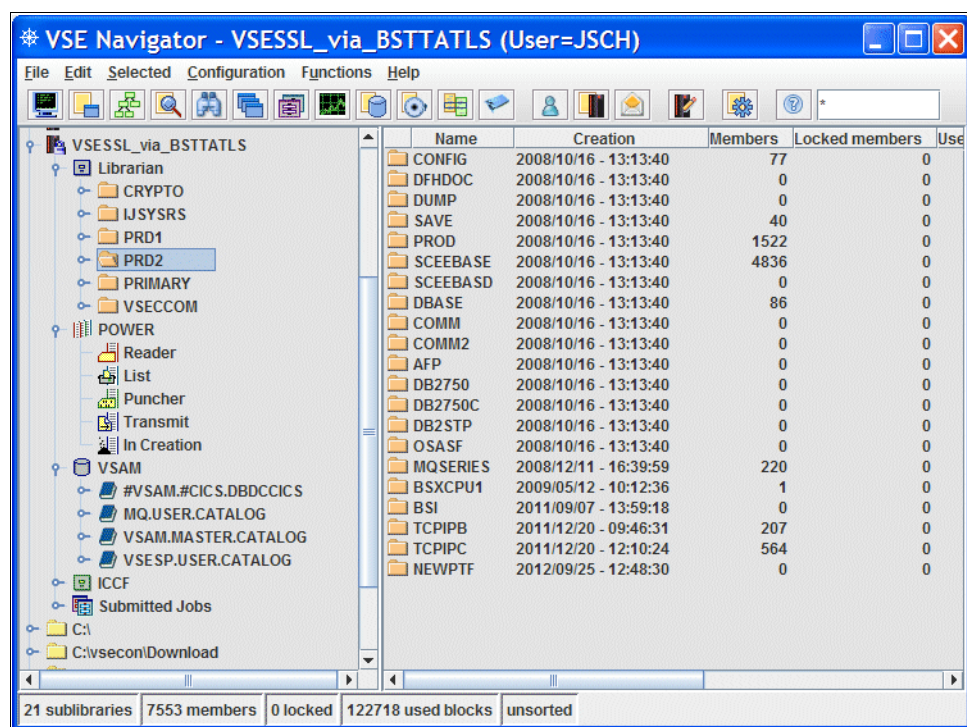


Figure 3-46 VSE Navigator main window that is connected to BSTTATLS

3.7.14 SSL client authentication

SSL client authentication is a server-side feature that requires an SSL client certificate to be sent from the client to the server during the SSL handshake process. When a server is configured for SSL client authentication, any client must have a valid client certificate.

The client certificate contains information about the user and a unique private-public key pair. It is signed by the same CA root certificate as the SSL server certificate that the server sends to the client. Although SSL client authentication is more secure than server authentication, more configuration is necessary for SSL clients.

Configuring BSTTATLS for client authentication

BSTTPRXY and BSTTATLS provide the following parameters to enable and control SSL client authentication:

```
HANDSHAKE_SERVER [ 1 | 2 ]
CA_AUTH          [ YES | NO ]
CA_AUTH_TYPE     [ 0 | 1 | 2 | 3 ]
```

HANDSHAKE_SERVER enables client authentication when the value is set to 2. CA_AUTH and CA_AUTH_TYPE control how the client certificate is verified by the server.

For more information about these parameters, see *IPv6/VSE SSL Installation, Programming, and User's Guide*, SC34-2676.

In our test, we used a BSTTATLS job, as shown in Example 3-34.

Example 3-34 Using BSTTATLS with SSL client authentication

```

* $$ JOB JNM=BSTTATLS,CLASS=S,DISP=D
// JOB BSTTATLS - AUTOMATIC TRANSPORT LAYER SECURITY
// OPTION PARTDUMP,NOSYSDDUMP,NOLOG
// OPTION SYSPARM='02'
// SETPARM IPTRACE='YYNNNNNN'
// UPSI 011
// SETPARM SSL$DBG='YES' OR 'NO' (DEFAULT = NO)
// SETPARM SSL$ICA='YES' OR 'NO' (DEFAULT = YES)
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD2.PROD,PRD2.TCPIPB,PRD2.SCEEBASE)
// EXEC BSTTATLS,SIZE=BSTTATLS,PARM='TRAP(OFF)/'
ID 02
*

KEYRING PRD2.CONFIG
KEYFILE SSL01
SECTYPE TSLV1
*

OPTION SERVER
HANDSHAKE_SERVER 2
CA_AUTH YES
CA_AUTH_TYPE 0
ATTLS 2893 AS 2894 SSL
*

/*
// EXEC LISTLOG
/*
/&
* $$ EOJ

```

With SSL\$DBG='YES', the trace output shows the OpenSSL and gsk-interface trace.

Configuring VSE Navigator for client authentication

Open your previously created .pfx file with Keyman/VSE. Select the root certificate and click **Generate user certificate**, as shown in Figure 3-47.

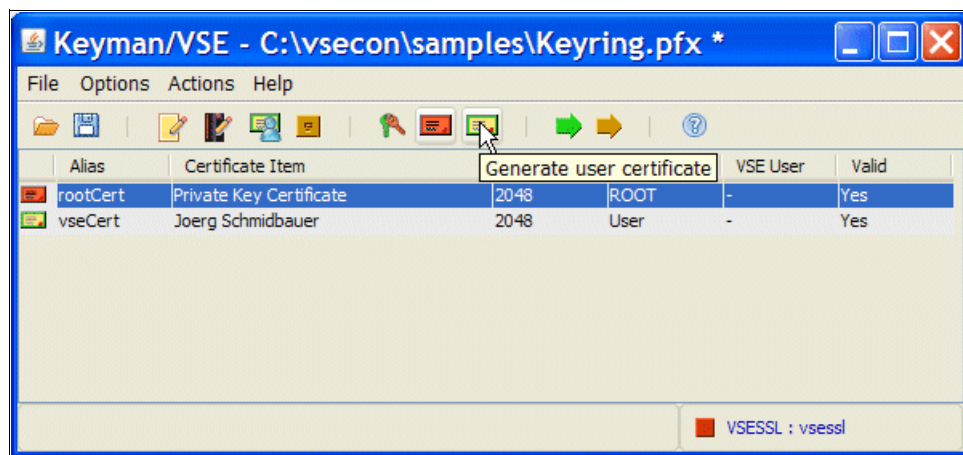


Figure 3-47 Keyman/VSE, generate user certificate

A certificate is created that is signed by your root certificate. This is the criteria that must be fulfilled for an SSL client certificate; it has its own key but is signed by the root certificate that also was used when the SSL server certificate was signed, as shown in Figure 3-48.

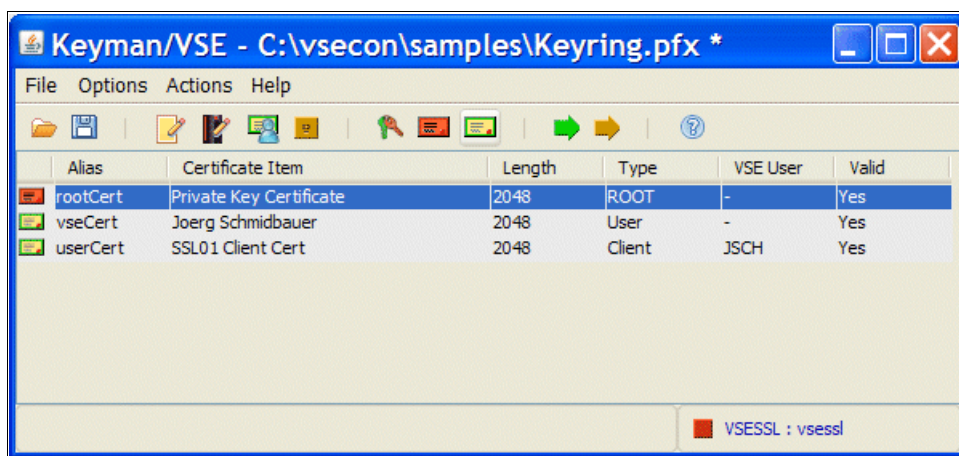


Figure 3-48 Keyman/VSE main window

Save the updated keyring file. You are now ready to connect to BSTTATLS by using SSL client authentication.

Connecting to BSTTATLS

In VSE Navigator, you cannot see whether you are connected through SSL server or SSL client authentication. However, the BSTTATLS trace shows some internal details about the SSL connection. In the output, search for `gsk_secure_soc_init`. Example 3-35 shows some parts of the output.

Example 3-35 Debug output from BSTTATLS

```
...
-> ijbslgsk.c, gsk_secure_soc_init ...
...
*** Info: Now entering SSL handshake ...
*** Info: we are running as server with client auth. A valid client certificate is
required.
...
-----
We are server with client auth ...
Client certificate info:
  cert_body_len      = 0
  current session ID =
[8855B332D9F4333B888BC1C60AD5B4C49CAE4F88CEB1E691BFA4D60AB04626E4]
  new session?      = 1
  serial number     = [63526DF6]
Subject:
  common name       = [SSL01 Client Cert]
  locality          = [Boeblingen]
  state or province = [Baden-Wuerttemberg]
  country           = [DE]
  organization      = [IBM]
  org unit          = [Development]
Issuer:
  common name       = [Private Key Certificate]
```

```

locality          = [Boeblingen]
state or province = [Baden-Wuerttemberg]
country           = [DE]
organization      = [IBM]
org unit          = [IBM Germany]

```

Summary:

```

fd                = 0x00000005
hs_type           = 2 (GSK_AS_SERVER_WITH_CLIENT_AUTH)
DName             = [SSL01]
keyring type      = LIBR
keyring           = DD:PRD2.CONFIG(SSL01.PEM)
sec_type          = [TLSV1]
cipher_specs (V2) = []
v3cipher_specs    = [05043538392F32330A161309151203060201]
skread            = 0x00512400
skwrite           = 0x0051C308
cipherSelected (V2) = []
v3cipherSelected  = [2F]
size of public key = 256 Bytes (2048 bits)
failureReasonCode = 0
gsk_data          = 0x0047C9B0

```

```

<- ijbslgsk.c, gsk_secure_soc_init, ok
...

```

Client authentication with FTP

We could not set up SSL client authentication with the following FTP clients that we tested:

- ▶ FileZilla client
- ▶ Total Commander
- ▶ Core FTP Language Environment

If you are trying to set up authentication by using a specific client, first check if this client allows the configuration of a keyring file (.pem, .pfx, and so on) that contains the client certificate. This is the prerequisite for any further setup steps.

Client authentication with Telnet

For more information about how to set up SSL client authentication with IBM Personal Communications and Attachmate Extra emulators, see *Security on IBM z/VSE*, SG24-7691.

3.7.15 Using TLSv1.2

TLSv1.2 is the newest SSL protocol version. It introduces new SSL cipher suites that use the SHA-256 hash algorithm instead of the SHA-1 function. This adds significant strength to the data integrity. TLSv1.2 is described in the memo *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, which is available at this website:

<http://tools.ietf.org/html/rfc5246>

The following new SSL cipher suites and their related hexadecimal values are available:

- ▶ 0x3B TLS_RSA_WITH_NULL_SHA256
- ▶ 0x3C TLS_RSA_WITH_AES_128_CBC_SHA256
- ▶ 0x3D TLS_RSA_WITH_AES_256_CBC_SHA256

Note: TLSv1.2 is supported by OpenSSL 1.0.1e, which is the current code level on z/VSE with APAR DY47499 / PTF UD53983 and later. TLSv1.2 is used by IPv6/VSE with APAR PM98875 / UK98397 and later.

There are only a few SSL servers and clients that support TLSv1.2. The following topics describe the scenarios in which TLSv1.2 can be used with IPv6/VSE.

Using VSE Java-based connector

TLSv1.2 is supported by Java 7 from IBM and Sun/Oracle. Therefore, any Java application that uses the VSE Connector Client can use TLSv1.2. Complete the following steps to set up VSE Navigator for TLSv1.2:

1. Set up the VSE Navigator SSL properties file (`ssl.prop`), as shown in the following example:

```
#-----  
# SSL Version:      SSL (3.0) or  
#                  TLS (TLS 1.0) or  
#                  TLSv1.2  
#-----  
SSLVERSION=TLSv1.2
```

2. Specify at least one of the TLSv1.2 cipher suites in `ssl.prop`, as shown in the following example:

```
#-----  
# TLS 1.2 cipher suites  
#-----  
CIPHERSUITES=TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_256_CBC_SHA256,TLS_RSA_WITH_NULL_SHA256
```

Note: The use of `TLS_RSA_WITH_AES_256_CBC_SHA256` requires the unlimited security strength files for Java. You can download these files from the following website:

<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>

No special definitions are required to set up BSTTATLS. Because we configured the client for TLSv1.2, you can use standard settings, such as the settings that are shown in the following example:

```
KEYFILE nnnn  
OPTION SERVER  
SECTYPE TLSV1  
ATTLS 2893 AS 2894 SSL
```

3. Start VSE Connector Server. No change is required. After the connection is made, right-click the VSE host icon in the VSE Navigator main window and select menu **Show server certificate**, as shown in Figure 3-49.

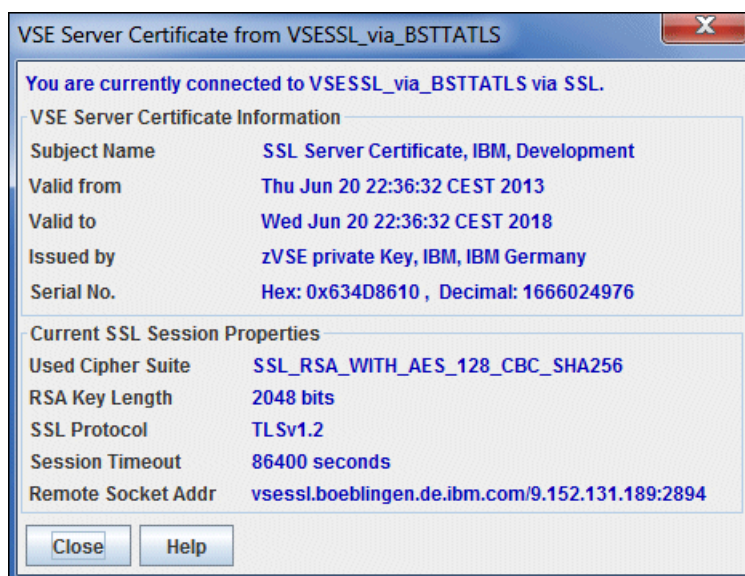


Figure 3-49 VSE Navigator Show server certificate box

Using secure FTP

We tested the following scenarios for FTP:

- ▶ VSE as the server. FileZilla client or Total Commander FTP client connect to BSTTATLS.
- ▶ VSE as the client. BSTTFTPC connects to FileZilla server by using BSTTPRXY.

VSE as the server

For this scenario, we must configure BSTTATLS and FileZilla client. Complete the following steps:

1. Configure and start BSTTATLS. You can specify SECTYPE TLSv1.2 to force all clients that are using TLSv1.2. However, in our tests, both clients (FileZilla and Total Commander) were using the highest available protocol version, even when TLSV1 was specified, as shown in the following example:

```
KEYFILE nnnn
SECTYPE TLSV1.2
OPTION SERVER
OPTION FTP
ATTLS 21 AS 990 SSL
```

2. Configure and start BSTTFTPS. No special definitions are required.
3. Configure and start FileZilla client.
 - a. In our tests, we received error "GnuTLS error -53: Error in the push function" immediately after the FileZilla client switched to passive mode. We resolved this issue by consulting the FileZilla Network Configuration page, which is available at this website:

https://wiki.filezilla-project.org/Network_Configuration

After we switched to active mode and specified the port range as 50000 - 60000, the error was corrected, as shown in Figure 3-50.

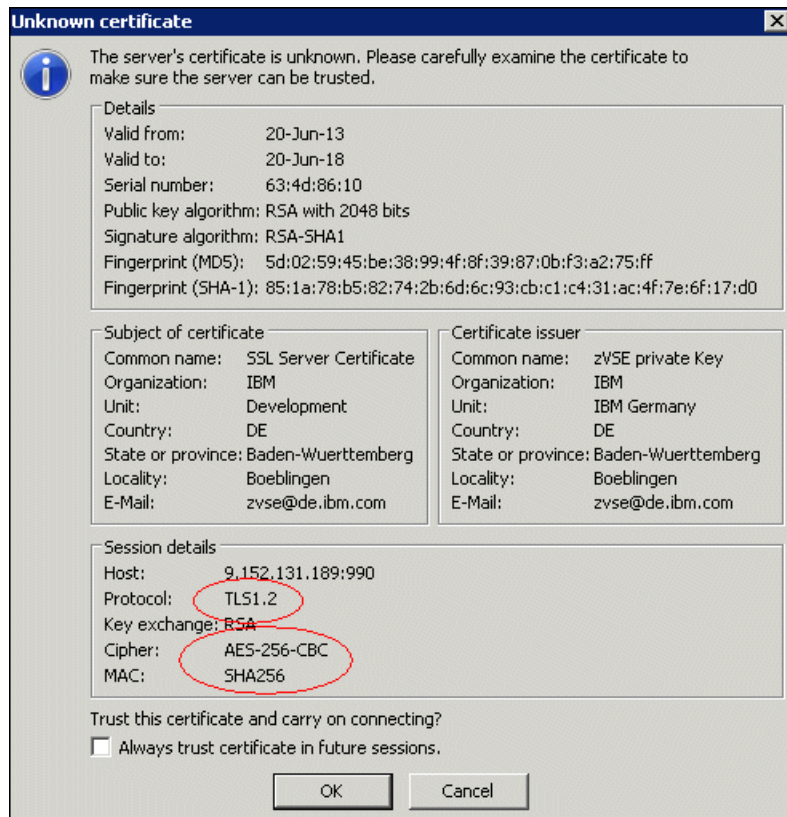


Figure 3-50 FileZilla connecting to BSTTATLS that uses TLSv1.2

Not all FileZilla client versions support TLSv1.2. In our tests, we used FileZilla client 3.7.0.2.

- b. Configure and start Total Commander. We also had to use active mode here. In our tests, Total Commander selected cipher suite 3D (AES256-SHA256). The SSL configuration was identical to what is shown in Figure 3-38 on page 97.

VSE as the client

In our tests, we used FileZilla server and Cerberus FTP Server (<http://www.cerberusftp.com>) on Windows. We also made a test with secure FTP daemon (VSFTPD), which runs on Linux. It appeared that FileZilla server (0.9.41 beta) does not support TLSv1.2. VSFTPD and Cerberus FTP Server support TLSv1.2. Complete the following steps to configure and use VSFTPD:

1. Configure and start BSTTPRXY or BSTTATLS. The use of SECTYPE TLSv1.2 is important here. BSTTPRXY is used in the following example:

```
KEYRING PRD2.CONFIG
KEYFILE FILEZ01
SECTYPE TLSv1.2
OPTION CLIENT
HANDSHAKE_CLIENT 0
OPTION FTP
PROXY TCP V4 21 TXT * TO V4 990 SSL * 9.152.131.28
```

2. Configure and start VSFTPD server. You can enforce the use of TLSv1.2 cipher suites by using the `ssl_ciphers` parameter in the `vsftpd.conf` file, as shown in the following example:

```
ssl_ciphers=AES256-SHA256:AES128:SHA256
```

Because VSFTPD uses OpenSSL internally, the `ssl_ciphers` parameter must be in OpenSSL syntax. The original OpenSSL documentation is available at this website:

<http://www.openssl.org/docs/apps/ciphers.html#>

From this documentation, it appears that keywords also can be specified for groups of cipher suites, such as “all ciphers with SHA256”. Here, you specify `ssl_ciphers=SHA256`.

3. Start BSTTFTPC. No special setup is necessary. For more information, see Example 3-29 on page 99.

TN3270

Most of our known TN3270 emulators do not support TLSv1.2, according to our research, including the following emulators:

- ▶ IBM Personal Communications. The latest PCOM version 6.0.7 supports up to TLSv1.1.
- ▶ Attachmate Extra. We could not find any information about TLSv1.2 in their documentation.
- ▶ The wc3270 emulator. Because this emulator uses openssl internally, support for TLSv1.2 depends on the installed openssl version on your workstation. It worked with openssl 1.0.1e installed.

The following setup was used for wc3270:

```
KEYFILE SSLCA  
SECTYPE TLSV1  
OPTION SERVER  
ATTLS 23 AS 992 SSL
```

With these BSTTATLS definitions, wc3270 automatically connected with TLSv1.2. Therefore, wc3270 appears to always use the highest possible protocol version. You also can explicitly specify `SECTYPE TLSV1.2` to enforce TLSv1.2 for all clients.

We are not aware of any wc3270 log file; therefore, you must check the BSTTATLS trace to see which protocol version is used.

Example 3-36 shows an extract from the BSTTATLS trace that shows the list of cipher suites that are sent by the client. It appears that wc3270 sent the TLSv1.2 cipher suites AES128-SHA256 and AES256-SHA256.

Example 3-36 List of SSL cipher suites that are sent by wc3270

Client sent 69 from 53B910:

```
...  
  AES256-SHA256  
  AES256-SHA  
...  
  AES128-SHA256  
  AES128-SHA  
...
```

The summary information from `gsk_secure_soc_init()` in Example 3-37 shows the used cipher suite.

Example 3-37 Summary from `gsk_secure_soc_init()` in BSTTATLS trace

Summary:	
fd	= 0x00000005
hs_type	= 1 (GSK_AS_SERVER)
DName	= [SSLCA]
keyring type	= LIBR
keyring	= DD:PRD2.CONFIG(SSLCA.PEM)
sec_type	= [TLSV1]
cipher_specs (V2)	= []
v3cipher_specs	= [3C3D05043538392F32330A161309151203060201]
skread	= 0x00514400
skwrite	= 0x0051D908
cipherSelected (V2)	= []
v3cipherSelected	= [3D]
size of public key	= 256 Bytes (2048 bits)
failureReasonCode	= 0
gsk_data	= 0x0047C9B0

For this connection, cipher suite 3D (TLS_RSA_WITH_AES_256_CBC_SHA256) was used.

CICS Web Support

When we review the most popular web browsers, we see the following restrictions:

- ▶ Mozilla Firefox
Firefox supports TLSv1.2. For more information, see this website:
<http://support.mozilla.org/en-US/questions/959936>
- ▶ Microsoft Internet Explorer
You can turn on TLSv1.2 by using the Internet Options window. However, we had a problem with latest browser versions and our self-signed certificates. For more information, see “Connect by using SSL” on page 119.
- ▶ Google Chrome
Although we did not test Chrome, Chrome 30 is reported to support TLS 1.2.

3.8 Setting up CICS Web Support

An overview of setting up CICS Web Support based on TCP/IP for VSE/ESA is provided in *Security on IBM z/VSE*, SG24-7691. In the following section, we describe the differences for IPv6/VSE only.

3.8.1 Modifying the CICS startup job

In the CICS startup job, you must specify the SYSID of the BSTTINET partition and put the `lib.slib` of the IPv6/VSE installation into the LIBDEF chain, as shown in the following example:

```
// JOB CICSBSI                CICS/ICCF STARTUP
// OPTION SYSPARM='02'
// SETPARM LOCALGT=YES
...
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD1.BASED,PRD2.TCPIPB,PRD1.BASE, ...
```

The LOCALGT parameter is used only for performance. For more information, see *IPv6/VSE Programming Guide*, SC34-2617.

3.8.2 Defining the TCP/IP host name

The TCP/IP host name is defined by using the HOST command in the IPv6/VSE stack startup job, as shown in the following example:

```
ID 02
HOST VSESSL 9.152.131.189
```

3.8.3 Considerations for SSL

The use of SSL within CICS TS CWS directly is not possible because of the restriction of OpenSSL on z/VSE that requires an LE/C runtime environment. For more information, see Chapter 5, “OpenSSL” on page 151.

As of this writing, CICS directly uses the CSI SSL functions when a TCP/IP service is used with SSL enabled.

However, when the BSTTPRXY or BSTTATLS facilities are used, SSL connections to CICS TS CWS by using OpenSSL are supported. The BSTTATLS facility automatically converts the SSL browser connection to a clear text socket as with any other connection.

In the next topics, we describe how to set up a non-SSL TCP/IP service, configure BSTTATLS to proxy the clear connection to SSL, and connect to a web browser by using SSL.

Setting up a TCP/IP service

For this example, you set up a TCP/IP service without SSL, as shown in Example 3-38.

Example 3-38 TCP/IP service for CWS

TCpipservice	:	NOSSL	
Group	:	EXTRACT	
Description	==>		
Urm	==>	DFHWBADX	
Portnumber	==>	01080	1-65535
Certificate	==>		
Status	==>	Open	Open Closed
SSL	==>	No	Yes No Clientauth
Attachsec	==>	Verify	Local Verify
TRansaction	==>	CWXN	
Backlog	==>	00001	0-32767
TSqprefix	==>		


```
Ipaddress    ==>
SOcketclose  ==> No
```

No | 0-240000

Before you continue, ensure that TCP/IP and the TCP/IP service are open.

Configuring BSTTATLS

Now we must configure BSTTATLS to proxy clear port 1080; for example, to secure port 1180, as shown in Example 3-39. For information about blocking the clear port against connections from outside, see 3.7.9, “Considerations on blocking clear ports” on page 84.

Example 3-39 BSTTATLS setup for CWS

```
// EXEC BSTTATLS,SIZE=BSTTATLS,PARM='TRAP(OFF) / '
ID 02
*
KEYRING PRD2.CONFIG
KEYFILE SSL01
SECTYPE TLV1
*
OPTION SERVER
ATTLS 1080 AS 1180 SSL
*
```

Configuring Firefox

In our tests, we experienced the problem that latest browser versions, such as Firefox 20, Internet Explorer 8, and Chrome, could not connect to CWS by using SSL. However, with early browsers, like Firefox 3.6, it worked. This behavior was the same whether we imported the keyring file (.pfx) into the browser or not.

Finally, we found out that it works when you import your certificates by using a .pem file rather than by using a .pfx file. To show this, we prepared two keyring files with the same content: One .pfx file and one .pem file.

First, we try to import the .pfx file in Firefox. At the time of writing this, we used Firefox 27.

Step 1: Open the Firefox Options box by clicking **Help** → **Options**.

Step 2: Open the Certificates tab and click **View Certificates**.

Step 3: On the Your Certificates tab, click **Import**.

Step 4: Select the .pfx file on the Open window. Note that the file filter shows “PKCS12 Files (*.p12;*.pfx)”.

Step 5: You are prompted to enter the .pfx file password.

In our tests, we now get an error shown in Figure 3-51. Obviously, Firefox cannot import the certificates for any unknown reason.

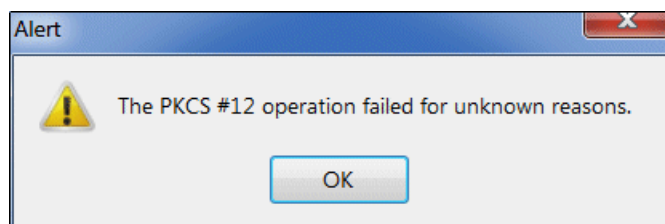


Figure 3-51 Error message from Firefox 27 when importing a .pfx file

However, when importing the same set of certificates by using a .pem file, the process works and we are able to connect to CWS.

Repeat steps 1 to 2. Then select the **Servers** tab and press **Import**. Note that the file filter on the Open window shows "Certificate files (.crt;*.cert;*.cer;*.pem;*.der)".

Select the .pem file and press **Open**. Firefox issues a warning message that the self-signed certificate is not trusted and can therefore not be imported, but the VSE server certificate is imported under Servers as shown in Figure 3-52.

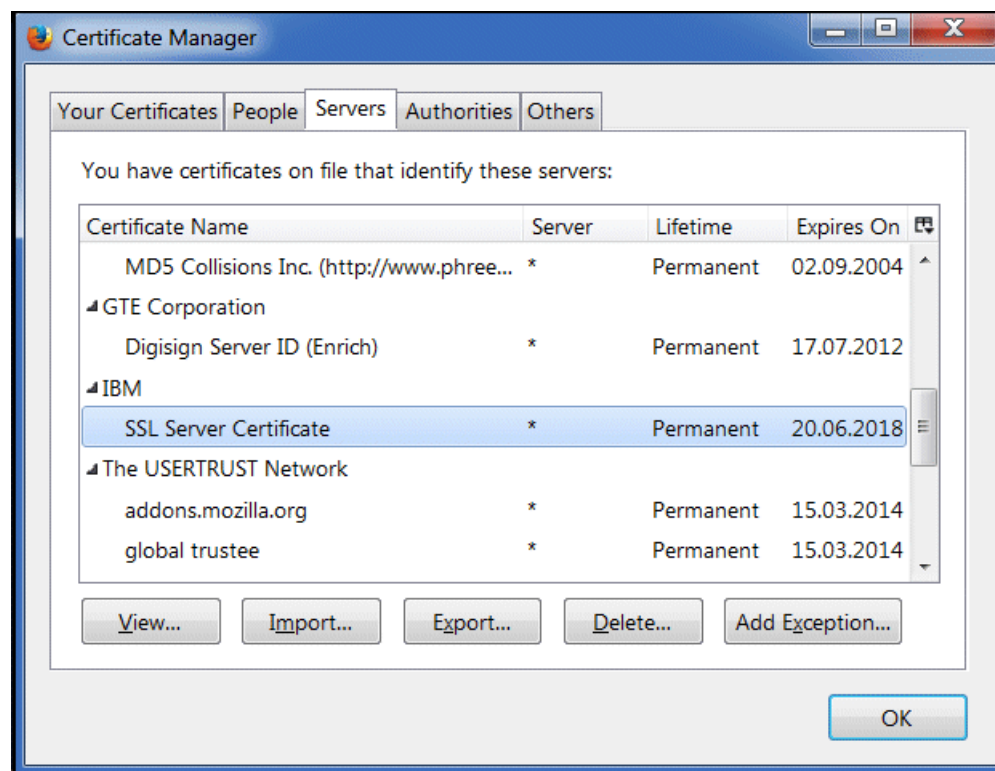


Figure 3-52 Imported self-signed VSE server certificate in Firefox 27

Firefox 27 is now enabled to connect to CWS by using SSL.

Connect by using SSL

Before we can connect by using SSL, you must close and reopen the TCP/IP service to give BSTTATLS control. When you are reopening the TCP/IP service, you should see the following line on the console:

```
BSTT719I      4:      4 Listening on port      1180 SSL
```

You can now connect to the secure port 1180 as shown in Figure 3-53. In this test, we used following URL:

`https://vsessl:1180/CICS/CWBA/DFHwbttta/cemt`

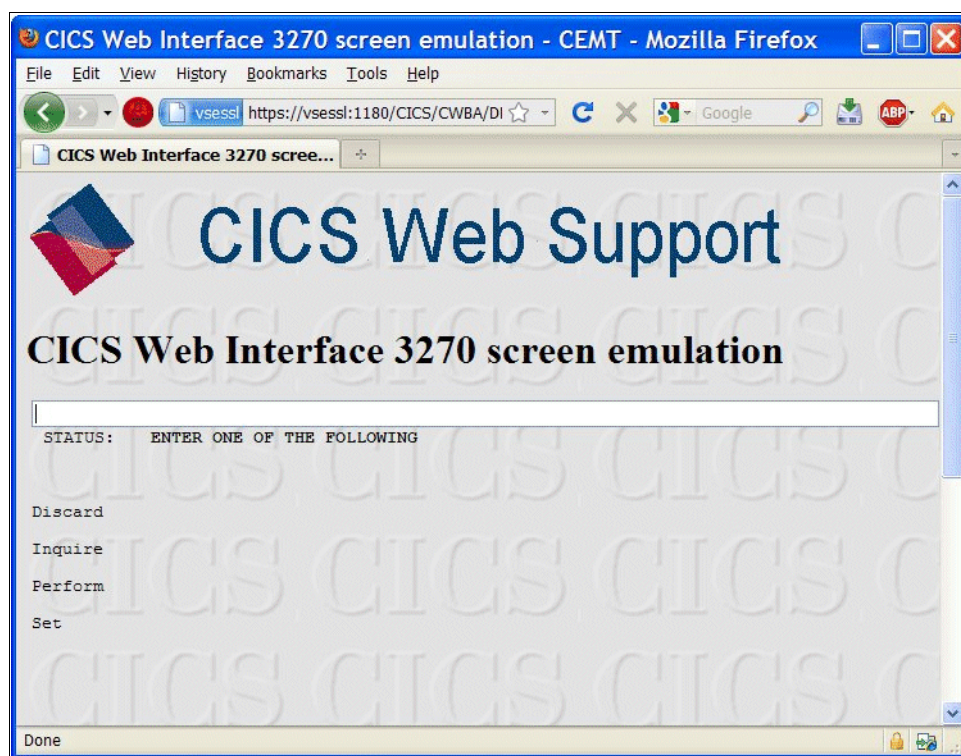


Figure 3-53 HTTPS connection to CICS Web Support

3.9 Known problems

In this section, we describe some of the issues and insights we had in our test setup.

3.9.1 VSE cannot be reached

A problem in reaching VSE was reported.

Symptom

Your workstation cannot reach your VSE system in the IPv6 network.

Possible reason

Check whether your workstation can reach other IPv6 hosts in your network. In our test setup, we found that a workstation program negatively influenced IPv6 on Windows XP.

3.9.2 BSTT075E LUNAME NOT AVAIL

A problem with BSTT075E LUNAME NOT AVAIL was reported.

Symptom

Message BSTT075E is issued repeatedly on console.

Possible reasons

This error occurred for the following possible reasons:

- ▶ Printer sessions are closed without disconnecting first.
- ▶ The same printer session is started twice. The problem can be solved by using the REACTIVATE command; for example, reactivate PRTA.
- ▶ You are not using parm OS390 on // EXEC BSTTVNET (for more information, see 3.2, “Obtaining and activating a license key” on page 48).
- ▶ VNETAPPL is not activated correctly (for more information, see 3.5.1, “Setting up VTAM” on page 62). Check ATCCON00.B.

3.9.3 SSL connect error with wc3270

A problem concerning SSL connecting with wc3280 was reported.

Symptom

We received the following error messages when wc3270 was started.

[wc3270]

```
SSL_connect: error in SSLv3 read server certificate B error:14090086:SSL
routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed
SSL_read: unknown error
wc3270>
```

Possible reason

You did not add your self-signed root certificate to the wc3270 root_certs.txt file.

3.9.4 Other SSL connect errors

An SSL connect error was reported.

Symptom

SSL connection cannot be established.

Possible reasons:

Check the SYSLST output of BSTTATLS or BSTTPRXY for errors. The following errors are typical:

- ▶ *** Error: DName is invalid.
Check whether the KEYFILE parameter points to the correct .pem file
- ▶ *** Error: Cannot load key file [DD:PRD2.CONFIG(SSL01.PEM)]
Check whether the .pem file is in the specified place.

- ▶ *** Error setting key stuff.

Your certificates might be in the wrong order in the .pem file. The order must be: key - vsecert - rootcert.

These errors are issued from the IJBSSL phase.

3.9.5 Hang situation with BSTTATLS/BSTTPRXY

A problem concerning a hang issue with BSTTATLS/BSTTPRXY was reported.

Symptom

The SSL client and BSTTATLS/BSTTPRXY hang.

Possible reasons

The BSTTATLS/BSTTPRXY partition has a lower priority than the application partition.

You can also add a // SETPARM SENDALL='YES' to the connector server JCL. For more information about this parameter, see *IPv6/VSE Programming Guide*, SC34-2617.

Your client is the VSE Connector Client, you are using a local .pem file, and this .pem file contains Diffie-Hellman (DH) parameters. In this case remove the DH parameters from your local .pem file. DH parameters should only be contained in the server .pem file.

3.9.6 Return codes 3100 / 1700 from IJBCRLIB

A problem concerning return codes 3100 / 1700 from IJBCRLIB was reported.

Symptom

Trace messages that are similar to the following example are displayed on the console:

```
IJBCRLIB.CrInit: CRYINITI, rc = 1701
```

Reason

The IJBSSL trace is activated (// SETPARM SSL\$DBG='YES'). These return codes are issued from phase IJBCRLIB when its internal crypto environment is started in the following cases:

- ▶ rc = 1700 - Phase IPCRYPTO (which is part of the CSI stack) cannot be loaded. This is not an error when the IPv6/VSE or Linux Fast Path (LFP) stack is used, for example.
- ▶ rc = 3100 - Function CRYINITI, which is part of the CSI crypto API, returned this code. This often means that the CSI stack is installed (that is, phase IPCRYPTO can be loaded), but the stack is inactive. This is not an error when the IPv6/VSE or LFP stack is used, for example.

Phase IJBCRLIB cannot determine which IP stack is used and always tries to CDLOAD the IPCRYPTO phase and call CRYINITI.

3.9.7 BSTTFTPC fails to connect to Windows Server 2008

A problem with BSTTFTPC failing to connect to Windows Server 2008 was reported.

Symptom

Running an FTP script on Windows Server 2008 and newer fails to connect for data connection.

Reason

By default, ftp.exe on Windows Server 2008 (and newer) is blocked. You must go into the firewall and unblock ftp.exe.

3.9.8 Batch email cannot relay mail

A problem with batch email's ability to relay mail was reported.

Symptom

Customers who try using batch email for the first time can encounter an error that indicates that they cannot relay mail.

Reason

For MS Exchange 2010, fixing this error includes the following steps:

1. Ensure that z/VSE is a trusted host.
2. Define a receive connector with the IP address of z/VSE, including the following information:
 - Allow Relay from: Give z/VSE a name (such as, z/VSE)
 - In Network tab: Use z/VSE's IP address, port 25
 - In Permission Groups tab: Select **Anonymous users**
 - In Authentication tab: Select **Externally Secured**

3.9.9 LDAP sign on by using SSL does not work

A problem with LDAP sign on by using SSL that does not work was reported.

Symptom

LDAP sign on by using SSL does not work with IPv6/VSE.

Reason

LDAP sign on by using SSL uses the GSK SSL API through the TCP/IP sockets phase as configured in the Language Environment multiplexer (skeleton EDCTCPMC in ICCF library 59). The GSK SSL API that is provided by OpenSSL cannot be accessed by using the Language Environment multiplexer. Therefore, LDAP SSL requires CSI SSL.

3.9.10 GnuTLS error -53: Error in the push function

A problem with GnuTLS error -53: Error in the push function was reported.

Symptom

When FileZilla client is used to connect to BSTATLS / BSTTFTPS, the GnuTLS error -53 is issued directly after FileZilla switches to passive mode.

Reason

This is a firewall issue. We resolved the problem with the information from the FileZilla Network Configuration page, which is available at this website:

https://wiki.filezilla-project.org/Network_Configuration

When active mode is used and the port range in FileZilla is restricted to 50000 - 60000, the connection worked.



Fast Path to Linux on System z

This chapter describes the Fast Path to Linux on System z function that allows selected TCP/IP applications that are running under z/VSE to communicate with a TCP/IP stack on Linux on System z without the use of a TCP/IP stack on z/VSE.

Linux Fast Path (LFP) was introduced with z/VSE V4.3 for use in a z/VM guest environment. With z/VSE 5.1, support for the use LFP in a z/VM IP Assist (IBM VIA®) environment and in a logical partition (LPAR) environment was added to extend the connectivity options for z/VSE clients. Linux Fast Path in an LPAR environment requires IBM zEnterprise technology with the HiperSockets Completion Queue function.

Although LFP provides TCP/IP socket APIs for programs that are running on z/VSE, it is not intended to replace existing TCP/IP stacks, such as TCP/IP for VSE/ESA and IPv6/VSE. You still need vendor applications, such as FTP, Telnet, LPR, or LPD in your IT environment.

This chapter includes the following topics:

- ▶ Overview
- ▶ Concept of LFP instances and LFP daemons
- ▶ LFP in a z/VM environment
- ▶ z/VM IP Assist
- ▶ LFP in an LPAR environment
- ▶ IBM applications that support LFP
- ▶ Using secure connections with SSL
- ▶ Known problems

4.1 Overview

By using LFP, selected TCP/IP applications can communicate with the TCP/IP stack on Linux on System z without the use of a TCP/IP stack on z/VSE. All socket requests are transparently forwarded to a Linux on System z system that is running in the same z/VM, or in an LPAR on the same System z processor. Since z/VSE V5.1, LFP supports IPv6.

LFP can be used in the following environments:

- ▶ LFP in a z/VM environment

When LFP is used in a z/VM environment, z/VSE and Linux on System z run as z/VM guests in the same z/VM-mode LPAR on IBM z10, z114, z196, zEC12, or zBC12 servers. They use an Inter-User Communication Vehicle (IUCV) connection between z/VSE and Linux.

- ▶ LFP in a z/VM IP Assist (z/VSE VIA) environment

The z/VSE VIA function uses a pre-configured appliance that is loaded from the system's Support Element (SE). This removes the need for configuring a Linux guest system. Another z/VM CMS guest is required to configure the z/VSE VIA function.

- ▶ LFP in an LPAR environment

When LFP is used in an LPAR environment, z/VSE and Linux on System z run in their own LPARs on a zEnterprise server and a HiperSockets connection is used between z/VSE and Linux on System z. In this case, LFP requires the HiperSockets Completion Queue (HSCQ) function that is available with a zEnterprise server (z196, z114, and zEC12).

The LFP on System z provides standard TCP/IP socket APIs for programs that are running on z/VSE. Other than the basic socket API, no other tools are provided.

Performance improvements include the following results:

- ▶ Less effect on TCP/IP processing on z/VSE (TCP, sequence numbers and acknowledging, checksums, resends, and so on).
- ▶ A more reliable communication method (IUCV) compared to HiperSockets, which is a network device with packet drops, resends, and so on.

Table 4-1 summarizes the availability of LFP.

Table 4-1 Availability of LFP

z/VSE	5.1.1	5.1.0	4.3
LFP	yes	yes	yes
LFP-VIA	yes	yes	N/A
LFP in LPAR (requires z196 GA2 or later, or z114 or later)	yes	N/A	N/A

Note: Connectivity Systems International (CSI) applications, such as FTPBATCH, do not work with the LFP stack because they use internal CSI interfaces. However, applications that are provided with the IPv6/VSE product from Barnard Software, Inc. and applications and utilities that are provided by IBM can be used. For more information, see 4.6, “IBM applications that support LFP” on page 144.

For a complete list of hardware and software prerequisites, see *z/VSE TCP/IP Support*, SC34-2640.

4.2 Concept of LFP instances and LFP daemons

Setting up LFP always means configuring one or more LFP instances on z/VSE, each having one counterpart on Linux or VIA and an LFP daemon (LFPD). LFP instances are identified by their system ID (SYSID) (00 - 99), which allows for a maximum of 100 instances per VSE system.

Figure 4-1 shows an overview of two VSE systems with a total of three LFP instances that are connecting to one Linux with three LFP daemons. All systems run under z/VM.

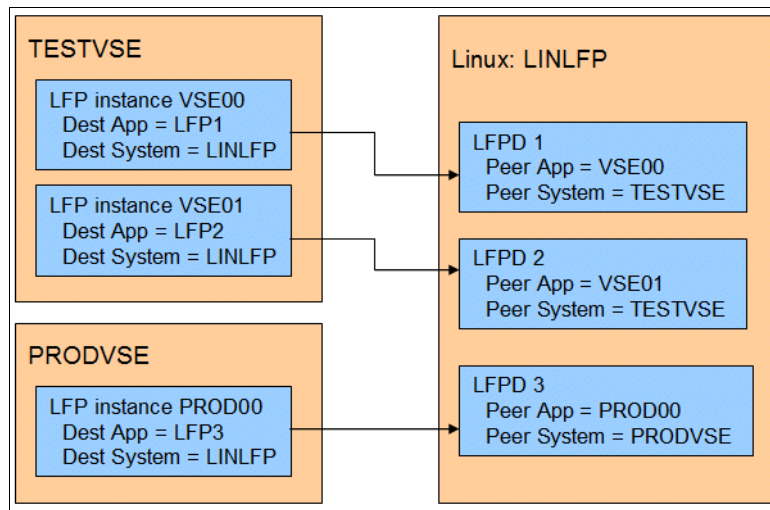


Figure 4-1 LFP instances and their counterparts on Linux on System z

In general, each LFP instance and LFP daemon identify their peers by using a destination system name and a destination application name. Depending on which environment is used (LFP under z/VM, VIA, or LFP in LPAR), these names have different meanings.

An LFP daemon often is dedicated to exactly one z/VSE LFP instance. This is controlled by an optional parameter `peer_iucv_vmid`, which specifies the peer z/VM ID that can connect to this LFPD. If this parameter is not specified, connections from any LFP instance on any VSE system are accepted. However, only one LFP instance can use an LFPD at the same time.

For more information about LFP configuration parameters, see *z/VSE TCP/IP Support*, SC34-2640.

The following sections describe each of the three LFP environments.

4.3 LFP in a z/VM environment

In a z/VM environment, the LFP uses an IUCV connection between z/VSE and Linux, where both systems run in the same z/VM-mode LPAR on a System z10™ or later processor. Selected TCP/IP applications can communicate with the TCP/IP stack on Linux without the use of a TCP/IP stack on z/VSE.

Figure 4-2 shows a scenario in which an IBM DB2 client that is running on z/VSE communicates with a DB2 Server for Linux by using LFP. No TCP/IP stack is running on z/VSE.

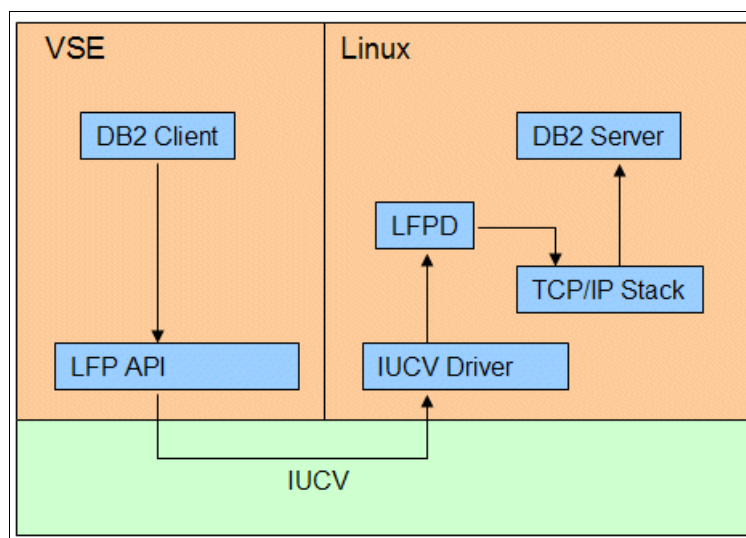


Figure 4-2 LFP in a z/VM environment

LFP directly communicates with the Linux device driver for IUCV, which passes the data to the LFP daemon. The LFPD receives the data, translates it into socket calls, and sends it to the Linux TCP/IP stack, which, in turn, forwards the data to the DB2 Server.

The following sections describe how to set up the VSE and Linux guest systems. The complete setup includes two z/VM users:

- ▶ VM user that is running a VSE guest
- ▶ VM user that is running the Linux guest

4.3.1 Linux guest setup

On the Linux side, you must install and configure the LFPD.

Downloading the LFPD

The LFPD is part of VSE Connectors and VSE/AF (5686-CF9-38/06). It is shipped on the z/VSE Extended Base Tape as part of component 5686CF8-38 CONN.C/W. After the VSE Connectors Workstation Code component is installed, it is available as member IJBLFPLX.W in PRD2.PROD. However, you can always download the latest version from the following VSE home page:

<http://www.ibm.com/systems/z/os/zvse/downloads/>

Installing the LFPD

When you download member IJBLFPLX.W from z/VSE, you must rename the downloaded file to IJBLFPLX.zip before you can install it by using IBM Rational® Portfolio Manager.

For more information about installing and configuring the LFPD, see Chapter 13, “Running z/VSE With a Linux Fast Path” of *z/VSE TCP/IP Support*, SC34-2640.

Configuring the LFPD

For each LFPD, you must provide a configuration file, which is read during the start of LFPD and cannot be changed while LFPD is running.

According to *TCP/IP Support*, each configuration file must be named `lfpd-XXX.conf`, where XXX is the `IUCV_SRC_APPNAME` or `HS_SRC_APPNAME` that is specified in the configuration file. The XXX characters in the file name must be specified in uppercase. For example, the configuration file with an `IUCV_SRC_APPNAME` of `LINLFP` is named `lfpd-TESTVSE.conf`.

You must store all available configuration files in the `/etc/opt/ibm/vselfpd/confs-available` directory. The `/etc/opt/ibm/vselfpd/confs-enabled` directory contains the enabled configuration files. If a configuration is enabled, it can be easily started with the LFPD control script `lfpd-ctl`. Additionally, the SysV init script starts all enabled configurations during the start of the Linux system. For each available configuration that is to be enabled, you must create a symbolic link.

In our tests, we created a configuration with the name `LINLFP`, so the configuration file is `/etc/opt/ibm/vselfpd/confs-available/lfpd-TESTVSE.conf`. Example 4-1 shows the contents of the configuration file.

Example 4-1 LFPD configuration file for LFP under z/VM

```
# lfpd configuration file
IUCV_SRC_APPNAME = LINLFP
# ensure that only TESTVSE from TESTVSE can connect
PEER_IUCV_VMID = TESTVSE
PEER_IUCV_APPNAME = TESTVSE
IUCV_MSGLIMIT = 1024
MTU_SIZE = 8192
MAX_SOCKETS = 1024
INITIAL_IO_BUFS = 128
WINDOW_SIZE = 65535
WINDOW_THRESHOLD = 25
VSE_CODEPAGE = EBCDIC-US
VSE_HOSTID = 10.0.0.1
RESTRICT_TO_HOSTID = yes
LOG_INFO_MSG = no
```

To enable the configuration, place a symbolic link from the configuration file to the enabled configurations directory by using the following command:

```
ln -s /etc/opt/ibm/vselfpd/confs-available/lfpd-LINLFP.conf
/etc/opt/ibm/vselfpd/confs-enabled
```

Starting the LFPD

Example 4-2 shows how to start the LFPD on Linux.

Example 4-2 Starting the LFPD on Linux

```
linlfp:~ # lfpd-ctl start testvse
Starting lfpd (TESTVSE): success
```

The next section describes how to set up the corresponding VSE guest.

4.3.2 VSE guest setup

In our test setup, the configuration in Example 4-3 was used for the VSE guest system.

Example 4-3 VSE configuration for LFP

```
User = TESTVSE
IP address: none
```

The VSE guest system was named TESTVSE. There is no IP stack active and there is no assigned IP address. To enable the VSE system for LFP, the VM directory entry must be updated. Then, an LFP instance must be configured and started on VSE.

VM directory entry

The statements that are shown in Example 4-4 must be included into the VM directory entry to enable IUCV communication.

Example 4-4 IUCV definitions in VM directory entry for LFP

```
IUCV ANY PRIORITY MSGLIMIT 1024
IUCV ALLOW
```

Example 4-5 shows the complete directory entry from our test system.

Example 4-5 Complete z/VM directory entry

```
USER TESTVSE BBRK 64M 128M GUB
*-----
  ACCOUNT 203 DOSSYS
  CPU 00 CPUID 100080 NODEDICATE
  CPU 01 CPUID 100081 NODEDICATE
  CPU 02 CPUID 100082 NODEDICATE
  IPL CMS
  IUCV ANY PRIORITY MSGLIMIT 1024
  IUCV ALLOW
  MACHINE ESA
  OPTION MAXCONN 15 QUICKDSP MAINTCCW TODENABLE
  DEDICATE OD00 F54F
  DEDICATE OD01 F550
  DEDICATE OD02 F551
*-----
  CONSOLE 0009 3215 T
  SPECIAL 0060 3270
  SPECIAL 0061 3270
  SPECIAL 0062 3270
  SPECIAL 0063 3270
  SPECIAL 0064 3270
*-----
  SPOOL 000C 2540 READER A
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 4248 W
*--OSA-----
  LINK MAINT 0190 0190 RR
  LINK MAINT 019E 019E RR
  LINK TCPMAINT 0592 0592 RR
```

After the VM directory entry is set up, LFP must be configured in Linux and z/VSE.

LFP configuration in VSE

ICCF library 59 contains the following skeletons for LFP-related configuration tasks:

- ▶ SKLFPACT: Activate Language Environment-socket interface for LFP. This is necessary to allow Language Environment programs that use LFP
- ▶ SKLFPINF: Get information for an LFP instance
- ▶ SKLLFPLST: List active LFP instances
- ▶ SKLLFPSTA: Start an LFP instance
- ▶ SKLFPSTO: Stop an LFP instance

Starting an LFP instance

The job control language (JCL) that is shown in Example 4-6 is derived from skeleton SKLFPSTA.

Example 4-6 JCL for starting an LFP instance

```
* $$ JOB JNM=LFPSTART,CLASS=A,DISP=D
// JOB LFPSTART START AN LFP INSTANCE
// EXEC IJBLFPOP,PARM='START DD:SYSIPT LOGALL'
ID                = 01          <- SYSID of the LFP stack
MTU               = 8192
IUCVMSGLIMIT     = 1024
INITIALBUFFERSPACE = 512 K
MAXBUFFERSPACE   = 4M
IUCVSRCAPPPNAME  = TESTVSE     <- must match the LFPD config file on Linux
IUCVDESTAPPPNAME = LINLFP      <- name of the LFP application on Linux
IUCVDESTVMID     = LINLFP      <- Linux VM user
WINDOWSIZE       = 65535
WINDOWTHRESHOLD  = 25
/*
/&
* $$ EOJ
```

When this LFP instance is active, it works like another IP stack on your VSE system. We used ID = 01 because SYSID=00 was already used by the CSI stack.

Running the console output that is shown in Example 4-7 starts an LFP instance with the SYSID.

Example 4-7 Console output from starting an LFP instance

```
BG 0000 // JOB LFPSTART START AN LFP INSTANCE
        DATE 08/24/2011, CLOCK 13/16/53
BG 0000 LFPB013I  STARTED LFP INSTANCE '01'.
BG 0000 EOJ LFPSTART  MAX.RETURN CODE=0000
```

An LFP instance is no long-running server task. When a TCP/IP application makes a socket call, LFP code is loaded and run in the caller's partition. When data arrives from outside, LFP code is triggered by the z/VSE interrupt handler and runs in the z/VSE supervisor's context.

When you are querying the status of the LFP daemon on Linux, it shows the TESTVSE application that is connected to the VSE system, as shown in Example 4-8.

Example 4-8 Status of LFP daemon

```
linlfp:~ # lfpd-ctl list
name      enabled  running  status
-----
TESTVSE   yes       yes      Connected to TESTVSE
```

The next step enables VSE Language Environment applications to use the LFP stack.

Configuring the Language Environment multiplexer

Enabling Language Environment applications for LFP is done by configuring the TCP/IP multiplexer phase, as shown in Example 4-9. You can use skeleton EDCTCPMC in ICCF library 62.

Example 4-9 Configuring the TCP/IP multiplexer phase

```
* $$ JOB JNM=EDCTCPMC,CLASS=A,DISP=D,LDEST=*,PDEST=*
// JOB EDCTCPMC - GENERATE TCP/IP MULTIPLEXER CONFIG PHASE
// LIBDEF *,CATALOG=PRD2.CONFIG
// LIBDEF *,SEARCH=(PRD2.SCEEBASE,PRD1.BASE)
// OPTION ERRS,SXREF,SYM,NODECK,CATAL,LISTX
// PHASE EDCTCPMC,*,SVA
// EXEC ASMA90,SIZE=(ASMA90,64K),PARM='EXIT(LIBEXIT(EDECKXIT)),SIZE(MAXC
-200K,ABOVE)'

EDCTCPMC CSECT
EDCTCPMC AMODE ANY
EDCTCPMC RMODE ANY
*
          EDCTCPME SYSID='00',PHASE='$EDTCPV'   For use with CSI/IBM
          EDCTCPME SYSID='01',PHASE='IJBFLPLE'   For use with LFP
          EDCTCPME SYSID='02',PHASE='BSTTTCP6'   For use with IPV6/VSE
*          EDCTCPME SYSID='03',PHASE='VENDTCPI'   Other vendor interface
*
          END

/*
// IF $MRC GT 4 THEN
// GOTO NOLINK
// EXEC LNKEDT,PARM='MSHP'
/. NOLINK
/*
/&
* $$ EOJ
```

Phase IJBFLPLE provides the socket API for LFP. The SYSID, with the Language Environment sockets interface phase, allows an application to use a specific IP stack. Although there might be multiple IP stacks active at any time, this means that one given application can use only one IP stack (which is specified by the SYSID in its startup JCL).

In the next section, we describe how to replace the Linux side by the z/VM IP Assist (VIA) function. The VSE side remains unchanged.

4.4 z/VM IP Assist

This section describes the setup of LFP by using z/VSE VIA (z/VM IP Assist). This simplifies the setup of LFP by using a pre-configured appliance with the system's Support Element (SE). The VIA image is booted from a file that is on the SE and loaded initially in a z/VM user that must be configured for IUCV. The complete setup includes the following VM users:

- ▶ VM user that is running a VSE guest
- ▶ VM user that is running the VIA guest
- ▶ VM user that is used for configuring the VIA user

Figure 4-3 shows an overview of the involved VM users and how they are connected.

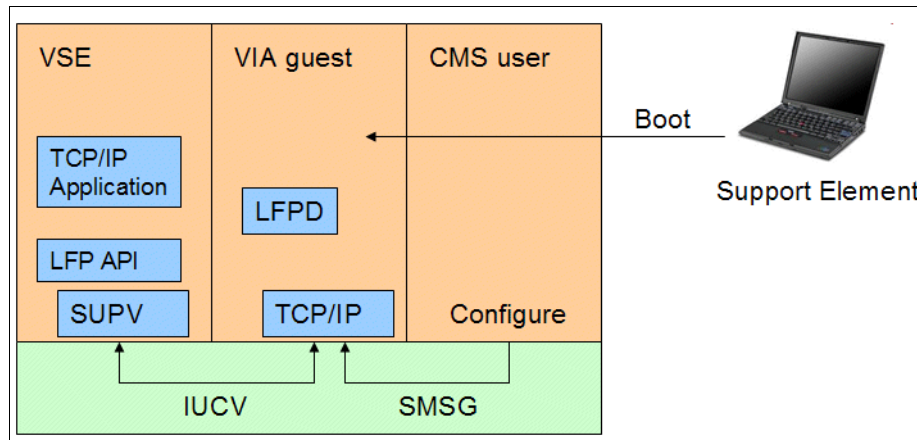


Figure 4-3 Overview of z/VM users for LFP

The VIA guest and configuration user must be configured to access two CMS minidisks with static addresses D4C (config disk) and D4D (data disk). The data disk is needed for debugging only, so it is normally not used. Table 4-2 shows how the disks are accessed.

Table 4-2 Access of config and data disks

Disk	VIA user	Configuration user
Config disk (D4C)	Read Only	Read/Write
Data disk (D4D)	Read/Write	Not needed

In the following sections, we describe how the three VM users are configured.

4.4.1 Configuration user setup

The configuration user is a standard CMS user that must be authorized to link the VIA configuration disk D4C in read/write mode.

Note: The first thing to check is that the two disks are formatted as CMS minidisks.

In our test setup, the configuration that is shown in Example 4-10 was used.

Example 4-10 Configuration user setup for LFP

VM user = JSCHMIDB

You must create the following files on the configuration disk:

- ▶ The SENDERS.ALLOWED file contains a list of authorized configuration users
- ▶ The LFPDCONF file contains information about the LFPD configuration

Configuring the SENDERS.ALLOWED file

To prevent unauthorized SMSG traffic, each sender is validated against a list of authorized users that is contained in a CMS file SENDERS.ALLOWED, which is on the configuration disk (0D4C). This file contains one z/VM user ID per line. All specified IDs are authorized to send SMSG commands to the z/VSE VIA guest.

To set up the CMS file, you must link the configuration disk in read/write mode, as shown in Example 4-11. Then, you use XEDIT to create the file.

Example 4-11 Linking the config disk

```
link vsevia d4c d4c mr
Ready; T=0.01/0.01 11:32:02
CP Q DASD
...
DASD 0D4C 3390 35456R R/W          10 CYL ON DASD  6078 SUBCHANNEL = 0008
```

The configuration disk is empty when it is accessed the first time. You can use XEDIT to create the SENDERS.ALLOWED file. Insert one line with the name of your configuration user, as shown in Example 4-12.

Example 4-12 Creating the senders.allowed file

```
===== * * * Top of File * * *
===== JSCHMIDB
===== * * * End of File * * *
```

Configuring the LFPDCONF files

You must configure one LFPDCONF file for each LFPD instance. For more information, see Chapter 13, “Running z/VSE with a Linux Fast Path”, of *z/VSE V4 R3.0 TCP/IP Support*, SC34-2640.

The configuration files have the same content as described in the official documentation of the LFPD config files in the z/VSE TCP/IP Support manual. Each configuration file must have the same file name as the IUCV_SRC_APP that it configures, and it must have the CMS file type LFPDCONF.

In Example 4-13, we create one configuration file that is called TESTVSE.LFPDCONF.

Example 4-13 Creating the lfpdconf file

```
# lfpd configuration file for TESTVSE
IUCV_SRC_APPNAME = TESTVSE
IUCV_MSGLIMIT = 1024
MTU_SIZE = 8192
MAX_SOCKETS = 1024
INITIAL_IO_BUFS = 128
WINDOW_SIZE = 65535
WINDOW_THRESHOLD = 25
VSE_CODEPAGE = EBCDIC-US
VSE_HOSTID = 9.152.111.11      <- valid static IP address of VSEVIA
RESTRICT_TO_HOSTID = no
```

At the end of this step, we have these files on the configuration disk, as shown in Example 4-14.

Example 4-14 Overview on configuration files

Cmd	Filename	Filetype	Fm	Format	Lrec1	Records	Blocks	Date	Time
	TESTVSE	LFPDCONF	C1	F	80	10	1	8/24/11	13:31:19
	SENDERS	ALLOWED	C1	F	80	1	1	8/24/11	11:36:06

The configuration user is now ready. The next step is configuring the VIA guest.

4.4.2 Setting up the VIA guest

VM setup for the VIA user includes the following tasks:

- ▶ Enabling the VIA user for IUCV, and
- ▶ Setting up the network connection to the VSE user

In our test setup, the configuration that is shown in Example 4-15 was used.

Example 4-15 VIA guest setup summary

```
VM user = VSEVIA
IP address: 9.152.111.11
Network address : 9.152.131.0/24
Subnet mask: 255.255.255.0
Broadcast: 9.152.131.255
Gateway: 9.152.108.1
Name server: ET : 9.152.120.241; 9.152.64.172
MTU Size: 1492
```

After initially loading the VIA image, there is no need (and no possibility) for any manual action from this VM user. The underlying appliance is configured by the configuration user by creating and modifying the configuration files on disk D4C. Therefore, you can use AUTOONLY to get the VM user loaded automatically at VM startup without any manual intervention.

VM directory entry

Example 4-16 shows the VM directory entry of our VSEVIA guest.

Note: The network definitions contain brackets (“[” and “]”) and braces (“{” and “}”). Therefore, you must use EBCDIC code page 924 in your terminal emulator to enter the characters correctly.

Example 4-16 VM directory entry for VIA user

```
USER VSEVIA BRITTA 1G 1G G
  COMMAND SET D8ONECMD * OFF
  COMMAND SET RUN ON
  COMMAND TERM LINEND #
* IPL VIA Image from SE
  IPL _0__1__
* IUCV definitions
  IUCV ANY PRIORITY MSGLIMIT 1024
  IUCV ALLOW
* Standard virtual devices
  LOADDEV PORT 0
  LOADDEV LUN 0
  LOADDEV BOOT 0
* Load Image 601 from SE
  LOADDEV BR_LBA 601
* Network adapters and configuration
  LOADDEV SCPDATA '{"profiles":["zVSE-VIA"],"networkCards":[{"OSA',
  '":"0D00","staticIPv4":"9.152.111.11/22"}],"defaultGateway":',
  '"9.152.108.1","DNS":["9.152.120.241","9.152.64.172"]',
  '","hostName":"vsevia"}'
* Machine type and number of CPUs
  MACH XA 1
  OPTION LXAPP LANG AMENG MAXCONN 128
  DEDICATE 0D00 F552
  DEDICATE 0D01 F553
  DEDICATE 0D02 F554
* Console
  CONSOLE 0009 3215 T
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
* OSA
  LINK MAINT 0190 0190 RR
  LINK MAINT 019D 019D RR
  LINK MAINT 019E 019E RR
* Minidisks
  MDISK 0D4C 3390 9279 10 35456R RR R W M
  MDISK 0D4D 3390 9289 10 35456R MR R W M
```

Administering the LFP daemon

After reloading the VSEVIA user, the LFPD daemon is automatically started in VSEVIA. You can check this by using SMSG commands from the configuration user.

The LFPD command features the following syntax:

```
SMSG <guest> APP LFPD (start|stop|restart|force-reload|status|startdbg) <IUCVNAME>
SMSG <guest> APP LFPD (list|startall|stopall)
```

Now we check to see whether the configuration is OK and if the LFPD is started. Normally, the output should look as it does in Example 4-17.

Example 4-17 Displaying the LFP daemon by using SMSG

```
smsg vsevia app lfpd list
Ready; T=0.01/0.01 14:13:40
14:13:40 * MSG FROM VSEVIA : NAME          ENABLED  RUNNING  STATUS
14:13:40 * MSG FROM VSEVIA : -----
14:13:40 * MSG FROM VSEVIA : TESTVSE    YES      YES      LISTENING
```

Debugging

If an error occurs, you can use the STARTDBG parameter of the LFPD command. In our first test (as shown in Example 4-18), the LFPD did not start and the debug output showed the reason.

Example 4-18 Debugging the LFP daemon

```
smsg vsevia app lfpd startdbg testvse
Ready; T=0.01/0.01 13:53:34
13:53:38 * MSG FROM VSEVIA : STARTING LFPD (TESTVSE): FAILED
13:53:38 * MSG FROM VSEVIA : STARTUP LOG:
13:53:38 * MSG FROM VSEVIA : LOGGING STARTUP MESSAGES TO FILE '/TMP/LFPD-CTL
.2567.TMP'.
13:53:38 * MSG FROM VSEVIA : READING CONFIGURATION FROM FILE '/ETC/OPT/IBM/V
SELFPD/CONFS-ENABLED/LFPD-TESTVSE.CONF'.
13:53:38 * MSG FROM VSEVIA : CONFIG_READCFGFILE: ERROR: CONFIGURATION PARAME
TER RESTRICT_TO_HOSTID IS ENABLED, BUT VSE_HOSTID6 IS NOT SET.
13:53:38 * MSG FROM VSEVIA : ERROR: CONFIGURATION COULD NOT BE LOADED OR IS
INCOMPLETE, FILE='/ETC/OPT/IBM/SELFPD/CONFS-ENABLED/LFPD-TESTVSE.CONF'.
13:53:38 * MSG FROM VSEVIA : STOPPING TO LOG THE STARTUP BECAUSE LFPD IS ABO
UT TO EXIT.
```

Adding RESTRICT_TO_HOSTID = no to the TESTVSE.LFPDCONF file solved the problem. You can use the **LFPD-CTL** command to restart the LFPD daemon after you make any configuration changes.

Now that the VIA user is configured, we can enable the VSE system for LFP.

4.4.3 Setting up VSE guest

Setting up the VSE guest system in a VIA environment is identical to the configuration that is done with LFP in a VM environment. Ensure that the IUCVDESTVMID is now set correctly to your VIA user, as shown in Example 4-19.

Example 4-19 Specifying the IUCVDESTVMID in the LFP start JCL

```
* $$ JOB JNM=LFPSTART,CLASS=A,DISP=D
// JOB LFPSTART START AN LFP INSTANCE
// EXEC IJBLFPOP,PARM='START DD:SYSIPT LOGALL'
ID                               = 01          <- SYSID of the LFP stack
```

```

MTU = 8192
IUCVMSGLIMIT = 1024
INITIALBUFFERSPACE = 512 K
MAXBUFFERSPACE = 4M
IUCVSRCAPPPNAME = TESTVSE <- must match the LFPD config file on D4C
IUCVDESTAPPPNAME = TESTVSE <- name of the LFP application on VSEVIA
IUCVDESTVMID = VSEVIA
WINDOWSIZE = 65535
WINDOWTHRESHOLD = 25
/*
/&
* $$ E0J

```

After the LFP instance is started, you should be able to query the VIA status. When you are querying the status of the LFP daemon from the control user, Example 4-20 shows that the TESTVSE application is connected to the VSE system.

Example 4-20 Displaying the LFP daemon status using SMSG

```

smsg vsevial app lfpd list
Ready; T=0.01/0.01 15:20:03
15:20:03 * MSG FROM VSEVIA : NAME          ENABLED  RUNNING  STATUS
15:20:03 * MSG FROM VSEVIA : -----
15:20:03 * MSG FROM VSEVIA : TESTVSE      YES      YES      CONNECTED TO TESTVSE

```

4.4.4 Using the LFP trace with VIA

By using the **LFPD-ADMIN** command, the communication between VSE and VIA can be traced. The command syntax is shown in Example 4-21.

Example 4-21 LFPD-ADMIN command syntax

```

SMSG <guest> APP LFPD-ADMIN <IUCVNAME> trace start FILENAME>(debug|packets|all)
(single|wrap) (maxsize)
SMSG <guest> APP LFPD-ADMIN <IUCVNAME> trace stop
SMSG <guest> APP LFPD-ADMIN <IUCVNAME> status

```

Before the trace is started, ensure that the VIA user linked the debug disk read/write and that the D4D disk is formatted as a CMS minidisk, as shown in Example 4-22.

Example 4-22 Accessing the data disk for LFP trace

```

DASD 0D4C 3390 35456R R/O          10 CYL ON DASD 6078 SUBCHANNEL = 000A
DASD 0D4D 3390 35456R R/W          10 CYL ON DASD 6078 SUBCHANNEL = 000B

```

Note: Only one trace can be started at one time.

The trace output is written to a CMS file whose name is specified in the trace start command. The name testvse is used in Example 4-23.

Example 4-23 Starting the LFP trace

```

SMSG vsevial APP LFPD-ADMIN testvse trace start testvse all single
Ready; T=0.01/0.01 16:23:39
16:23:40 * MSG FROM VSEVIA : TRYING TO CONNECT TO LFPD WITH IUCV APPLICATION
NAME TESTVSE...

```

```

16:23:40 * MSG FROM VSEVIA : CONNECTED.
16:23:40 * MSG FROM VSEVIA : ANSWER FROM LFPD:
16:23:40 * MSG FROM VSEVIA : -----
16:23:40 * MSG FROM VSEVIA : TRACE STARTED TO OUTPUTFILE /MNT/LFPD-TRACE/TESTVSE.LFPDTRC
16:23:40 * MSG FROM VSEVIA : -----

```

The trace is stopped by using the **trace stop** command, as shown in Example 4-24.

Example 4-24 Stopping the LFP trace

```

MSG vsevia APP LFPD-ADMIN testvse trace stop
Ready; T=0.01/0.01 16:29:38
16:29:38 * MSG FROM VSEVIA : TRYING TO CONNECT TO LFPD WITH IUCV APPLICATION
NAME TESTVSE...
16:29:38 * MSG FROM VSEVIA : CONNECTED.
16:29:38 * MSG FROM VSEVIA : ANSWER FROM LFPD:
16:29:38 * MSG FROM VSEVIA : -----
16:29:38 * MSG FROM VSEVIA : TRACE IS STOPPED.
16:29:38 * MSG FROM VSEVIA : -----

```

When you are reviewing the trace file from the configuration user (as shown in Example 4-25), it still appears to be empty.

Example 4-25 Checking the size of the trace file

Cmd	Filename	Filetype	Fm	Format	Lrec1	Records	Blocks	Date	Time
	TESTVSE	LFPDTRC	D1	V	-	0	0	8/29/11	15:23:40

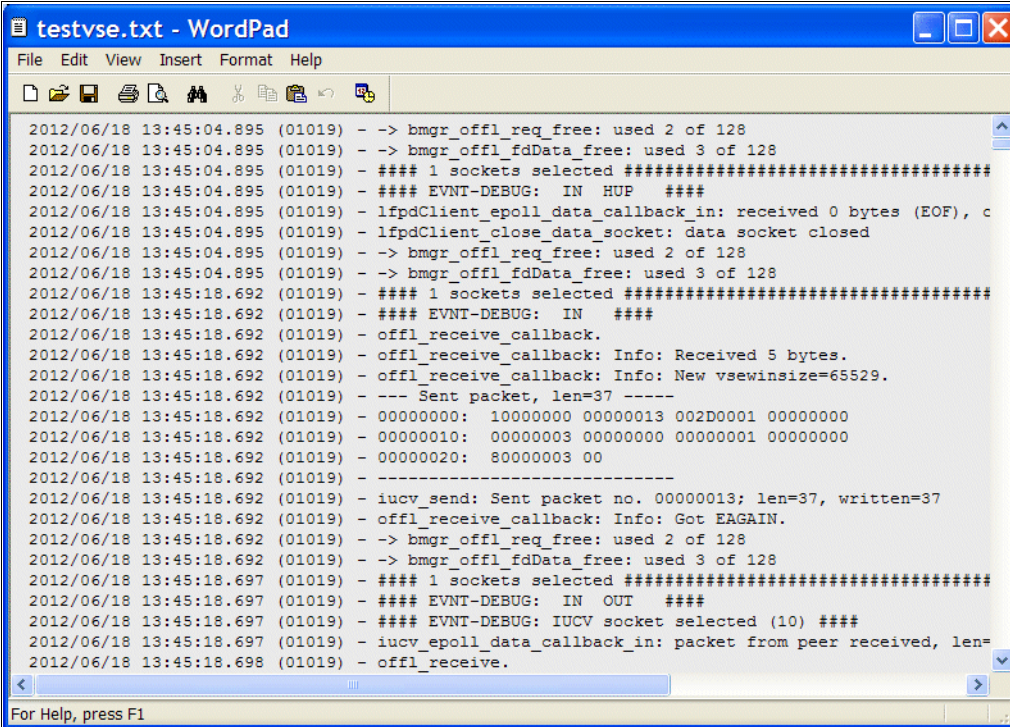
After the D4D disk is detached and relinked, the trace contents are available, as shown in Example 4-26.

Example 4-26 Accessing the trace file

Cmd	Filename	Filetype	Fm	Format	Lrec1	Records	Blocks	Date	Time
*	TESTVSE	LFPDTRC	D1	V	4096	20	20	8/29/11	15:29:38

The trace data is written in plain ASCII text; therefore, it cannot be read on z/VM. Download the trace file in binary to any ASCII platform, such as PC. The trace file contains UNIX style line breaks; therefore, when a Windows based system is used, you must use an editor that can correctly read and display UNIX style text files.

On Windows XP, you can use the WordPad editor, as shown in Figure 4-4.



The screenshot shows a WordPad window titled "testvse.txt - WordPad". The menu bar includes File, Edit, View, Insert, Format, and Help. The toolbar contains icons for opening, saving, printing, undo, redo, and other editing functions. The text area displays a log of system events and network activity, including timestamps, process IDs, and various callbacks and socket operations. The status bar at the bottom indicates "For Help, press F1".

```
2012/06/18 13:45:04.895 (01019) - -> bmgr_offl_req_free: used 2 of 128
2012/06/18 13:45:04.895 (01019) - -> bmgr_offl_fdData_free: used 3 of 128
2012/06/18 13:45:04.895 (01019) - #### 1 sockets selected #####
2012/06/18 13:45:04.895 (01019) - #### EVNT-DEBUG: IN HUP ####
2012/06/18 13:45:04.895 (01019) - lfpdClient_epoll_data_callback_in: received 0 bytes (EOF), c
2012/06/18 13:45:04.895 (01019) - lfpdClient_close_data_socket: data socket closed
2012/06/18 13:45:04.895 (01019) - -> bmgr_offl_req_free: used 2 of 128
2012/06/18 13:45:04.895 (01019) - -> bmgr_offl_fdData_free: used 3 of 128
2012/06/18 13:45:18.692 (01019) - #### 1 sockets selected #####
2012/06/18 13:45:18.692 (01019) - #### EVNT-DEBUG: IN ####
2012/06/18 13:45:18.692 (01019) - offl_receive_callback.
2012/06/18 13:45:18.692 (01019) - offl_receive_callback: Info: Received 5 bytes.
2012/06/18 13:45:18.692 (01019) - offl_receive_callback: Info: New vswinsize=65529.
2012/06/18 13:45:18.692 (01019) - --- Sent packet, len=37 -----
2012/06/18 13:45:18.692 (01019) - 00000000: 10000000 00000013 002D0001 00000000
2012/06/18 13:45:18.692 (01019) - 00000010: 00000003 00000000 00000001 00000000
2012/06/18 13:45:18.692 (01019) - 00000020: 80000003 00
2012/06/18 13:45:18.692 (01019) - -----
2012/06/18 13:45:18.692 (01019) - iucv_send: Sent packet no. 00000013; len=37, written=37
2012/06/18 13:45:18.692 (01019) - offl_receive_callback: Info: Got EAGAIN.
2012/06/18 13:45:18.692 (01019) - -> bmgr_offl_req_free: used 2 of 128
2012/06/18 13:45:18.692 (01019) - -> bmgr_offl_fdData_free: used 3 of 128
2012/06/18 13:45:18.697 (01019) - #### 1 sockets selected #####
2012/06/18 13:45:18.697 (01019) - #### EVNT-DEBUG: IN OUT ####
2012/06/18 13:45:18.697 (01019) - #### EVNT-DEBUG: IUCV socket selected (10) ####
2012/06/18 13:45:18.697 (01019) - iucv_epoll_data_callback_in: packet from peer received, len=
2012/06/18 13:45:18.698 (01019) - offl_receive.
```

Figure 4-4 Displaying the LFP trace

4.5 LFP in an LPAR environment

From a functionality perspective, LFP in LPAR is based on HyperSockets Completion Queue support (HSCQ).

Running an LFP in an LPAR environment means Linux on System z primarily runs on the cheaper, full-speed Integrated Facility for Linux (IFL) processors in one LPAR, whereas z/VSE runs on standard processors in a different LPAR.

Figure 4-5 shows the overall setup.

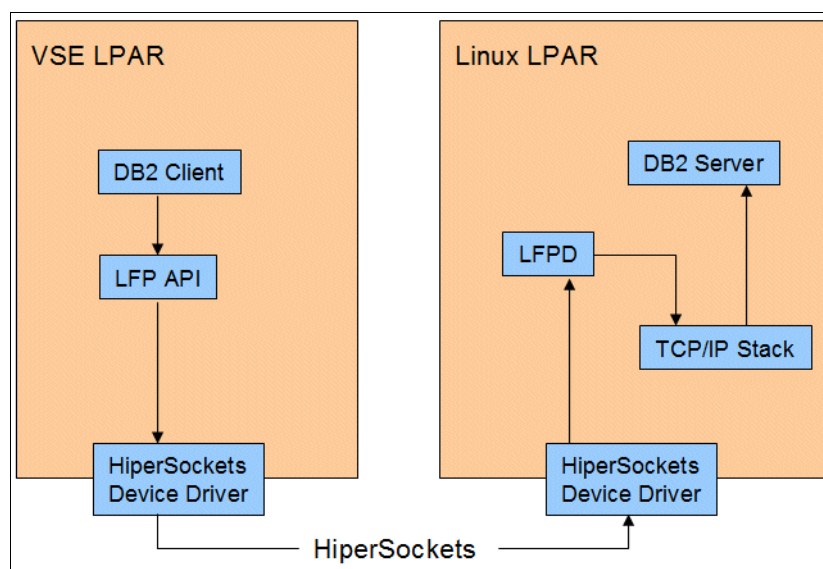


Figure 4-5 LFP in an LPAR environment

In this case, LFP starts the HiperSockets device driver to send the data to the Linux image. The HiperSockets Completion Queue function ensures successful data transmission. The Linux HiperSockets device driver forwards the data directly to the LFP daemon.

Note: The communication between z/VSE and Linux on System z is established directly by using HiperSockets without the involvement of the OSA device driver.

4.5.1 Prerequisites

The use of LFP in an LPAR environment includes the following prerequisites:

- ▶ A zEnterprise server at driver level 93 or later because of the HiperSockets Completion Queue function
- ▶ z/VSE Version 5 Release 1 with the following APARs/PTFs installed:
 - DY47300 / UD53758
 - PM56023 / UK76218
 - PM56056 / UK76252
- ▶ One of the following Linux on System z Operating Systems:
 - SUSE SLES 11 SP2 or later
 - Red Hat RHEL 7 or later
- ▶ One z/VSE system and one Linux on System z system that is running in LPAR mode

4.5.2 Hardware setup

To use LFP in an LPAR environment, you must define a HiperSockets network between the two LPARs. Use a separate network for LFP; do not use the same network for other TCP/IP stacks.

4.5.3 VSE setup

Configuring an LFP instance when it is running in LPAR mode is similar to z/VM mode. Table 4-3 shows the LPAR-specific parameters.

Table 4-3 LPAR-specific parameters for z/VSE

Parameter	Meaning
HSDevices	Specifies device addresses of a HiperSockets device. Must be defined with the device type OSAX in the IPL procedure.
HSSrcAppName	Defines a local port. Must be unique in combination with the HsSrcSystemName for the z/VSE system.
HSSrcSystemName	Defines the system name of the LFP instance. The name must be unique across the HiperSockets network that is used.
HSDestAppName	Must match the HS_SRC_APPNAME configuration parameter for an LFPD that is running on Linux.
HSDestSystemName	Must match the HS_SRC_SYSTEMNAME configuration parameter (HSUID) for an LFPD that is running on Linux.
HSKeepAliveTime	A decimal number that represents seconds. The default is 5 seconds. The keep alive mechanism is used to detect an unexpected termination of the connection between z/VSE and Linux.
HSMsgLimit	Specifies the maximum number of outstanding messages that are not yet received by the LFP instance. Higher values might result in a better performance, but need more memory. A reasonable value is 32.

Example 4-27 shows a sample setup that is taken from *z/VSE TCP/IP Support*, SC34-2640.

Example 4-27 Starting an LFP instance for LFP in LPAR

```
* $$ JOB JNM=LFPSTART,CLASS=0,DISP=L
// JOB LFPSTART
// EXEC IJBLFPOP,PARM='START DD:SYSIPT LOGALL'
* Instance ID
ID = 02
InitialBufferSpace = 1M
MaxBufferSpace = 4M
WindowSize = 65535
WindowThreshold = 25
DeviceType = HS
HSDevices = 500,501,502
HSMsgLimit = 128
HSSrcAppName = TESTV
HSDestAppName = LNXSYS1
HSSrcSystemName = VSELPAR
HSDestSystemName = LNXLPAR
HSKeepAliveTime = 30
/*
/&
* $$ E0J
```

4.5.4 Linux setup

Similar to the LFP setup for LPAR on the z/VSE side, there also are LPAR-specific parameters for the LFPD configuration for Linux, as shown in Table 4-4.

Table 4-4 LPAR-specific parameters for Linux

Parameter	Meaning
HS_MSGLIMIT	Specifies the maximum number of outstanding messages that are not yet received by LFPD. Higher values might result in a better performance, but need more memory.
HS_SRC_APPNAME	Defines a local port. Must be unique for the Linux on System z system. This parameter value must match the HSDestAppName that is used for an LFP instance on z/VSE.
HS_SRC_SYSTEMNAME	Defines the HSUID of the HiperSockets device that LFPD uses. Internally, the HSUID is converted to an IPv6 link-local address that is used to start the HiperSockets device. Multiple LFPDs can run on the same HiperSockets device, if each one uses a different HS_SRC_APPNAME. This parameter value must match with the HsDestSystemName that is used for an LFP instance on z/VSE.
PEER_HS_APPNAME	Defines the HsSrcAppName of an instance on z/VSE. If this parameter is set, LFPD checks the name of the source application of any incoming connection. If the name does not match the name in the configuration, the incoming connection is revoked.
PEER_HS_SYSTEMNAME	Defines the HsSrcSystemName of an instance on z/VSE. If this parameter is set, LFPD checks the name of the source system of any incoming connection. If the name does not match the name in the configuration, the incoming connection is revoked.

Example 4-28 shows a sample setup that is taken from *z/VSE TCP/IP Support*, SC34-2640.

Example 4-28 LFPD configuration for LPAR

```
DEVICETYPE = HS
HS_MSGLIMIT = 128
HS_SRC_APPNAME = LNXSYS1
HS_SRC_SYSTEMNAME = LNXLPAR
# ensure that only TESTV from VSELPAR can connect
PEER_HS_APPNAME = TESTV
PEER_HS_SYSTEMNAME = VSELPAR
MAX_SOCKETS = 1024
INITIAL_IO_BUFS = 128
WINDOW_SIZE = 65535
WINDOW_THRESHOLD = 25
VSE_CODEPAGE = EBCDIC-US
VSE_HOSTID = 10.0.0.1
RESTRICT_TO_HOSTID = no
LOG_INFO_MSG = no
```

Starting, stopping, and operating LFP in an LPAR environment is the same as for LFP in a z/VM environment.

4.6 IBM applications that support LFP

The following IBM applications can be used with LFP:

- ▶ VSE Connector Server
- ▶ CICS Web Support
- ▶ VSE Web Services (SOAP) support (client and server)
- ▶ CICS Listener
- ▶ DB2/VSE Server and Client
- ▶ WebSphere MQ Server and Client
- ▶ VSAM Redirector
- ▶ VSE VTape
- ▶ VSE LDAP Support
- ▶ VSE Script Client
- ▶ VSE/POWER PNET
- ▶ All applications that are included in the IPv6/VSE product
- ▶ DB Call Level Interface (DBCLI)
- ▶ z/VSE Monitoring Agent
- ▶ z/VSE SNMP Trap Client
- ▶ IBM Geographically Dispersed Parallel Sysplex™ (GDPS) Client

In addition to these applications, you can download tools that support LFP. For example, the z/VSE Virtual FTP daemon was developed especially for LFP.

Customer applications should run unchanged if they use one of the supported Socket API (Language Environment/C, EZA, or ASM SOCKET).

In the next sections, we describe the setup that is necessary for the use of the VSE Connector Server and z/VSE virtual FTP server.

4.6.1 VSE Connector Server

The only change to enable the VSE Connector Server for LFP is specifying the correct SYSID as assigned to the LFP stack, as shown in the following example:

```
// OPTION SYSPARM='01'
```

However, the STARTVCS job can contain a statement to check whether the TCP/IP stack is running, as shown in the following example:

```
*   WAITING FOR TCP/IP TO COME UP
// EXEC REXX=IESWAITR,PARM='TCPIP00'
```

Remove this statement when LFP is used. Starting the Connector Server by using the LFP stack is not apparent for any connector client application.

4.6.2 Using the Virtual z/VSE FTP Daemon

The Virtual z/VSE FTP Daemon is a Java application that provides FTP server functionality and uses the VSE Connector Server as the VSE back end. Therefore, this is a type of proxy solution. It requires that the VSE Connector Server is running on VSE and can connect to a VSE host only.

Note: The Virtual z/VSE FTP daemon can be used with all IP stacks, not with LFP only.

You can download the Virtual z/VSE FTP Daemon from the VSE home page, which is available at this website:

<http://www.ibm.com/systems/z/os/zvse/downloads/#vseftp>

After the Virtual z/VSE FTP Daemon is installed, you must edit its properties file (VirtualVseFtpServer.properties), which is in the server's installation directory. Here, you specify the IP address of the Linux system or VIA guest, as shown in the following example:

```
defaultVseHost=9.152.111.11
```

Before the FTP daemon is started, check whether the VSE Connector Server runs LFP-enabled on VSE.

In our VIA environment, we started the FTP daemon on a workstation, as shown in Figure 4-6. Therefore, the FTP goes to the local host first.

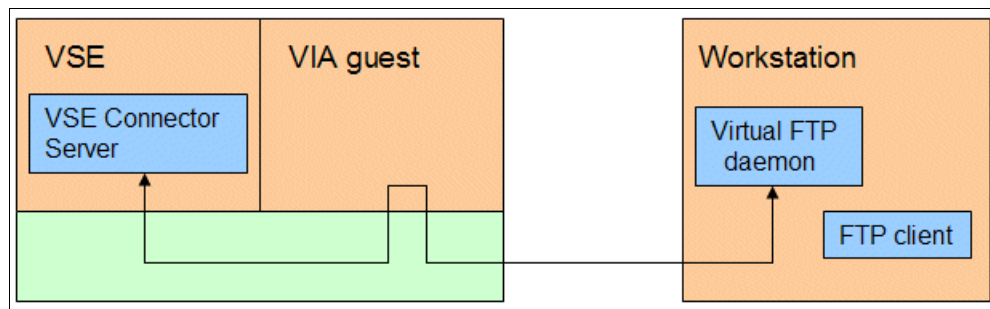


Figure 4-6 Using the z/VSE virtual FTP daemon for LFP

The FTP daemon then forwards the connection to the VIA user, which, in turn, uses IUCV or HiperSockets to get to the VSE system.

Example 4-29 shows some sample output of the FTP connect.

Example 4-29 FTP connect by using the Virtual z/VSE FTP Daemon

```
D:\>ftp 127.0.0.1
Connected to 127.0.0.1.
220 IBM Virtual z/VSE FTP Server on BR8ERNAN (version 1.0) ready to serve.
User (127.0.0.1:(none)): jsch
331 Password required for jsch.
Password:
230 User jsch logged in. Idle timeout is 15 minutes.
ftp> dir
200 PORT command successful.
150 ASCII data connection for / (127.0.0.1,2570).
drwxrwxrwx      1 vse      folder      0 Aug 29 14:52 ICCF
drwxrwxrwx      1 vse      folder      0 Aug 29 14:52 LIBR
drwxrwxrwx      1 vse      folder      0 Aug 29 14:52 POWER
drwxrwxrwx      1 vse      folder      0 Aug 29 14:52 VSAM
226 ASCII transfer complete.
ftp: 261 bytes received in 0.00Seconds 261000.00Kbytes/sec.
ftp>
```

Note: When a Linux on System z is used for running the LFPD instead of the pre-configured VIA image, you install the FTP Daemon on this Linux for best performance. In this case, the FTP goes to the Linux IP, as shown in Figure 4-7.

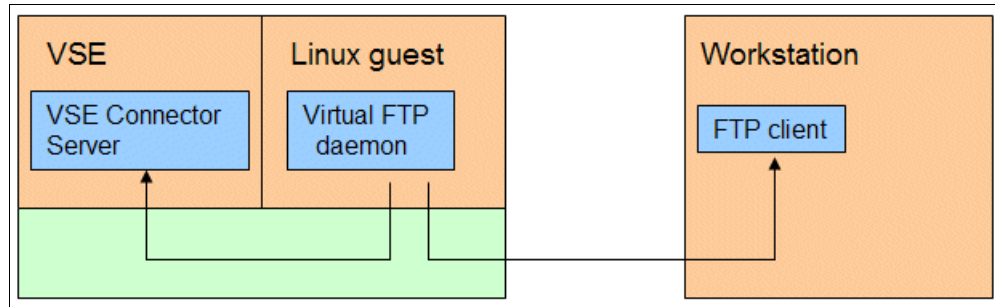


Figure 4-7 Running the Virtual z/VSE FTP Daemon server on Linux on System z

On Windows, the virtual FTP daemon can be started as a Windows service. For more information, see the server's online help.

4.7 Using secure connections with SSL

The LFP stack does not provide an SSL implementation. To use SSL, you must have the CSI stack installed and running. LFP then uses the SSL implementation from CSI, but transfers the data by using the LFP stack.

4.7.1 Using a VIA guest

Figure 4-8 shows the data flow when VIA is used. In this case, the VIA guest serves as a proxy that forwards data between VSE and any remote platform. It is not possible to install any other applications in the VIA guest. In particular, it is not possible to run an FTP daemon there.

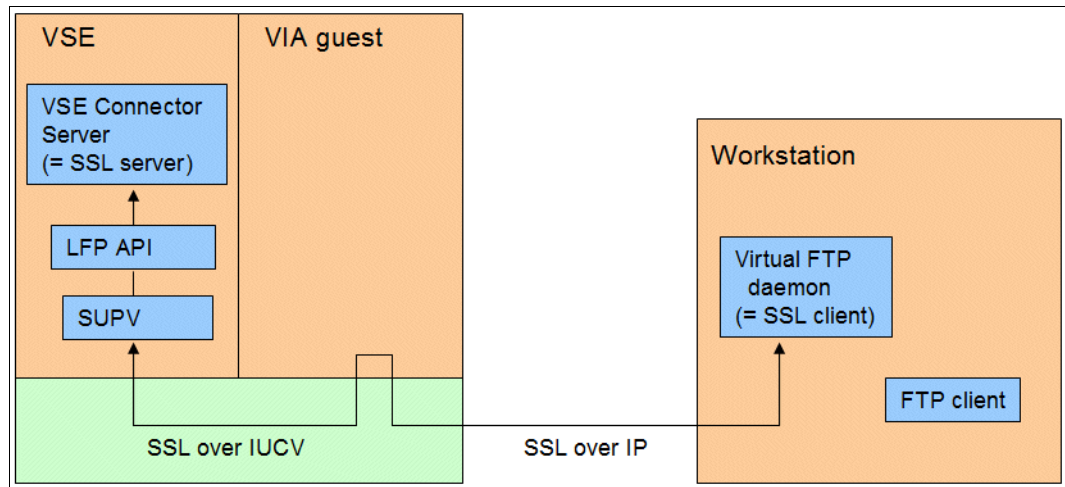


Figure 4-8 Using the z/VSE virtual FTP server in a VIA environment

In this example, the virtual FTP daemon runs on the same workstation as the FTP client. It acts as the SSL client and connects to the VSE Connector Server, which acts as the SSL server. The VIA guest, which acts as a proxy that forwards data to the VSE guest, is not visible to the two SSL end points. VIA forwards the encrypted data over the IUCV connection to VSE and vice versa. Therefore, SSL is configured between the workstation and VSE. The VIA guest is not visible.

4.7.2 Using a Linux on System z guest

Figure 4-9 shows the setup with a Linux on System z that is running the LFP daemon. In this case, the FTP daemon can be installed on the same Linux instance.

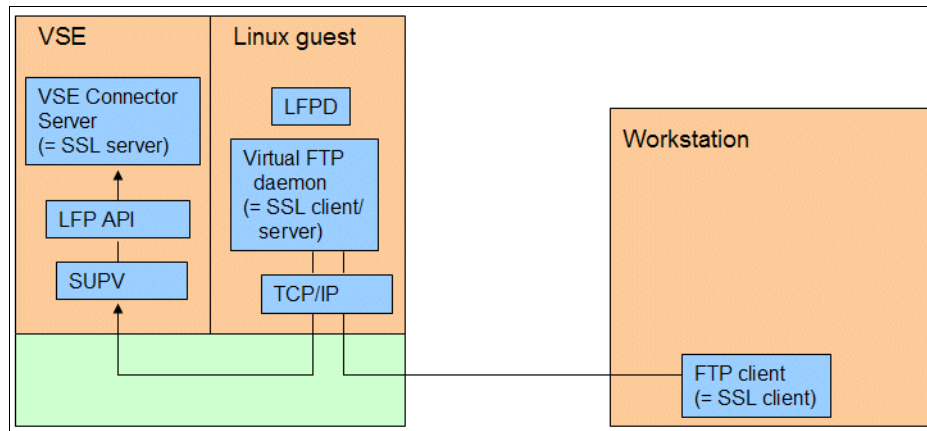


Figure 4-9 Running the z/VSE virtual FTP server on Linux on System z

In this example, the virtual FTP daemon is an SSL server and client at the same time. In the next section, we describe how to configure the virtual FTP daemon for both sides.

4.7.3 Configuring the z/VSE virtual FTP daemon for SSL

The Virtual z/VSE FTP daemon can be configured by using one of the following methods:

- As an SSL client with multiple SSL servers
 - Use the `HostAliases.properties` file to specify the SSL properties for one or multiple SSL servers (for example, VSE Connector Servers).
- As SSL server
 - Use the `VirtualVseFtpServer.properties` file to configure the SSL server properties.

Figure 4-10 shows the overall scenario.

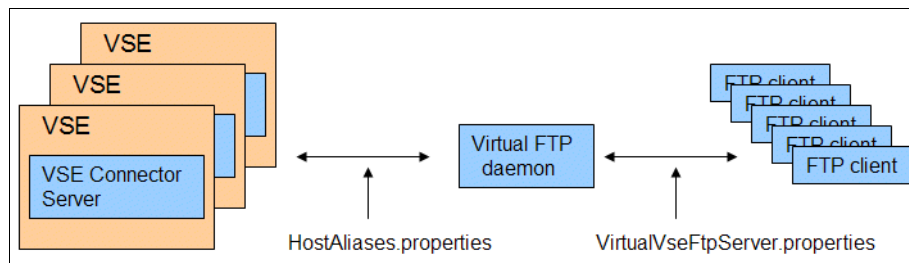


Figure 4-10 Scenario with Virtual z/VSE FTP daemon

For more information about configuring SSL with the z/VSE Virtual FTP daemon, see the FTP daemon's online help. Open the `index.html` file that is in the `/doc` directory within the server's installation directory.

For more information about setting up SSL with the VSE Connector Server and creating the necessary keys and certificates, see *Security on IBM z/VSE*, SG24-7691.

4.8 Known problems

In this section, we describe issues that we experienced and insights that we had during our test setup.

4.8.1 Error accessing the config disk

An error in accessing the config disk was reported.

Symptom

```
acc d4c c
DMSACP112S C(D4C) device error
```

Possible reason

The disk is not formatted. Try **format d4c c**.

4.8.2 SE file transfer had a problem

An error in the SE file transfer was reported.

Symptom

```
Following messages appear during the VIA boot process:
100.584937$ Wait: DIAG 0x2C4 RETURNED BAD STATUS_CODE=0x11.
100.584942$ VMSE: SE file transfer had a problem !
100.584946$ VMSE: Could not get container size.
```

Reason

VIA attempted to download a file from the SE that is needed only for debugging. This is not an error. Everything should work as expected.

4.8.3 User ID not authorized for SMSG

An error in the user ID not authorized for SMSG was reported.

Symptom

```
smsg vsevia app lfpd list
Ready; T=0.01/0.01 14:38:33
  14:38:33 * MSG FROM VSEVIA : YOUR USER ID IS NOT AUTHORIZED TO CALL THIS
FUNCTION.
```


Reason

You did not add your configuration user ID to the SENDERS.ALLOWED file on the configuration disk.

4.8.4 Invalid command response from VIA user

An error concerning an invalid command response from the VIA user was reported.

Symptom

Apparently, the LFPD command is not recognized. Although the command is entered correctly, the following command syntax is displayed:

```
smsg vsevia app lfpd startdbg testvse
Ready; T=0.01/0.01 14:00:01
14:00:01 * MSG FROM VSEVIA : USAGE: SMSG <GUEST> APP LFPD (START|STOP|RESTART|FORCE-RELOAD|STATUS|STARTDBG) <IUCVNAME>
14:00:01 * MSG FROM VSEVIA : SMSG <GUEST> APP LFPD (LIST|STARTALL|STOPALL)
```

Possible reason

The VIA user cannot link the configuration disk D4C. In our case, the disk was linked read/write by the configuration user, but the VIA user also was configured to link it read/write.

4.8.5 No response from VIA user

An error concerning receiving no response from the VIA user was reported.

Symptom

There is no response from the VIA user on an SMSG command.

Possible reason

It is likely that there is a typographical error in the called LFP script, so that you are, in fact, calling a non-existent script, as shown in the following example:

```
SMSG vsevia APP LFP-ADMIN2
Ready; T=0.01/0.01 17:12:19
```

The LFP script is called by using the Linux uevent mechanism. SMSG sends the string APP LFPD-ADMIN2 to the VIA guest. In this example, the uevent mechanism calls a script that is named LFPD-ADMIN2, which, in this case, does not exist. Normally, an entry is written to the Linux system log, but this is not visible to the VSE user.

4.8.6 Profile cannot be loaded

An error in loading the profile was reported.

Symptom

At the end of the VIA boot process, the following messages are displayed:

```
fpcConfigurationExpander(704): Sorry: cannot receive file 'profiles/LFP.json' from  
SE as '/tmp/LFP.json' under VM  
fpcConfigurationExpander(704): Could not load file profiles/LFP.json from SE. RC=2
```

Possible reason

You are using the incorrect name for the LFP profile. The correct name is zVSE-LFP.json. Change the VM directory entry and try again, as shown in the following example:

```
* Network adapters and configuration  
LOADDEV SCPDATA '{"profiles":["zVSE-VIA"],"networkCards":[{"OSA"',  
': "0D00","staticIPv4":"9.152.111.11"}],"defaultGateway":',  
'"9.152.108.1","DNS":["9.152.120.241","9.152.64.172"]',  
',"hostName":"vsevia"}'
```



OpenSSL

This chapter describes the new z/VSE SSL support that is based on OpenSSL and the benefits for customers and vendors. It outlines the functional parts that are common among z/VSE and other platforms and shows the differences between them.

Special issues, such as creating random numbers and keystores, also are described. We also show how the IPv6/VSE product from IBM and Barnard Software, Inc., uses OpenSSL to provide network security for their applications.

This chapter includes the following topics:

- ▶ Overview
- ▶ Access to the OpenSSL API
- ▶ Creating random numbers
- ▶ Keystore considerations
- ▶ Programming
- ▶ Performing the OpenSSL speed test
- ▶ How OpenSSL is used on z/VSE
- ▶ OpenSSL vulnerabilities
- ▶ Considerations on TLSv1.2
- ▶ Considerations about Diffie-Hellman
- ▶ Using DHE-RSA with OpenSSL on z/VSE
- ▶ Considerations on Elliptic Curve Cryptography
- ▶ Using ECDHE-RSA with OpenSSL on z/VSE
- ▶ Restrictions

5.1 Overview

OpenSSL is an open source project that provides an SSL implementation and key management utilities. OpenSSL is written in C and is available for many operating systems and hardware platforms.

OpenSSL is ported to z/VSE for several reasons. It provides SSL for those IP stacks that lack their own SSL implementation: IPv6/VSE and Linux Fast Path (LFP). Also, vendors that already are providing SSL for their stack can now offer their customers another alternative; for example, TCP/IP for VSE/ESA.

For more information about OpenSSL, see this website:

<http://www.openssl.org/>

5.1.1 What is available on z/VSE

The following OpenSSL-based SSL runtime environment is provided since z/VSE 5.1 as part of a new system component:

VSE CryptoServices
5686-CF9-17
CLC=51S (z/VSE 5.1) or 52S (z/VSE 5.2)

The new z/VSE Cryptographic Services component is installed in PRD1.BASE and includes the following features:

- ▶ IJBSSL phase (the SSL runtime component, which is based on OpenSSL)
- ▶ SPEEDTST phase (starts the built-in OpenSSL speed test)
- ▶ NOTICES.Z (license information)
- ▶ IJBVLVSE.OBJ (provides access to the APIs and must be linked to your application)
- ▶ IJBSSL.H (provides function prototypes)

Notes: The initial OpenSSL version on z/VSE 5.1 was 1.0.0.d with APAR DY47397 (PTF UD53864).

An update is provided with APAR DY47472 (PTF UD53952), which removes RC4-based SSL cipher suites because of known security issues.

APAR DY47499 (PTF UD53983) was upgraded OpenSSL 1.0.1e.

Check the Security section on the VSE home page for further OpenSSL updates:

<http://www.ibm.com/systems/z/os/zvse/support/preventive.html#security>

On z/VSE 5.1, OpenSSL on z/VSE requires being explicitly supported by an IP stack through linking the IJBVLVSE.OBJ to the stack code. This is done by the IPv6/VSE product from Barnard Software, Inc.

On z/VSE 5.2, OpenSSL on z/VSE can also be used transparently by using the new SSLPHASE parameter in the LE/C Multiplexer EDCTCPMC. Refer to section 5.7, “How OpenSSL is used on z/VSE” on page 173 for details about configuring the SSL phase.

5.1.2 What is unique in z/VSE

The following unique features are available only in z/VSE:

- ▶ A z/OS-compatible SSL programming interface

This API was described in *z/OS Cryptographic Services, SSL Programming*, SC24-5901 and was adapted for z/VSE in *z/VSE TCP/IP Support*, SC34-2640. It is used by all existing SSL applications on z/VSE, such as CICS Web Support, VSE Connector Server, and WebSphere MQ for z/VSE. Wrapping the native OpenSSL functions by this z/OS SSL API allows existing z/VSE SSL applications to run unchanged with OpenSSL.

- ▶ Support for IBM System z cryptographic hardware

Although OpenSSL can perform all encryption algorithms with all key lengths in software, performance is dramatically improved by using hardware crypto support. Hardware functionality also can be used that is not available in software, such as hardware-based generation of random numbers.

5.1.3 Runtime variables

There are two variables for controlling the behavior of OpenSSL on z/VSE. The use of crypto hardware can be turned on and off by using the SSL\$ICA parameter, as shown in the following example:

```
// SETPARM SSL$ICA = [ 'YES' | 'NO' ]
```

The debug trace can be controlled by using the SSL\$DBG variable, as shown in the following example:

```
// SETPARM SSL$DBG = [ 'YES' | 'NO' ]
```

Note: These variables can be used only when the SSL application uses the GSK interface. This is the case for IPv6/VSE from Barnard Software, Inc.

5.1.4 What is not available in z/VSE

The following functions are not available in z/VSE 5.1:

- ▶ The OpenSSL command-line tool. Therefore, key management is done on a workstation (Windows, Linux, and so on). Created keystores are then uploaded to z/VSE. Keystores can be created by using OpenSSL on a workstation or by using the Keyman/VSE utility that can be downloaded from the z/VSE home page.
- ▶ Other command line-based functions, such as encryption of files and certificate verification. Encryption of z/VSE files and tapes is provided by Encryption Facility for z/VSE.
- ▶ Some algorithms are not available on z/VSE because of legal reasons: IDEA, RC5, MDC2.
- ▶ Non-LE/C applications. Because OpenSSL is coded in C, an LE/C-runtime environment is required for callers.

5.2 Access to the OpenSSL API

Because each VSE phase has only one main entry point that can be called after a CDLOAD, a special mechanism is established to provide access to the OpenSSL API functions.

The IJBSSLVSE.OBJ file is provided for accessing the API functions in phase IJBSSL. Consider the following points when you are using this file:

- ▶ It must be linked to any OpenSSL application on VSE.
- ▶ It performs a CDLOAD of the IJBSSL phase.
- ▶ It provides access to OpenSSL functions by obtaining the function pointer by calling the OpenSSL main entry point with a specific function ID.
- ▶ Unfortunately, the list of callable functions cannot be dynamic; that is, each called function must be contained in the list. For a complete list, see “SSL APIs” on page 221.

A counterpart of IJBSSLVSE is linked to phase IJBSSL. This counterpart implements the main entry point of IJBSSL and returns the function pointer of the requested OpenSSL API function by using fetchep().

Figure 5-1 shows how an IP stack accesses the OpenSSL API by using module IJBSSLVSE.

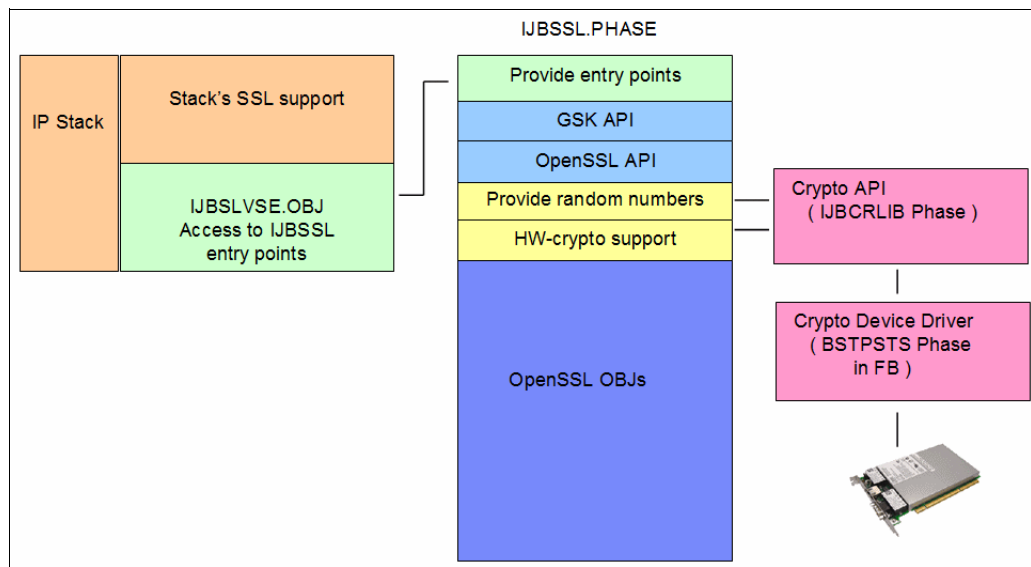


Figure 5-1 Access to the OpenSSL API

The IP stack uses IJBSSLVSE to access the native OpenSSL API or the IBM GSK API. Phase IJBSSL uses a low-level z/VSE crypto API that is provided by phase IJBCLIB to perform cryptographic functions on hardware.

5.3 Creating random numbers

Creating random numbers is a sensitive task in every crypto system. There are various random number generators on the different hardware platforms and operating systems. In the following sections, we describe how OpenSSL creates random numbers on z/VSE and which z/VSE interfaces are used.

5.3.1 Characteristics of random number generators

An optimal random number generator has the following characteristics:

- Probability

Random numbers are independent of each other and all possible numbers have the same probability of being generated. That is, each bit of each number has the probability of 0.5 of being '1' or '0'.

- Unpredictability

A sequence of numbers must not be predictable. Looking at any part of a random number sequence, no previously generated nor any number to be generated in future can be derived from this part.

- Performance

The generator should provide high output rates at low latency.

- Security

There must be no chance for any attacker to influence the generation process.

Random number generators can be placed into the following categories:

- Hardware-based generators
- Software-based generators
- A mixture of both

Hardware-based generators best fulfill all four requirements, primarily being unpredictable and secure. Software-based generators are deterministic and need some unpredictable seed value to create a sequence of numbers that is usually the same for a seed. Mixed generators try to overcome the disadvantages of pure software-based generators; however, they do not reach the unpredictability and security level of hardware-based generators.

For these reasons, hardware-based generators are often called true random number generators (TRNGs) and all other types are called pseudo random number generators (PRNGs).

Examples of a TRNG are the IBM Crypto Express adapters, while the PRNG function of the CPACF feature is a PRNG.

5.3.2 Random number generation in OpenSSL

When you review the original OpenSSL code, there are various source modules that use platform-dependent functionality for providing random numbers for any supported operating system. In the OpenSSL distribution, these source files are in `/crypto/rand`.

A new source module is provided for VSE when porting OpenSSL to VSE. It uses the API that is provided by phase IJBCRLIB to obtain random bytes. This function is also used by Encryption Facility for z/VSE for creating encryption keys. Phase IJBCRLIB, in turn, uses the z/VSE crypto device driver to access any available crypto hardware.

5.3.3 Alternatives

In z/VSE 5.1 and later, random numbers can be obtained by using one of the following methods:

- ▶ Using a crypto coprocessor card (PCIXCC, CEX2C, CEX3C, or CEX4C)

This is the best possibility because a crypto card returns true random numbers that are independent of any seed value. The cards provide built-in statistic tests transparently checking the random number quality. It is possible to obtain 8 KB (8192 bytes) of random output per request.

- ▶ Using the CPACF-provided pseudo random number generator (PRNG)

The PRNG function is a mixture of a hardware-based generator and the need of a software-provided seed value. It is up to the user to obtain this seed value from various sources of randomness in the system. The PRNG function is available on System z10 and later. It needs a 32-byte input parameter block and some clear text with the same length as the wanted number of random bytes.

- ▶ Using a pure software-based random number generator

This is the weakest alternative and is used only if the other methods are unavailable. It has deficiencies in probability, predictability, and security.

Phase IJBCRLIB transparently checks for hardware availability and always uses the best possible random number generator that is available on your system.

Example 5-1 is an extract from the IJBSSL trace that shows the available random number generators and which generator was used. Apparently, there was no Crypto Express coprocessor card available and the random bytes were generated by using CPACF.

Example 5-1 Extract from IJBSSL trace that shows the used random number generator

```
-> ijbslcry.c, get_bytes, num=32
Crypto lib not yet initialized ...
Check for CPACF and CEX2C ...
Using FUNC_RANDOM_GEN ...
rc from IJBCRLIB : 0
used generator   : 2
highest available : 2
(3 = CEXnC, 2 = CPACF/PRNG, 1 = SW)
<- ijbslcry.c, get_bytes, rc=0, outlen=32
...
```

5.3.4 Considerations for the z/VSE crypto device driver

The z/VSE crypto device driver runs as subtask IJBCRYPT as part of the Basic Security Manager (BSM) security server (SECSESV), which runs by default in partition FB. It controls access to any installed crypto cards.

For more information about configuring crypto cards and the z/VSE crypto device driver, see *Security on IBM z/VSE*, SG24-7691, which is available at this website:

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

The crypto device driver includes the following performance relevant parameters:

- The polling time interval. This is the time interval in 1/300 fractions of seconds to check for a reply from a crypto card. The default value is 1/300 of a second. Specifying zero minimizes the elapsed job time, but increases CPU cycles. You can specify the polling time interval with the APWAIT command, as shown in the following example:

```
msg fb,data=apwait=n
```

- The AP interrupts setting. With z10 BC/EC or later, crypto cards can notify a caller when an enqueued crypto request is processed and is available in the card's output queue. This removes the need for polling. AP interrupts are not supported by z/VM. Therefore, this is possible only when z/VSE is run in an LPAR. By default, AP interrupts are disabled. The commands to enable and disable AP interrupts are APEAI (enable) and APDAI (disable), as shown in the following example:

```
msg fb,data=[apeai | apdai]
```

For more information about the commands, see *z/VSE Administration*, SG34-2627, which is available at this website:

<http://www.ibm.com/systems/z/os/zvse/documentation/#vse>

There is no general statement possible about which setting, polling, or interrupts is better for your environment. Tests showed that this depends on the type of workload.

Note: If you are using an External Security Manager (ESM), such as TopSecret or Alert, you can run the crypto device driver separately as phase IJBCRYPT in any static or dynamic partition. For more information about supporting an ESM, see *z/VSE Administration*, SG34-2627.

5.3.5 Performance

Performance tests that use a Crypto Express3 coprocessor card on a z10 EC with z/VSE 5.1 that is running under z/VM showed the influence of the crypto device driver polling time interval and internal card processor burden when different amounts of random numbers were requested. As this system ran under z/VM, AP interrupts were not supported. We left the polling time interval at 1/300 second.

Figure 5-2 shows the performance results that generated random output from 128 bytes up to 8 KB per request.

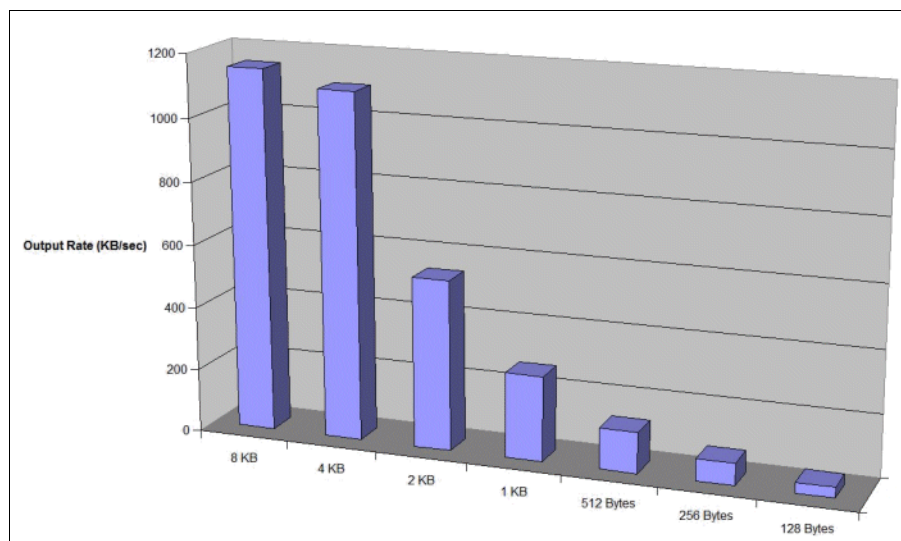


Figure 5-2 Random number generation performance with Crypto Express3

The results show that the card works optimally when it is requesting 4 - 8 KB random bytes per request because this minimizes the number of calls.

5.4 Keystore considerations

Before any SSL application can be run, you need a keystore that contains the RSA public/private key pair and the SSL certificates. Two types of keystores are available.

The following keystore formats are relevant for z/VSE:

- PFX

The Personal Information Exchange (PFX) format was defined by RSA Security and conforms to the PKCS#12 standard. PFX files can contain multiple keys and certificates and usually are password-protected. PFX files are supported by web browsers, such as Microsoft Internet Explorer and Mozilla Firefox. Sometimes, the file extension p12 is also used for the PFX format.

- JKS

The Java keystore (JKS) format is provided by Oracle and is the standard keystore format for Java applications. JKS files are mostly protected by a password. JKS files often cannot be handled by web browsers. Part of each Java installation is the `keytool.exe`, which also can be used for maintaining JKS keystores. In a VSE environment, JKS files are used by the VSE Connector Client.

- VSE keyring members

This type of keystore was created by Connectivity Systems International. It consists of three VSE library members with the same member name, but different member types: PRVK (contains the RSA key pair), ROOT (contains the SSL root certificate), and CERT (contains the SSL server certificate). Although other formats store all keys and certificates in one file, the CSI keyring format uses one separate VSE library member for each item. You can optionally protect the PRVK member with a custom passphrase by cataloging a CIALEXIT phase.

► PEM

The Privacy-enhanced mail (PEM) format is used by OpenSSL, but also by applications, such as FileZilla server. A PEM file can contain an RSA key pair, an SSL certificate, or both. It might be password protected. Although other keystore formats are binary only, the PEM file contents is base-64 encoded.

In the following sections, we describe how PEM keystores can be used on VSE with OpenSSL.

5.4.1 Creating a PEM keystore

Creating a PEM file is described in 3.7.2, “Creating the keystore” on page 71. Use Keyman/VSE because this is the simplest way to create a PEM file for use on z/VSE.

5.4.2 Exporting PEM to PFX

By using OpenSSL, the certificates in a PEM file can be exported into a PFX file, which can then be imported, for example, into web browsers, as shown in the following command:

```
openssl pkcs12 -export -out mycert.pfx -in mycert.pem
```

During this process, you are prompted to specify an export password. Earlier versions of Keyman/VSE did not allow reading a PFX file without a password. With Keyman/VSE, build January 2013 and later, this ability is now supported.

5.4.3 Importing PFX to PEM

By using the following command, you can import the certificates from a PFX file into a PEM file:

```
openssl pkcs12 -in mycert.pfx -out mycert.pem
```

During the process, you are prompted to specify an import password and a PEM passphrase. The import password is the password that was specified when the PFX was created with Keyman/VSE or the PFX was exported from another PEM.

Note: When you are editing the created PEM file, you see that it contains some information lines between the certificate and key data. We removed these lines before the PEM file was sent to VSE.

5.4.4 Password-protected keystores

By using OpenSSL, you can protect the key in a PEM file by using a passphrase. The use of a passphrase makes the key more secure; however, each server or client that is using this PEM must specify this passphrase every time it is started. If you do not want a password, specify the `-nodes` parameter (no DES encryption) when you are creating the PEM file. When this parameter is included, the private key is not encrypted.

Because only the key is protected by the passphrase, a PEM file can be created with a protected key. Certificates can be added later by editing the file by using a text editor.

The following command does not specify the `-nodes` parameter. Therefore, it creates a PEM with a password protected private key:

```
openssl req -x509 -days 365 -newkey rsa:4096 -keyout rsa4096p.pem -out  
rsa4096p.pem
```

The encrypted key is now indicated in the PEM file by BEGIN/END ENCRYPTED PRIVATE KEY, as shown in Figure 5-3.

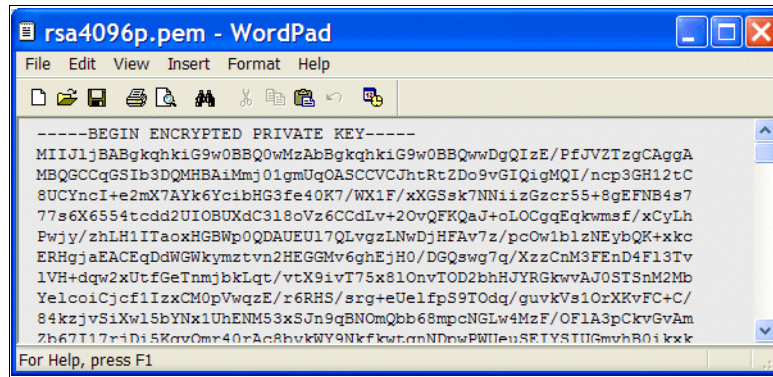


Figure 5-3 PEM file with an encrypted private key

OpenSSL requires a password length of 4 - 511 characters. Private keys and encrypted private keys cannot be processed further by Keyman/VSE.

Note: You should not use a PEM passphrase because IPv6/VSE does not support password protected PEM files.

5.5 Programming

By using one of the following methods, you can write z/VSE applications by using OpenSSL:

- ▶ The use of the native OpenSSL API

The OpenSSL API is described on the OpenSSL website (<http://www.openssl.org/>) and consists of several hundred functions. On z/VSE, a subset of this API is provided by IJBSLVSE.OBJ that you link to your application to access the functions in the IJBSLVSE phase. For more information about supported native OpenSSL functions, see “OpenSSL API” on page 222.

- ▶ Using the z/OS SSL API

This method is the suggested method to use OpenSSL on z/VSE. The API is described in Chapter 9 of *z/OS Cryptographic Services, SSL Programming*, SC24-5901. The C data structures are defined in the include file `gskssl.h`, which is part of LE/C and included with PRD2.SCEEBASE. The object file IJBSLVSE must be linked to your application when the z/OS SSL API is used.

Although this API is now outdated on z/OS and should not be used by new z/OS applications, it is still the SSL API for z/VSE. All existing VSE SSL applications use this API because TCP/IP for VSE/ESA provides this exact API.

For more information about an API description of the GSK functions, see “z/OS SSL API” on page 221.

The following sections provide details about the use of the z/OS SSL API and switching between the two APIs. There is much literature on using the OpenSSL API on the Internet, including the following website:

<http://www.ibm.com/developerworks/opensource/library/l-openssl/index.html?ca=dat>

5.5.1 Include files

The following SSL-related include files are provided with VSE:

- ▶ `sslvse.h`: Shipped with the CSI SSL-implementation in the TCP/IP `lib.sublib`
- ▶ `gskssl.h`: Shipped with the LE/C socket interface in `PRD2.SCEEBASE`

OpenSSL and OpenSSL applications must be compiled against `gskssl.h` because function prototypes in `sslvse.h` are defined with `#pragma linkage OS`, which causes errors in passing parameters to the GSK functions. Even more important is the fact that OpenSSL cannot have any dependency to vendor software.

As a consequence of the use of `gskssl.h`, the GSK API functions use return codes that are described in the z/OS API. These return codes are different from the codes that are used by the CSI SSL implementation. For example, the VSE Connector Server is compiled against `gskssl.h`, but there might be customer applications that use the CSI include file.

5.5.2 Passed socket number

The socket number as passed by a gsk-application (`int s`) is not necessarily the socket number only. VSE applications, such as CICS Web Support (CWS), VSE Connector Server, and WebSphere MQ pass a pointer to a private struct that contains the socket somewhere in the struct.

Example 5-2 shows how to pass the socket number as part of a C structure.

Example 5-2 Using a C structure for passing the socket number

```
typedef struct _mysock {
    int     somefield;
    char*   someptr;
    int     socketno;
} MYSOCK;

#pragma linkage (skread,OS)
int skread(int s, void * data, int len)
{
    int rc;
    MYSOCK* mys = (MYSOCK*)s;
    rc = recv(mys->socketno, data, len, 0);
    return(rc);
}
```

Routine `skread` uses this C structure to access the socket number. The application then specifies the function pointers of `skread` and `skwrite` in the `sk_soc_init_data` structure, as shown in Example 5-3.

Example 5-3 gsk_soc_init_data structure

```
typedef struct _gsk_soc_init_data { /* Secure soc init data      */
    int     fd;                  /* file descriptor        */
}
```

```

gsk_handshake hs_type;          /* client or server handshake */
char * DName;                   /* keyring entry Distinguished */
                                /* name. When NULL, the default */
                                /* keyring entry is used */
char * sec_type;                /* Type of application */
                                /* 0 CLIENT */
                                /* 1 SERVER */
                                /* 2 SERVER_WITH_CLIENT_AUTH */
                                /* 3 CLIENT_NO_AUTH */
char * cipher_specs;            /* SSLV2 not used by VSE */
char * v3cipher_specs;         /* SSLV3 cipher suites */
int (* skread)                  /* User supplied READ routine */
    (int fd, void * buffer, int num_bytes);
int (* skwrite)                 /* User supplied WRITE routine */
    (int fd, void * buffer, int num_bytes);
unsigned char cipherSelected[3]; /* SSLV2 not used by VSE */
unsigned char v3cipherSelected[2]; /* Cipher Spec used */
int failureReasonCode;         /* failure reason code */
gsk_cert_info * cert_info;     /* This information is read from */
                                /* from the client certificate */
                                /* when client authentication is */
                                /* enabled */
gsk_init_data * gsk_data;      /* Pointer to init data */
} gsk_soc_init_data;

```

When the function pointers are specified, fetchep must be used, as shown in the following example:

```

typedef void (*FETCH_PTR)(int);
init_data.skread = (int (*)(int,void*,int))fetchep((FETCH_PTR)skread);
init_data.skwrite = (int (*)(int,void*,int))fetchep((FETCH_PTR)skwrite);

```

The reason for this is that the skread and skwrite functions are implemented by the application, but are called from phase IJBSSL.

5.5.3 Socket calls

When the GSK API is used, the caller specifies a read and a write callback routine that is used for sending data over the socket. This means that the SSL component never accesses the socket directly. It also means that the GSK API has no way of using other socket calls, such as givesocket, takesocket, and ctrl.

Unfortunately, OpenSSL performs socket calls. The low-level socket functions are defined in *eos.h*, where for VSE these #defines are used, as shown in Example 5-4.

Example 5-4 Low-level socket functions

```

#define get_last_socket_error()  errno
#define clear_socket_error()    errno=0
#define ioctlsocket(a,b,c)      ioctl(a,b,c)
#define closesocket(s)          close(s)
#define readsocket(s,b,n)       read((s),(b),(n))
#define writesocket(s,b,n)      write((s),(b),(n))

```

This causes OpenSSL to use the application-specified read/write routines (skread/skwrite). As of this writing, we do not have a solution or other socket calls because the GSK API considers these two routines only.

The GSK API implementation, in turn, implements the `vse_readsocket` and `vse_writesocket` routines and calls the application-specified read/write routines, as shown in Example 5-5.

Example 5-5 Specifying socket call functions

```
int (*skread)(int s, void* b, int n);
int (*skwrite)(int s, void* b, int n);

int vse_readsocket(int s, void* b, int n)
{
    return skread(s, b, n);
}

int vse_writesocket(int s, void* b, int n)
{
    return skwrite(s, b, n);
}
```

5.5.4 Switching between GSK and OpenSSL socket calls

So that applications can use both APIs (GSK and native OpenSSL) dynamically, `gsk_initialize()` sets a global variable `ssl_use_gsk_callbacks` in Phase IJBSSL, which causes the OpenSSL code to use the GSK callbacks. There are only two OpenSSL modules (`bssconn.c` and `bsssock.c`) that perform socket calls. The following change is made for VSE:

```
if (ssl_use_gsk_callbacks)
    ret=vse_readsocket(b->num,out,outl);
else
    // original openssl code ...
```

The global variable is reset in `gsk_uninitialize()`.

The following sections describe some z/VSE specific aspects, such as how to specify the name and location of your keyring file and the list of SSL ciphers to be used by your application.

5.5.5 Specifying the key ring

Depending on whether you uploaded the PEM file as a VSE library member or as a VSAM file, there are two ways to specify the keyring in a GSK application. In both cases, the keyring location is specified when the `gsk_initialize()` function is called and the keyring name is specified when the `gsk_secure_soc_init()` function is called.

When an empty string for `lib.sublib` is specified, the GSK wrapper expects the keyring label (DName) to be a VSAM file name. When valid values are specified for `lib` and `sublib`, the keyring label is handled as a VSE library member name. The member type must be PEM in this case.

Keyring type Librarian

For Librarian type keyrings, a VSE library and sublibrary are specified in the `gsk_initialize()` call, as shown in Example 5-6.

Example 5-6 Specifying a Librarian type keyring

```
char * keyring = "CRYPTO.KEYRING";
gsk_init_data init_data;
...
init_data.keyring = keyring;
rc = gsk_initialize(&init_data);
```

The member name of the PEM file is specified in the `gsk_secure_soc_init()` call as the `DName` parameter, as shown in Example 5-7. The member type PEM is implicitly assumed.

Example 5-7 Specifying the DName for a Librarian-type keyring

```
gsk_soc_data * socdata;
gsk_soc_init_data sock_init_data;
...
sock_init_data.DName = "MYKEY"; // VSE library member name
socdata = gsk_secure_soc_init(&sock_init_data);
```

Keyring type VSAM

For VSAM type keyrings, specify an empty string for parameter “keyring” in the `gsk_initialize()` call, as shown in Example 5-8.

Example 5-8 Specifying a VSAM type keyring

```
char * keyring = "";
gsk_init_data init_data;
...
init_data.keyring = keyring;
rc = gsk_initialize(&init_data);
```

The keyring name (`DName`) as specified in the `gsk_secure_soc_init()` call is then considered to be a VSAM file label. As shown in Example 5-9, the `DName` variable specifies the VSAM file label.

Example 5-9 Specifying the DName for a VSAM-type keyring

```
gsk_soc_data * socdata;
gsk_soc_init_data sock_init_data;
...
sock_init_data.DName = "MYKEY"; // VSAM file label
socdata = gsk_secure_soc_init(&sock_init_data);
```

5.5.6 Using a password-protected keyring

As described in 5.4.4, “Password-protected keystores” on page 159, a PEM file can be password protected. When password-protected keyrings are used, each client or server must specify the PEM passphrase when the keystore is accessed.

When the GSK API is used, you specify the keyring password in the `gsk_init_data` structure when `gsk_initialize()` is called, as shown in Example 5-10.

Example 5-10 Specifying a keyring password

```
char * keyring = "";
gsk_init_data init_data;
...
init_data.keyring = keyring;
init_data.keyring_pw = "ssltest";
rc = gsk_initialize(&init_data);
```

OpenSSL on z/VSE now makes the following assumptions:

- ▶ The PEM file was created on an ASCII platform; therefore, the password is ASCII-encoded.
- ▶ The GSK application specifies the password in EBCDIC.

Therefore, before the related OpenSSL function for opening the PEM file is called, the password is translated to ASCII.

Note: Depending on the password characters, there might be an issue with the EBCDIC to ASCII translation. As of this writing, it is not possible to specify code pages.

5.5.7 Supported cipher suites

OpenSSL provides more cipher suites than are available with TCP/IP for VSE/ESA. Similar to TCP/IP for VSE/ESA, OpenSSL allows keywords, such as `DEFAULT`, `ALL`, and `HIGH`. Hardware support is available only for cipher suites that use the RSA algorithm for the SSL handshaking, AES, DES, or Triple-DES for encryption, and SHA-1, SHA-224, and SHA-256 as the hash algorithm.

An application that uses the GSK API must specify a list of hex codes as was done when the CSI implementation was used. The list of hex codes is then internally translated into an OpenSSL-readable list of cipher suites. For more information, see 5.5.8, “Specifying cipher suites” on page 166 for details.

Table 5-1 shows the list of cipher suites that are recommended for use with OpenSSL and the CSI SSL-implementation. Refer to section 5.10, “Considerations about Diffie-Hellman” on page 176 for setting up Diffie-Hellman and section 5.12, “Considerations on Elliptic Curve Cryptography” on page 184 for setting up Elliptic-Curve-based cipher suites.

Table 5-1 Recommended SSL cipher suites

Hex code	OpenSSL notation	TCP/IP for VSE/ESA notation
0A	DES-CBC3-SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA
2F 35	AES128-SHA AES256-SHA	TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA
3C 3D	AES128-SHA256 ^a AES256-SHA256 ^a	-

Hex code	OpenSSL notation	TCP/IP for VSE/ESA notation
15 16 33 39 67 6B	EDH-RSA-DES-CBC-SHA ^b EDH-RSA-DES-CBC3-SHA ^b DHE-RSA-AES128-SHA ^b DHE-RSA-AES256-SHA ^b DHE-RSA-AES128-SHA256 ^b DHE-RSA-AES256-SHA256 ^b	-
C011 C012 C013 C014 C027	ECDHE-RSA-RC4-SHA ^c ECDHE-RSA-DES-CBC3-SHA ^c ECDHE-RSA-AES128-SHA ^c ECDHE-RSA-AES256-SHA ^c ECDHE-RSA-AES128-SHA256 ^c	-

a These cipher suites belong to TLSv1.2 and require OpenSSL 1.0.1e or later.

b These cipher suites require the setup of DH parameters when VSE is the server.

c These cipher suites require the setup of DH parameters and an EC key when VSE is the server.

Note: For more information about OpenSSL cipher suites, see this website:

<http://www.openssl.org/docs/apps/ciphers.html>

For a description of the cipher suites, see RFCs 2246 and 3268.

5.5.8 Specifying cipher suites

Your C program specifies the list of cipher suites as required by the GSK API. The list that is shown in Example 5-11 is then translated into an OpenSSL readable form.

Example 5-11 Specifying the SSL cipher suites

```
char * ciphers = "2F350A09";
...
sock_init_data.v3cipher_specs = ciphers;
...
socdata = gsk_secure_soc_init(&sock_init_data);
```

In this example, the following string is passed internally to OpenSSL:

AES128-SHA:AES256-SHA:DES-CBC3-SHA:DES-CBC-SHA

5.5.9 Supported RSA key lengths

On VSE, the RSA key length is limited to 4096 bits because this is current upper limit of what is supported by crypto cards. OpenSSL can process RSA keys up to 16384 bits. For VSE, the change in the include file `rsa.h` (as shown in Example 5-12) was made.

Example 5-12 Limiting the RSA key size to 4096 bits

```
#ifndef OPENSSL_RSA_MAX_MODULUS_BITS
//# define OPENSSL_RSA_MAX_MODULUS_BITS 16384
# define OPENSSL_RSA_MAX_MODULUS_BITS 4096
#endif
```

Removing this restriction and the use of software-based encryption does not make sense because software performance is limited when compared to the use of cryptographic hardware. For more information about OpenSSL performance, see 5.6, “Performing the OpenSSL speed test” on page 170.

5.5.10 Debugging

OpenSSL has different built-in debug capabilities. Debugging is enabled by recompiling the sources with related compiler debug switches. However, the SSL data structure (see typedef `ssl_st` in OpenSSL include file `ssl.h`) has an int field “debug”, which is used for internal debugging purposes.

Using the GSK API

When the z/OS GSK API is used, you can use the JCL variable `SSL$DBG = [YES | NO]` to turn the debug trace on and off. The variable is evaluated in the `gsk_initialize()` function, so it is usable with the z/OS GSK interface only.

Using both APIs

On VSE, debugging can be turned on and off without recompiling any source modules by the following new functions that are provided in modules IJBVLVSE and IJBVLACC:

```
extern int debug=0;
void ssl_enable_debug(void);
void ssl_disable_debug(void);
```

Enabling debugging causes a static global variable `debug` to be set to 1; disabling debugging resets the value to zero. The two functions can be used by any VSE GSK or OpenSSL application.

Debug output from phase IJBSSL is printed to SYSLST when debugging is enabled. You might consider enabling your GSK or OpenSSL application for debugging by providing a `DEBUG` parameter in the `PARM` string when your application is called, as shown in the following example:

```
// EXEC MYAPP,PARM='DEBUG'
```

The application code reads the parameter and calls the related function to enable or disable debugging.

5.5.11 Hardware crypto support

Hardware crypto support is provided transparently for existing VSE SSL applications because they use the z/OS GSK API. Depending on which API your application uses, the following possibilities apply.

Using the GSK API

When the z/OS GSK API is used, you can use the JCL variable `SSL$ICA = [YES | NO]` to turn hardware crypto support on and off. The variable is evaluated in the `gsk_initialize()` function; therefore, it is usable with the z/OS GSK interface only.

Using both APIs

Hardware crypto support can be turned on and off without recompiling any source modules by the following new functions that are provided in modules IJBSLVSE and IJBSLACC:

```
extern int ssl_use_ibmca=1;
void ssl_enable_ibmca(void);
void ssl_disable_ibmca(void);
```

Enabling hardware crypto support causes a static global variable `ssl_use_ibmca` to be set to 1, calling `ssl_disable_ibmca()` resets the value to zero. The two functions can be used by any VSE GSK or OpenSSL application.

Consider providing a parameter `IBMCA` in the `PARM` string when your application is called, as shown in the following example:

```
// EXEC MYAPP,PARM='IBMCA'
```

Your application code reads the parameter and calls the related function to enable or disable hardware crypto support.

5.5.12 Programming example

This section provides an example of a C program that uses OpenSSL.

Include files

In addition to the standard header files, you must include the statements that are shown in Example 5-13.

Example 5-13 Includes

```
#include <stdio.h>
#include <features.h>
#include <socket.h>
#include <stdlib.h>
#include <types.h>
#include <in.h>
#include <inet.h>
#include <iocctl.h>
#include <iconv.h> // for ASCII to EBCDIC translation
#include <gskssl.h>
```

Include file `gskssl.h` is part of the LE/C run time and normally is in `PRD2.SCEEBASE`.

Prototypes

The function prototypes that are shown in Example 5-14 allow enabling and disabling the OpenSSL debug trace and hardware crypto support.

Example 5-14 z/VSE provided functions

```
void ssl_enable_debug(void);
void ssl_disable_debug(void);
void ssl_enable_ibmca(void);
void ssl_disable_ibmca(void);
```

Data types and variables

The declarations that are shown in Example 5-15 are necessary to use the C socket API.

Example 5-15 Declarations for using the C socket API

```
typedef struct sockaddr_in    SOCKADDR_IN;
typedef struct sockaddr      SOCKADDR;
```

Callback routines

When the GSK API is used, the caller specifies a read and a write routine that is to be used for sending data over the socket. This means that the SSL component never accesses the socket directly. It also means that the GSK API has no way of using other socket calls, such as `givesocket`, `takesocket`, and `ctrl`.

Unfortunately, OpenSSL performed socket calls. The low-level socket functions that are defined in OpenSSL include file `e_os.h` where the following `#defines` are used for VSE:

```
#define readsocket(s,b,n)    read((s),(b),(n))
#define writesocket(s,b,n)  write((s),(b),(n))
```

This causes OpenSSL to use the application-specified read/write routines (`skread/skwrite`). As of this writing, we do not have a solution or other socket calls because the GSK API considers these two routines only.

The socket number as passed by the application (`int s`) is not necessarily the socket number. VSE applications, such as CWS, VSE Connector Server, and WebSphere MQ, pass a pointer to a private struct that contains the socket somewhere in the struct.

To be flexible, you write your code as shown in Example 5-16 and put the socket number into a data structure.

Example 5-16 Handling the socket number in a user application

```
typedef struct _mysock {
    int      somefield;
    char*    someptr;
    int      socketno;
} MYSOCK;
```

Your callback functions then resemble the functions that are shown in Example 5-17.

Example 5-17 Socket callback functions

```
/* I/O routine to perform a read function for SSL VSE */
#pragma linkage (skread,OS)
int skread(int fd, void * data, int len)
{
    int rc;
    MYSOCK* mysocket = (MYSOCK*)fd;
    rc = recv(mysocket->socketno, data, len, 0);
    return(rc);
}

/* I/O routine to perform a write function for SSL VSE */
#pragma linkage (skwrite,OS)
int skwrite (int fd, void * data, int len)
{

```

```

int rc;
MYSOCK* mys = (MYSOCK*)fd;
rc = send(mysocket->socketno, data, len, 0);
return(rc);
}

```

The addresses of the two callback routines `skread()` and `skwrite()` are specified in the `gsk_soc_init_data` structure before `gsk_secure_soc_init()` is called, as shown in Example 5-3 on page 161.

5.6 Performing the OpenSSL speed test

OpenSSL provides a built-in speed test with which you can check the speed of your system when cryptographic algorithms are performed, such as RSA, AES, SHA, or DES. On z/VSE, this test is included in phase IJBSSL and can be started with JCL, as shown in the following statement:

```
// EXEC SPEEDTST,PARM='OPENSSL'
```

For a list of supported parameters, see the original OpenSSL documentation on the Internet (<http://www.openssl.org/>) or enter the following command on a workstation with OpenSSL installed:

```
# openssl speed ?
```

5.6.1 Test parameters

On other platforms, this test is started with the `speed` parameter of the `openssl.exe`, as shown in the following command:

```
# openssl speed
```

You can run the speed test with specific parameters, as shown in the following command:

```
# openssl speed rsa
```

This command performs the RSA tests only. On VSE, you specify the following parameters:

```
// EXEC SPEEDTST,PARM='OPENSSL RSA'
or simply
// EXEC SPEEDTST,PARM='RSA'
```

RSA includes RSA512, RSA1024, RSA2048, and RSA4096.

You can display the available speed test parameters on Windows, as shown in Example 5-18.

Example 5-18 Speed test parameters

```

C:\Program Files\OpenSSL-Win32\bin>openssl speed ?
Error: bad option or value
Available values:
mdc2      md4      md5      hmac      sha1      sha256    sha512    whirlpoolrmd160
idea-cbc  seed-cbc  rc2-cbc  bf-cbc
des-cbc   des-ede3  aes-128-cbc aes-192-cbc aes-256-cbc aes-128-ige aes-192-ige
aes-256-ige
camellia-128-cbc camellia-192-cbc camellia-256-cbc rc4
rsa512    rsa1024    rsa2048    rsa4096

```

```

dsa512  dsa1024  dsa2048
ecdsap160  ecdsap192  ecdsap224  ecdsap256  ecdsap384  ecdsap521
ecdsak163  ecdsak233  ecdsak283  ecdsak409  ecdsak571
ecdsab163  ecdsab233  ecdsab283  ecdsab409  ecdsab571
ecdsa
ecdhp160  ecdhp192  ecdhp224  ecdhp256  ecdhp384  ecdhp521
ecdhk163  ecdhk233  ecdhk283  ecdhk409  ecdhk571
ecdhb163  ecdhb233  ecdhb283  ecdhb409  ecdhb571
ecdh
idea      seed      rc2      des      aes      camellia  rsa      blowfish
Available options:
-engine e      use engine e, possibly a hardware device.
-ecp e         use EVP e.
-decrypt       time decryption instead of encryption (only EVP).
-mr            produce machine readable output.

```

On VSE, you omit the first parameter “speed” because this is implied by starting the test by using the SPEEDTST phase, as shown in the following examples:

```

// EXEC SPEEDTST,PARM='AES-128-CBC'
// EXEC SPEEDTST,PARM='SHA1'
// EXEC SPEEDTST,PARM='DES-EDE3 SHA256 AES-128-CBC'

```

In addition to these openssl speed test parameters, you can use the two VSE-specific parameters IBMCA and DEBUG. Although the use of DEBUG does not make sense in a speed test, IBMCA enables hardware crypto support, as shown in the following example:

```

// EXEC SPEEDTST,PARM='AES-128-CBC IBMCA'

```

5.6.2 Test results

We performed the speed test on a zEnterprise 196 (2817-729 51) with z/VSE 5.1 running in an LPAR. The following sections describe the performance test results for CPU Assist for Cryptographic Function (CPACF) and Crypto Express3.

CPACF results

The CPACF feature accelerates symmetric crypto algorithms, such as AES, DES, and TDES and hash functions, such as SHA.

AES is one of the most important algorithms for SSL. It is faster and more secure than the formerly used DES or Triple-DES algorithms. Figure 5-4 shows the AES results for different blocks of plaintext data. For each key length (128, 192, and 256 bits), five test runs were performed with data block sizes from 16 bytes up to 8 KB. The chart in Figure 5-4 shows the number of operations with a specific key length and block size can be performed per second.

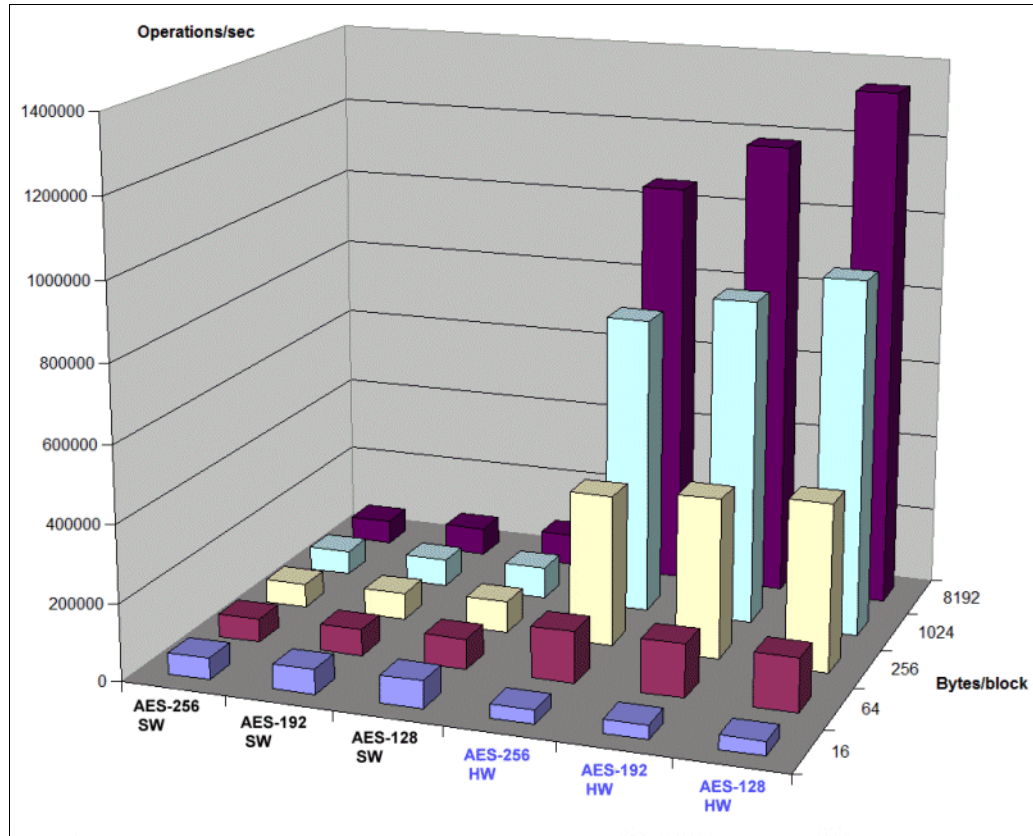


Figure 5-4 Software versus hardware performance results for AES

The results show that, for small blocks of data (less than 64 bytes), software performance is slightly better than hardware performance because of the processor burden of starting the CPACF function. However, this changes with increasing block sizes, where CPACF performs up to 16 times faster than software.

Crypto Express3 results

Crypto cards accelerate RSA operations that are part of each SSL handshake. Therefore, this pays off when there are workloads with many SSL handshakes in a certain time interval.

Examples are secure FTPs, Telnet sessions, and web browser connections to CICS Web Support. Figure 5-5 shows that RSA-1024 was performed 40 - 60 times faster when compared to software. This factor increases to 112 up to 122 for RSA-2048, and to over 300 for RSA-4096.

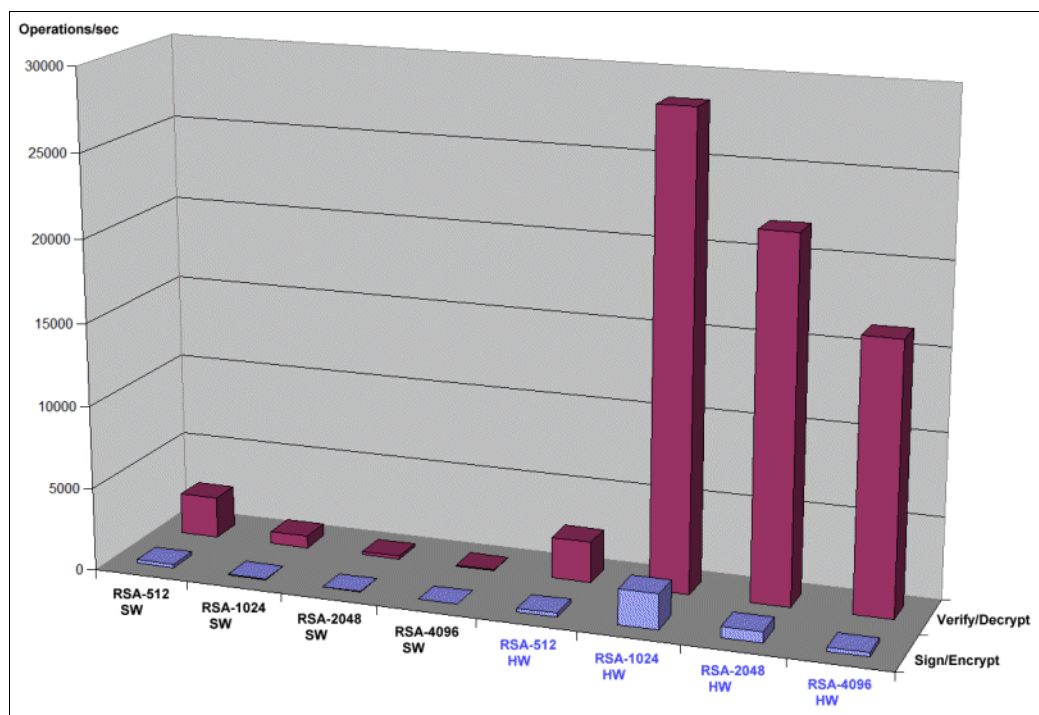


Figure 5-5 Software versus hardware performance results for RSA

On a z196, you need more than 1 second to create one signature with RSA-4096 when software is used. With a Crypto Express3 card, you can create 242 signatures per second for RSA-4096.

For RSA-512, the software and hardware results are equal because requests with this key length are not sent to a crypto card. Instead, they are always performed in software. In practice, RSA-512 was not used for decades, which tells us that without having Crypto Express adapters, it does not make sense to use RSA keys with a key length of more than 1024 bits.

5.7 How OpenSSL is used on z/VSE

In z/VSE 5.1, the only user for OpenSSL is the IPv6/VSE product from Barnard Software, Inc. (BSI), as of this writing. IPv6/VSE can be licensed through IBM or directly from BSI. The BSI stack provides two servers that provide SSL for VSE server and client applications. Both servers provide SSL/TLS transparently to the application. They also support batch and CICS applications that are written in any supported API, including applications that use the ASM SOCKET macro, EZASMI, EZASOKET, and LE/C APIs.

In z/VSE 5.2, an enhancement in the LE/C Multiplexer allows you to use any combination of IP stack (TCP/IP for VSE/ESA, IPv6/VSE, or LFP) and SSL implementation (OpenSSL, or the CSI-provided SSL implementation that is part of the CSI stack).

Skeleton EDCTCPMC in ICCF library 62 is an example of how to configure the use of TCP/IP socket and SSL APIs. The newly introduced parameter SSLPHASE allows specifying the phase implementing the SSL API independently from the phase implementing the TCP/IP socket API.

Example 5-19 shows how the new parameter is used:

- ▶ SYSID='00': TCP/IP for VSE/ESA using its own SSL implementation
- ▶ SYSID='01': Linux Fast Path using OpenSSL
- ▶ SYSID='02': IPv6/VSE using OpenSSL

However, every combination is possible.

The new parameter SSLPHASE is optional. If you omit it, the phase specified by the PHASE parameter is also assumed to implement the SSL API as shown in Example 5-19.

Example 5-19 LE Multiplexer example with new SSLPHASE parameter

```
* $$ JOB JNM=EDCTCPMC,CLASS=A,DISP=D,LDEST=*,PDEST=*
// JOB EDCTCPMC - GENERATE TCP/IP MULTIPLEXER CONFIG PHASE
// LIBDEF *,CATALOG=PRD2.CONFIG
// LIBDEF *,SEARCH=(PRD2.SCEEBASE,PRD1.BASE)
// OPTION ERRS,SXREF,SYM,NODECK,CATAL,LISTX
  PHASE EDCTCPMC,*,SVA
// EXEC ASMA90,SIZE=(ASMA90,64K),PARM='EXIT(LIBEXIT(EDECKXIT)),SIZE(MAXC
      -200K,ABOVE)'

EDCTCPMC CSECT
EDCTCPMC AMODE ANY
EDCTCPMC RMODE ANY
*
      EDCTCPME SYSID='00',PHASE='$EDCTCPV',SSLPHASE='$EDCTCPV'
      EDCTCPME SYSID='01',PHASE='IJBFLPLE',SSLPHASE='IJBSSLLE'
      EDCTCPME SYSID='02',PHASE='BSTTTCP6',SSLPHASE='IJBSSLLE'
*
      END

/*
// IF $MRC GT 4 THEN
// GOTO NOLINK
// EXEC LNKEDT,PARM='MSHP'
/. NOLINK
/*
/&
* $$ EOJ
```

For more information about how IPv6/VSE uses OpenSSL, see 3.7, “Setting up SSL” on page 71.

5.8 OpenSSL vulnerabilities

National Vulnerabilities Database (NVD) is the US government repository of standards-based vulnerability management data that uses the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. NVD includes databases of security checklists, security-related software flaws, misconfigurations, product names, and impact metrics.

For more information, see this website:

<http://nvd.nist.gov/home.cfm>

On the NVD home page, click **Vulnerabilities** and search for SSL. You see a list of items that you can check against the current z/VSE OpenSSL version.

A list of recent security issues and available fixes is also provided on the OpenSSL website:

<http://www.openssl.org>

Check the newsflash section for recent updates. Available security fixes are regularly provided for z/VSE as IBM PTFs. Check the security section on the VSE home page for latest information and available security-related PTFs:

<http://www.ibm.com/systems/z/os/zvse/support/preventive.html#security>

5.9 Considerations on TLSv1.2

As of this writing, TLSv1.2 is the newest SSL protocol version. It introduces new SSL cipher suites that use the SHA-256 hash algorithm instead of the SHA-1 function, which adds significant strength to the data integrity. For more information about TLSv1.2, see this website:

<http://tools.ietf.org/html/rfc5246>

The following new SSL cipher suites and their related hexadecimal values are available:

```
0x3B  TLS_RSA_WITH_NULL_SHA256
0x3C  TLS_RSA_WITH_AES_128_CBC_SHA256
0x3D  TLS_RSA_WITH_AES_256_CBC_SHA256
```

TLSv1.2 is supported with OpenSSL 1.0.1e, which is the current code level on z/VSE.

z/VSE needs TLSv1.2 for the following reasons:

- ▶ In *NIST Special Publication 800-131A*, dated January 2011, entitled “*Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*”, Table 9 shows that the use of the SHA-1 hash function is not allowed after December 31, 2013, except for non-digital signature applications.
- ▶ The IBM global security policy enforces all IBM products to be compliant with *NIST Special Publication 800-131*.

For more information and examples of how to set up and use TLSv1.2 with the IPv6/VSE product, see 3.7.15, “Using TLSv1.2” on page 110.

5.10 Considerations about Diffie-Hellman

The Diffie-Hellman (DH) key agreement method is an alternative to the traditional way of negotiating encryption keys during the SSL handshaking process using RSA. Diffie-Hellman does not provide authentication, and is therefore usually used together with an additional authentication mechanism, for example RSA. Diffie-Hellman is described in RFC 2631.

<http://www.ietf.org/rfc/rfc2631.txt>

In the following, we compare DH with RSA in terms how an SSL session key is created and exchanged, not how the communication partners are authenticated.

- ▶ The main advantage of DH over RSA is the fact that a session key is never sent over the network, and therefore provides “perfect forward secrecy” (PFS). With PFS, it is not possible to decrypt a recorded SSL session in future when the RSA private key potentially got compromised or broken.
- ▶ The main disadvantage of DH is its higher CPU consumption. Establishing an SSL session by using DH consumes approximately 30% more CPU than compared to RSA.

The next sections explain how exchanging session keys works using RSA and show how DH differs from RSA.

5.10.1 RSA

Exchanging the session key by protecting it with an RSA public key is the way SSL worked on z/VSE with TCP/IP for VSE/ESA, but also with IPv6/VSE based on OpenSSL. Figure 5-6 on page 177 shows the SSL session key exchange using RSA.

The client first contacts the server. Second, the server sends its public RSA key, wrapped into a digital SSL certificate that is signed with the server's private RSA key. In a third step, the client can verify the server's signature with the help of a certificate authority (CA). Now comes the weak point: In step 4 the client creates a random session key, encrypts it with the server's public key, and sends it to the server. Because of this, the session key is part of the SSL session data. Finally, in step 5, the server confirms using this session key and they can start transferring encrypted data.

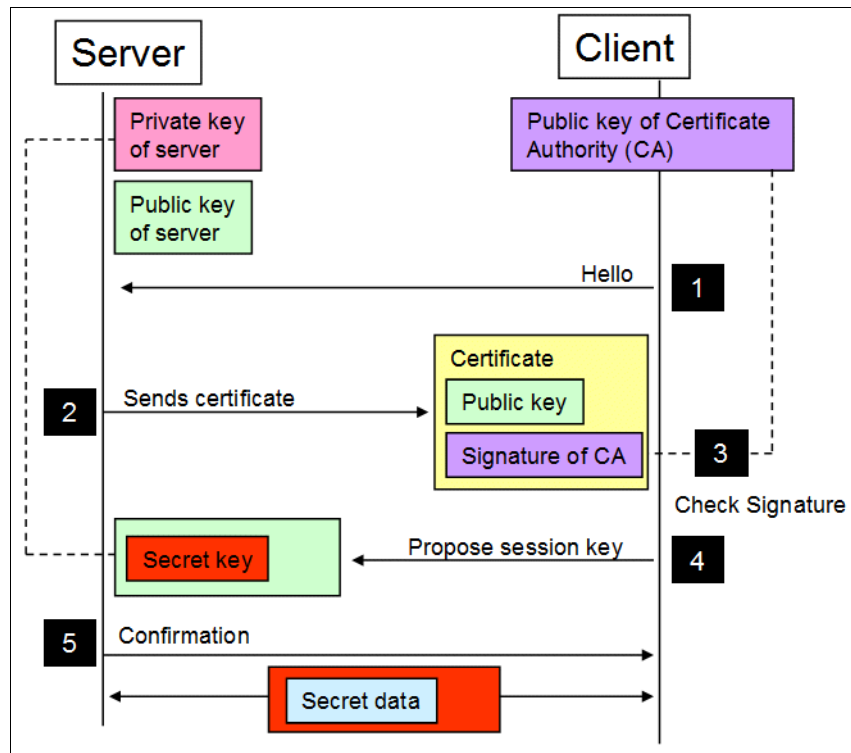


Figure 5-6 SSL session key exchange using RSA

The important point is that the secret session key is part of the SSL session data, because the key is sent from the client to the server, encrypted by using the server's public RSA key. If the server's private RSA key is ever compromised, stolen, or broken, the session key is no longer secure. It is then possible to decrypt and read the complete session data if the session was recorded and stored.

5.10.2 Diffie-Hellman

Using Diffie-Hellman, the session key is never sent over the network and is therefore never part of the network session data. Figure 5-7 shows how the session key is negotiated using DH. Therefore, we do not talk about “exchanging a session key”, but rather talk about “agreeing on a common session key” through the DH key agreement process.

Like we started before with RSA, the client contacts the server in a first step. The server now sends its Diffie-Hellman parameters to the client. These DH parameters consist of a large prime number (p) and a so called generator (g) where $0 < g < p$. For OpenSSL, g is always equal to 2. In a second step, both communication partners generate a random number in the range $\{1 \dots p-2\}$. These two random numbers are kept secret and not sent over the line. They are called the DH private keys. In a third step, both parties perform certain calculations to derive two values A and B , which are exchanged over the insecure medium. Both parties can now derive the same secret session key.

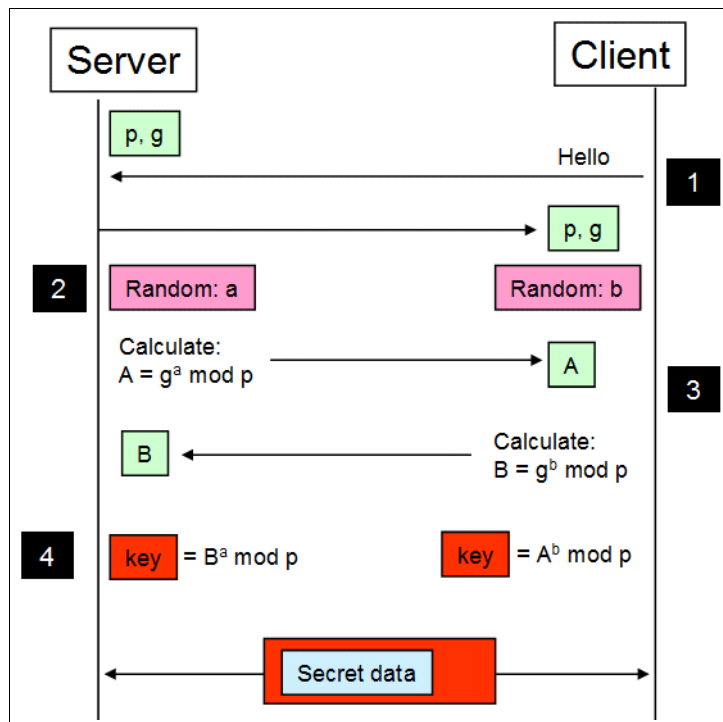


Figure 5-7 SSL session key agreement using Diffie-Hellman

The important point is that the encryption key is created independently on both sides. Therefore, it is not possible to unveil the session key from a given recorded network session later.

For simplification, Figure 5-7 does not show how the two communication partners are authenticated. In practice, DH is mostly used together with authentication by using RSA.

5.10.3 Variants of Diffie-Hellman

There are three main variants of Diffie-Hellman in SSL/TLS:

- Anonymous mode

Anonymous mode does not use authentication and is therefore vulnerable to man-in-the-middle attacks. You should not use anonymous Diffie-Hellman.

- Static mode

Static Diffie-Hellman reuses at least one of the two DH private keys (see Figure 5-7 on page 178) unchanged for all connections. When both DH private keys are reused, the term “static-static” is used. When only one side uses the same key, the term is “ephemeral-static”. In some implementations, it might make sense to have one static DH private key, especially on the server side, for performance reasons.

- Ephemeral mode

Ephemeral Diffie-Hellman generates a new temporary DH private key for every connection, which enables PFS. When both sides always create new DH private keys for new connections, this is called “ephemeral-ephemeral”.

On z/VSE, only ephemeral mode is supported, so the VSE side always creates a new DH private key for a new connection. It is not possible to provide a pre-generated DH private key to the API. The related SSL cipher suites are all prefixed with DHE-RSA. The next section shows how to set up and use DHE-RSA on z/VSE.

5.11 Using DHE-RSA with OpenSSL on z/VSE

With z/VSE 5.2 onwards, OpenSSL can be used together with all IP stacks on z/VSE by configuring the SSLPHASE by using the LE/C Multiplexer. It can still be used with IPv6/VSE and its utilities BSTTATLS and BSTTPRXY.

Note: DHE-RSA is supported by APAR DY47545.

The next sections show how to use the DHE-RSA-based SSL cipher suites in practical examples.

5.11.1 Generating DH parameters

The first task for setting up Diffie-Hellman is generating a set of DH parameters that consist of two numbers *p* (a large prime number) and *g* (the generator value, which is always 2 for OpenSSL). Parameter generation is CPU expensive, and is therefore normally done once in advance.

You have two options for generating the parameters:

- Using openssl on your workstation.
- Using the Keyman/VSE tool.

Based on these parameters, different temporary session keys are then created for your SSL connections.

Using openssl on your workstation

The following openssl command generates a .pem file that contains the new DH parameters:

```
openssl dhparam -out dhparam.pem 1024
```

The parameters are stored in Base64-encoded text form and look similar to Example 5-20.

Example 5-20 Sample DH parameters

```
D:\>type dhparam.pem
-----BEGIN DH PARAMETERS-----
```

```
MIGHAoGBANc1zZUH12RONYH5D4cIHcfM8ATuk75Ne02iaV3FhcAAfs9lj10uJaVn
UDH9qd19A4YrDi3VPm55r/YHA4v3wx42Xaq4Yfb1jeGOKfT6HuhIVS9/n3ZjwNFe
2IAJeiV4VCRAmjVrgZcUodpEK+jEH4tULNS3N03p6BbvU/6gyCQLAgEC
-----END DH PARAMETERS-----
```

To enable the DHE-RSA-based SSL cipher suites on VSE, just copy and paste this text form to the end of your .pem file.

Using the Keyman/VSE tool

The Keyman/VSE tool provides an even more convenient method of creating DH parameters and adding them to your .pem file.

Note: You need Keyman/VSE, build May 2014 or later for this function. You can download it from:

<http://www.ibm.com/systems/z/os/zvse/downloads/#vkeyman>

Keyman/VSE from May 2014 or later provides a new tool bar button for generating the DH parameters as shown in Figure 5-8.

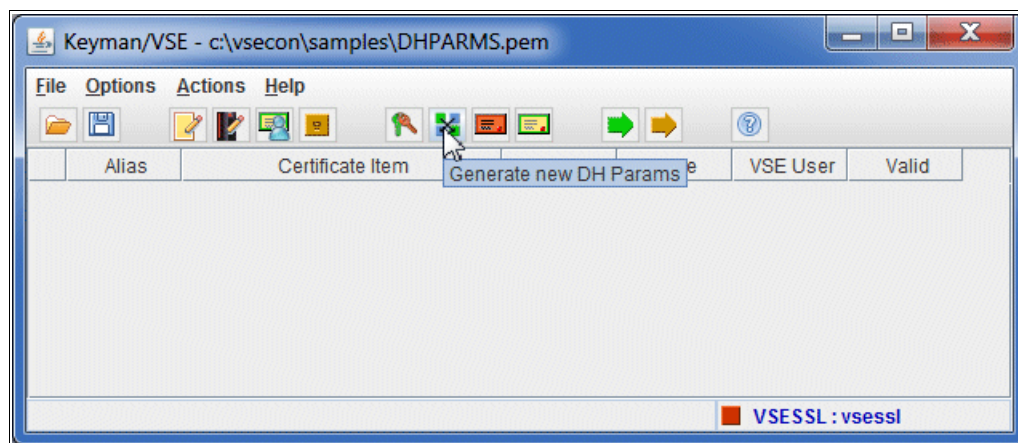


Figure 5-8 Generate DH parameters in Keyman/VSE

Clicking **Generate new DH Params** displays the dialog box shown in Figure 5-9.

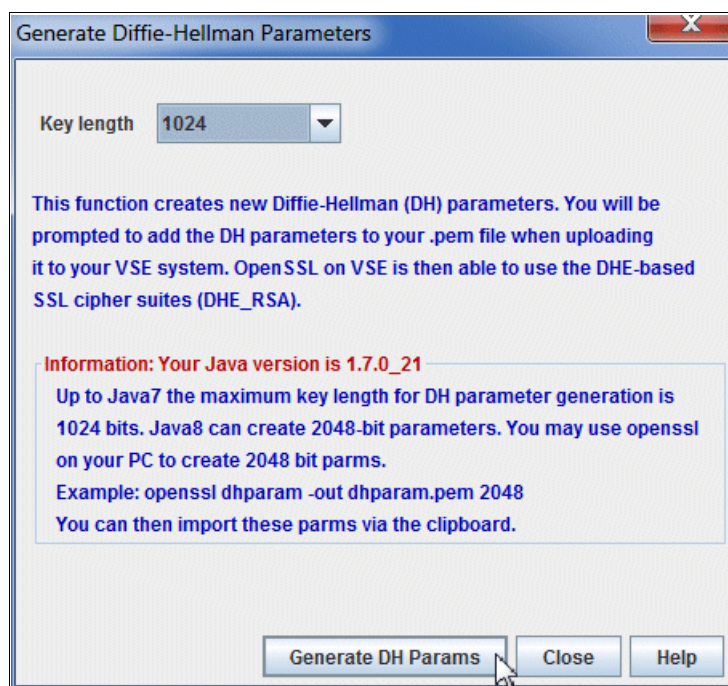


Figure 5-9 Generate Diffie-Hellman Parameters dialog box in Keyman/VSE

It is a restriction in Java that the maximum key length for DH parameter generation is 1024 bits for all Java versions before Java 8. If you need a longer key length, you must use either Java 8 or use openssl directly, as shown in “Using openssl on your workstation” on page 179, and import them by using the clipboard.

It does not matter whether you create (or import by using the clipboard) the DH parameters before or after the RSA key and your SSL certificates. Figure 5-10 shows a complete set of items: RSA key, CA root certificate, SSL server certificate, and DH parameters.

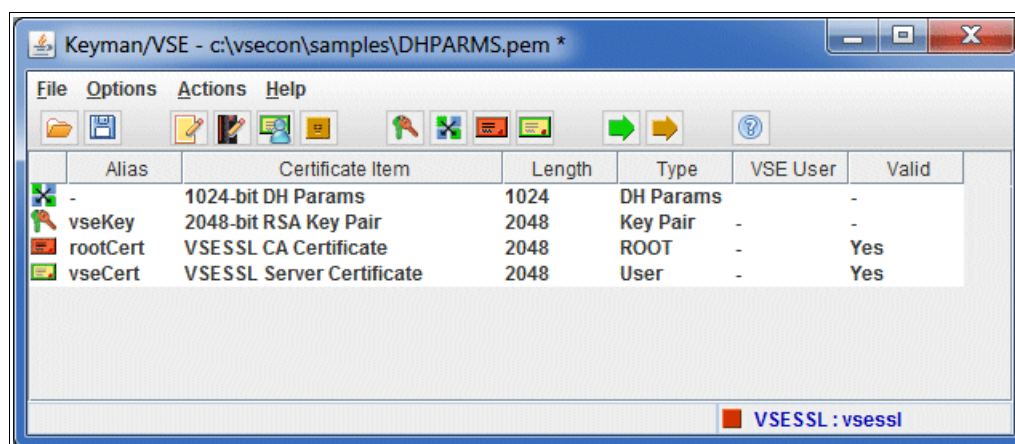


Figure 5-10 Complete set of items in Keyman/VSE

When uploading the .pem file to VSE, you now have the choice for including the DH parameters into your .pem file as shown in Figure 5-11. This enables the use of the DHE-RSA-based cipher suites on VSE. Select **File** → **Save as PEM file on VSE**.

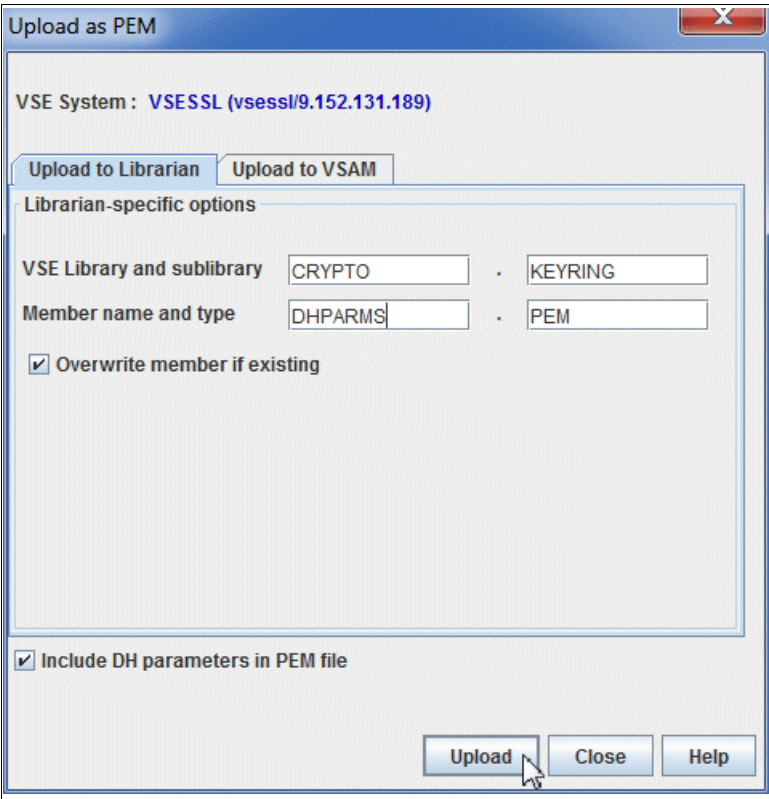


Figure 5-11 Upload PEM file to VSE

Clicking **Upload** uploads all items into a new .pem file on VSE. The .pem file can either be a VSE/Librarian member or a VSAM file. Make sure that the VSE Connector Server is started at this time. The DH parameters are added at the end of the .pem file on VSE as shown in Example 5-21.

Example 5-21 PEM file with DH parameters on VSE

DITTO/ESA for VSE				LE - Library Member Edit	
Member	DHPARMS.PEM	Library	CRYPTO.KEYRING	Col 1	Format CHAR
					SYSIPT data NO
					1...5...10...5...20...5...30...5...40...5...50...5...60...5...70..
00066	9w0BAQUFAA0CAQEAlWL15WIoIWGfS90yJaSJmcMKsUNUjXSMnH2cSm8jnIuCwCKb				
00067	vKBaY0eiX3QAJdW9zON068E7nDgEQ8IPrWz4b0j2EhffFPLZfNTwgX0iUj/AImhd				
00068	cqoRVaQgmWAjj6qMY9FZWnk6RNb310umEexZVPas35wIUI0ZtjrXAhp0c9nMWrd				
00069	4QVnc7Nx4JYK1Z1h8mKdc5UeSCpdIfa/+OnMXw9SbjoYt9Hm2LpbqaQr0D3Z6Ur				
00070	9FUaUSUcnB00YLeUi05iWofy4p2A3E0j4nuEIch+wwrf4E7GwoeniE/wCAAGiwXg				
00071	dJa01PAL2QLudmDs94L2Rvg0pV36cBLE10XmJw==				
00072	-----END CERTIFICATE-----				
00073	-----BEGIN DH PARAMETERS-----				
00074	MIGHAoGBANclzZUH12RONYH5D4cIHcfM8ATuk75Ne02iaV3FhcAAfs9lj10uJaVn				
00075	UDH9qd19A4YrDi3VPm55r/YHA4v3wx42Xaq4YfbljeG0KfT6HuhIVS9/n3ZjwNFe				
00076	2IAJeiV4VCRAmJVrgZcUodpEK+jEH4tULNS3N03p6BbvU/6gyCQLAgEC				

```
00077 -----END DH PARAMETERS-----
00078 **** End of data ****
```

Note: You must include DH parameters in your .pem file **only** if VSE is the server. When VSE is the client (for example, as LDAP or FTP client), the remote server is responsible for providing the DH parameters during the session setup, and OpenSSL on VSE uses DHE-RSA transparently if required by the server.

5.11.2 Using DHE-RSA with Java-based connector

Java supports all DHE-RSA related SSL cipher suites, but uses different names than OpenSSL. Java uses the underline character '_' to separate the parts within a cipher suite name, whereas OpenSSL uses dashes '-'. Table 5-2 shows the corresponding cipher suite names for OpenSSL and Java.

Table 5-2 DHE-RSA cipher suite names for OpenSSL and Java

Hex code	OpenSSL	Java
0x15	EDH-RSA-DES-CBC-SHA	SSL_DHE_RSA_WITH_DES_CBC_SHA
0x16	EDH-RSA-DES-CBC3-SHA	SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
0x33	DHE-RSA-AES128-SHA	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
0x39	DHE-RSA-AES256-SHA	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
0x67 ⁽¹⁾	DHE-RSA-AES128-SHA256	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
0x6B ⁽¹⁾	DHE-RSA-AES256-SHA256	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

⁽¹⁾ Cipher suites 67 and 6B require TLSv1.2

Note: Support for DHE-RSA in the VSE Connector Client is added by using an APAR. Check for the latest connector client version.

The following sections use VSE Connector Server and VSE Navigator as an example of how to configure DHE-RSA for Java-based connector.

Client-side configuration

For VSE Navigator, make following changes in the SSL properties file:

```
KEYRINGFILE=c:\\vsecon\\samples\\dhparms.pem
SSLVERSION=TLSv1.2
CIPHERSUITES=TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,TLS_DHE_RSA_WITH_AES_128_CBC_SHA
```

Note: The use of AES-256 in Java requires the unlimited security strength files for Java. You can download these files from the following website:

<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>

VSE-side configuration with IPv6/VSE

IPv6/VSE does not allow choosing specific SSL cipher suites. Therefore, the use of DHE-RSA depends on the configuration of the remote server or client:

- ▶ If VSE is the server, a client can request specific cipher suites.
- ▶ If VSE is the client, all supported cipher suites are sent to the server for negotiation.

To enable Diffie-Hellman support, the DH parameters must be available in the .pem file as specified by using the KEYFILE parameter for BSTTATLS or BSTTPRXY. The OpenSSL trace shows whether DH support is available:

```
*** DH parameters read successfully. DHE-RSA cipher suites are available.
```

VSE-side configuration with LE/C Multiplexer

In the VSE Connector Server's SSL configuration member (SKVCSSSL in ICCF library 59), specify SSL version, .pem file name, and SSL cipher suites by using the already provided parameters. However, with OpenSSL there are more parameter values:

```
SSLVERSION      = SSL30 | SSLV3 | TLS31 | TLSV1(1) | TLSV1.2(1) | ALL(1)
KEYRING         = CRYPTO.KEYRING
CERTNAME        = DHPARMS      <- The name of your .pem file
SESSIONTIMEOUT  = 86400
AUTHENTICATION  = SERVER
```

⁽¹⁾ These parameters are only available with OpenSSL.

To enable DHE-RSA for VSE Connector Server, you must add the DHE-RSA cipher suites to the CIPHERSUITES parameter:

```
CIPHERSUITES = ; COMMA SEPARATED LIST OF NUMERIC VALUES
0A, ; RSA1024_3DESCBC_SHA
2F, ; TLS_RSA_WITH_AES_128_CBC_SHA (SINCE TCP/IP 1.5E)
35, ; TLS_RSA_WITH_AES_256_CBC_SHA (SINCE TCP/IP 1.5E)
3C, ; TLS_RSA_WITH_AES_128_CBC_SHA256 (requires TLSv1.2)
39, ; TLS_DHE_RSA_WITH_AES_256_CBC_SHA (requires Diffie-Hellman)
67  ; TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (requires TLSv1.2 and DH)
```

The DH parameters must be available in the .pem file.

5.12 Considerations on Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is an encryption technique that provides public-key encryption similar to RSA. While the security strength of RSA is based on very large prime numbers, ECC uses the mathematical theory of elliptic curves and achieves the same security level with much smaller keys. The mathematical background of ECC is described in RFC 6090:

<http://tools.ietf.org/html/rfc6090>

The use of ECC in SSL/TLS is described in RFC 4492.

<http://tools.ietf.org/html/rfc4492>

In practice, ECC is often used with Diffie-Hellman to speed up performance. ECC does not replace RSA for authenticating the communication partners, but is used for generating the ephemeral DH session key with the help of an EC private key. RSA is still used for providing authentication. The related SSL cipher suites all have ECDHE-RSA in their names and complement the plain DHE-based cipher suites.

The main advantage of Elliptic Curve Cryptography with Diffie-Hellman (ECDHE-RSA) over plain Diffie-Hellman (DHE-RSA) is better performance and the same level of security with less key bits. A disadvantage is the additional effort for creating and maintaining the EC key.

The next section shows how to set up and use ECDHE-RSA on z/VSE.

5.13 Using ECDHE-RSA with OpenSSL on z/VSE

This section shows how to use the ECDHE-RSA-based SSL cipher suites on z/VSE.

Note: ECDHE-RSA is supported by APAR DY47545.

There are two different ECC implementations in OpenSSL, version 1.0.0 and later:

- ▶ A 32-bit implementation that is used on z/VSE.
- ▶ A 64-bit implementation that provides better performance, but requires latest 64-bit ecc compiler support and cannot be used on z/VSE. The OpenSSL code on z/VSE is compiled using `OPENSSL_NO_EC_NISTP_64_GCC_128`.

The next sections show how to create an EC key and upload it to VSE for use by OpenSSL.

5.13.1 Generating the EC key

Generating the EC key must be done using `openssl` on your workstation. At the time of writing, the Keyman/VSE utility does not provide support for generating and uploading EC keys. For our tests, we used these commands:

```
openssl ecparam -out ecparam.pem -name prime256v1
openssl genpkey -paramfile ecparam.pem -out ecdhkey.pem
```

The EC key is now contained in file `ecdhkey.pem` and looks as shown in Example 5-22.

Example 5-22 Generated EC key

```
D:\>type ecdhkey.pem
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGBYqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgOPvMUyVFy1S1fwx8
H0q1Er8ve9pgcvs6Ezfty8yq6qKhRANCAAS9d/zL/ZIwydd5EvLoLF3+GnUHQ/pu
PiN945ucTiLTj08YjZ7SCIWbGskb+DH32viG6+4goAoZQT3Tzoi3EYz8
-----END PRIVATE KEY-----
```

The EC key has the same string delimiters as an RSA private key, and therefore cannot be stored in the same PEM file together with the RSA key.

5.13.2 Uploading the EC key to VSE

You can upload the PEM file that contains the EC key to VSE by using FTP or any other file transfer mechanism that provides ASCII to EBCDIC translation. You can even use copy/paste into a new DITTO created file by using your Terminal Emulator. The target file on VSE must be a VSE/Librarian member. The member contents on VSE must look identical to the character representation on your workstation. Example 5-23 shows the uploaded member in DITTO.

Example 5-23 Uploaded EC key in VSE/Librarian

DITTO/ESA for VSE	LE - Library Member Edit		
Member ECDHKEY.PEM	Library PRD2.CONFIG	Col 1	Format CHAR
			SYSIPT data NO
			1...5...10...5...20...5...30...5...40...5...50...5...60...5...70..
00000	**** Top of data	****	

```

00001 -----BEGIN PRIVATE KEY-----
00002 MIGHAgEAMBMGBYqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQg0PvMUyVFy1S1fwx8
00003 H0q1Er8ve9pgcvs6Ezfty8yq6qKhRANCAAS9d/zL/ZIwydd5EvLoLF3+GnUHQ/pu
00004 PiN945ucTiLTj08YjZ7SCIWbGskb+DH32viG6+4goAoZQT3Tzoi3EYz8
00005 -----END PRIVATE KEY-----
00006 **** End of data ****

```

The following section shows how to use ECDHE-RSA with the Java-based connector.

5.13.3 Using ECDHE-RSA with Java-based connector

Java supports all ECDHE-RSA related SSL cipher suites, but uses different names than OpenSSL. Table 5-3 shows the supported cipher suites on z/VSE with their corresponding names for Java.

Table 5-3 ECDHE-RSA cipher suite names for OpenSSL and Java

Hex code	OpenSSL	Java
0xC012	ECDHE-RSA-DES-CBC3-SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
0xC013	ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
0xC014	ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
0xC027 ⁽¹⁾	ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

⁽¹⁾ Cipher suite C027 requires TLSv1.2

Note: ECDHE-RSA cipher suites with SHA-384 are currently not supported on z/VSE due to the general restriction of SHA-384 and SHA-512 in OpenSSL on z/VSE.

Client-side configuration

For VSE Navigator, make the following changes in the SSL properties file:

```

KEYRINGFILE=c:\\vsecon\\samples\\ecd.h.pem
SSLVERSION=TLSv1.2
CIPHERSUITES=TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

```

Specify your PEM file that contains the RSA key and DH parameters here. The EC key is not needed on the client side. It is only on VSE in a second file, ECDHKEY.PEM.

VSE-side configuration for IPv6/VSE

IPv6/VSE does not allow selecting specific SSL cipher suites. Instead, the string returned by `gsk_get_cipher_info` is passed back to OpenSSL when calling the `gsk_secure_soc_init` API function. This string contains all supported ECDHE-RSA based cipher suites.

To enable Elliptic Curve support, an EC key must be available in VSE library member ECDHKEY.PEM in the same VSE sublibrary specified by using the **KEYRING** parameter for **BSTTATLS** or **BSTTPRX**. The OpenSSL trace shows whether an EC key is available and can be used:

```

*** EC Private Key read successfully.
*** EC key set. ECDHE-RSA cipher suites are available.

```

VSE-side configuration for LE/C Multiplexer

OpenSSL in general is available for TCP/IP for VSE/ESA and Linux Fast Path (LFP) only by using the LE/C Multiplexer. The availability of ECDHE-RSA depends on the related application. For example, when using the VSE Connector Server, specify the ECDHE cipher suites in the server's SSL config member (SKVCSSSL):

```
SSLVERSION      = TLSV1.2
KEYRING         = CRYPTO.KEYRING
CERTNAME        = ECDHSSL
SESSIONTIMEOUT  = 86400
AUTHENTICATION  = SERVER
```

The EC key must be contained in a second member ECDHKEY.PEM in the VSE sublibrary specified by the **KEYRING** parameter. The **CIPHERSUITES** parameter includes the ECDHE-related cipher suites:

```
CIPHERSUITES = ; COMMA SEPARATED LIST OF NUMERIC VALUES
                2F, ; TLS_RSA_WITH_AES_128_CBC_SHA
                35, ; TLS_RSA_WITH_AES_256_CBC_SHA
                67, ; TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
                6B, ; TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
                C013, ; TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
                C014, ; TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
                C027, ; TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
```

Note: Support for ECDHE-RSA cipher suites in the VSE Connector Server by using the LE/C Multiplexer is added by using APAR. Check for the latest connector server version.

5.14 Restrictions

This section describes the restrictions and a known problem with OpenSSL on z/VSE.

5.14.1 No SHA-512 support

Apparently, the VSE C-compiler is unable to compile the SHA-512 related modules. The OpenSSL FAQ.txt file says:

“OpenSSL SHA-512 implementation depends on compiler support for 64-bit integer type. Few elder compilers [ULTRIX cc, SCOcompiler to mention a couple] lack support for this and, therefore, are incapable of compiling the module in question. The suggestion is to disable SHA-512 by adding no-sha512 to ./config [or ./Configure] command line. Another possible alternative might be to switch to GCC.”

As of this writing, the VSE code is compiled by using the **OPENSSL_NO_SHA512** option. This might change by using SHA-512, which is provided by CPACF.



Comparison of stacks and protocols

In this chapter, we compare the different IP stacks and their applications in terms of functionality and usability. We also provide hints and tips for using particular functions.

We also describe commonalities and differences in terms of provided application programming interfaces, performance, and Secure Sockets Layer (SSL).

Finally, considerations for IBM applications provide details about how IBM applications are affected by a particular IP stack.

This chapter includes the following topics:

- ▶ Stacks comparison
- ▶ Applications comparison
- ▶ Performance
- ▶ SSL
- ▶ Comparison of APIs
- ▶ Considerations for DB2 Server for VSE interfaces
- ▶ Considerations for IBM applications
- ▶ Known problem: ftp.exe hangs on Windows 7

6.1 Stacks comparison

In this section, we describe the IP stacks and compare their basic properties, applications, and programming interfaces.

6.1.1 Licensing

In this topic, we describe how the stacks are licensed:

- ▶ TCP/IP for VSE/ESA

Licensing of the stack provided by Connectivity Systems International depends on whether you obtain it from IBM as part of z/VSE or directly from CSI. If you obtain it from IBM by using the IBM Key Center, your license includes the base and application pak. If you obtain it directly from CSI, you have several options. Contact CSI for more information in this case.

A CSI license key is independent from your hardware.

- ▶ IPv6/VSE

A BSI license key depends on your CPU ID and is therefore only usable for one specific LPAR. If you run z/VM in this LPAR with multiple z/VSE guests, the same key can be used for all of these VSE systems. For more information, see 3.2, “Obtaining and activating a license key” on page 48.

- ▶ Linux Fast Path

The LFP function is part of z/VSE Advanced Functions and does not need to be licensed separately.

6.1.2 Installation libraries

In this topic, we describe the installation libraries of the following IP stacks:

- ▶ TCP/IP for VSE/ESA

The installation library for TCP/IP for VSE/ESA is PRD1.BASE.

Update with z/VSE 5.2: the default installation library is now PRD2.TCPIPC (C for CSI).

- ▶ IPv6/VSE

The installation library for IPv6/VSE is PRD2.TCPIPB (B for Barnard).

- ▶ Linux Fast Path

The Linux Fast Path binary files belong to VSE Advanced Functions and are installed in IJSYSRS.SYSLIB. Related job skeletons are provided in ICCF library 59.

6.1.3 Virtual storage organization

In this topic, we describe the storage requirements of the different stacks.

TCP/IP for VSE/ESA

The default partition size for TCP/IP for VSE/ESA is 20 MB per default. However, use a partition with at least 30 MB to benefit from the 31-bit use of the product.

Some of the TCP/IP applications run in the same partition as the IP stack, whereas the following applications use their own partitions:

- ▶ TN3270 daemons, FTPD daemons, automatic FTP (AutoFTP), and automatic line printer requester (AutoLPR) run in the same partition as the IP stack.
- ▶ FTPBATCH runs in a separate partition.

Different partition priorities for TN3270 versus FTP/LPR require multiple IP stacks.

TCP/IP for VSE/ESA requires 1050 K of partition GETVIS (ANY) space and 1050 K for SETPFIX (ANY) for each LINK of an OSAX device.

The minimum suggested partition size for FTPBATCH as a client is 4 MB. The minimum suggested partition size for FTPBATCH as a server is 8 MB to support up to four active connections. For a server, add 256 KB for each additional concurrent session.

So, if you have, for example,

```
// EXEC FTPBATCH,SIZE=FTPBATCH,PARM='FTPDPORT=2121,UNIX=BIN,MAXACT=12',
```

the suggested size of the FTPBATCH server is:

8192 KB (8 MB) + (8*256 KB) = 8192 KB + 2048 KB = 10240 KB (10 MB)

This assumes for the client and server that DEBUG=ON, SET DIAGNOSE ON, or SET DIAGNOSE EVENTS was not issued in the FTPBATCH partition. If any of these diagnostics are active, add another 2 MB.

IPv6/VSE

The IPv6/VSE stack requires a minimum 20 MB partition. This amount of storage allows for stack start and storage for a few users, but is too small for production usage.

BSI recommends the use of a separate dynamic partition for each function for more granularity of control. Customers can increase VSIZE and DPD space to allow for more dynamic partitions.

Sample partition allocations are shown in the following examples:

```
Q1 6M autolpr process
Q2 6M autolpr print jobs
R1 6M autoftp process
R2 6M autoftp transfer jobs
S1 24M IP stack for ftp/lpr
T1 24M IP stack for tn3270
T2 24M tn3270/3270 printing sessions
U1 20M ftp server
```

IPv6/VSE requires SETPFIX LIMIT=(256 K, 1100 K) per DEVICE command.

Linux Fast Path

Linux Fast Path (LFP) does not run in a partition. Most control blocks are allocated in SVA PFIX memory in the ANY area.

6.1.4 Commands

In this topic, we describe some examples that show how commands are specified for the various stacks.

TCP/IP for VSE/ESA

For more information about the following available commands for TCP/IP for VSE/ESA, see *TCP/IP for VSE Command Reference*, SC33-6764:

```
SET IPADDR=172.16.1.252
SET MASK=255.255.252.0
DEFINE LINK,ID=OSA,TYPE=3172,DEV=600,MTU=4096
DEFINE ADAPTER,LINKID=OSA,NUMBER=0,TYPE=TOKEN_RING
DEFINE ROUTE,ID=WORLD,LINKID=OSA,IPADDR=0.0.0.0, -
GATEWAY=172.16.1.1
SET DNS1=172.16.1.222
DEFINE NAME,NAME=SERVER00,IPADDR=172.16.1.254
DEFINE TELNETD ...
DEFINE FTPD ...
DEFINE EVENT ...
```

Typically, you include TCP/IP commands into your IPINIT member to be run at stack start. It is also possible to include other commands in more L-members. Any command also can be entered online from the VSE operator console at the TCP/IP command prompt.

IPv6/VSE

For more information about IPv6/VSE stack commands, see *IPv6/VSE IPv6 Installation Guide*, SC34-2616. For more information about the following application-related commands (FTP, Telnet, mail, and so on), see *IPv6/VSE IPv6 User's Guide*, SC34-2618:

```
DEVICE LCS600 LCS 600 TOKENRING
LINK LCS600 1 172.16.1.252 255.255.252.0 4096
ROUTE LCS600 172.16.0.0 255.255.252.0 172.16.1.1 0
ROUTE LCS600 0.0.0.0 0.0.0.0 172.16.1.1 1
HOST SERVER00 172.16.1.254
DNS 172.16.1.222
```

All server applications can be stopped by using the **SHUTDOWN** command.

LFP

For more information about commands that are available for LFP, see *z/VSE TCP/IP Support*, SC34-2640. Depending on the environment that is used (under z/VM, VIA, or in LPAR), commands are divided into Linux shell commands, z/VM SMSG commands, and JCL to maintain an LFP instance on z/VSE.

6.1.5 Comparison of protocols

In this topic, we describe the supported protocol versions of the different IP stacks.

TCP/IP for VSE/ESA

The supported protocol is IPv4.

IPv6/VSE

The following protocols are supported:

- ▶ IPv4
- ▶ IPv6

Linux Fast Path

The following protocols are supported:

- ▶ IPv4 (since z/VSE 4.3)
- ▶ IPv6 (since z/VSE 5.1)

6.2 Applications comparison

In this section, we describe the TCP/IP applications that are provided by the different stacks and their similarities and differences.

6.2.1 FTP server

There are several differences in how FTP servers can be used in the different stacks. TCP/IP for VSE/ESA provides an internal FTP daemon (FTPD) that runs as a subtask in the TCP/IP partition, and an external FTP server (FTPBATCH) that runs in a separate partition. IPv6/VSE provides one FTP server (BSTTFTPS) that runs in any static or dynamic partition. LFP does not provide its own FTP server, but you can use BSTTFTPS and the Virtual z/VSE FTP Daemon with LFP.

TCP/IP for VSE/ESA

TCP/IP for VSE/ESA provides the following types of FTP daemons:

- ▶ **FTPD**

This daemon runs as a subtask in the TCP/IP partition. Depending on the UNIX parameter, graphical FTP clients, such as FileZilla, can display the VSE file systems. You define an FTPD by using the following command.

DEFINE FTPD,ID=FTP,UNIX=BIN,PORT=21,COUNT=3

With UNIX=NO (default), you can connect to a batch FTP client (for example, ftp.exe on Windows). However, a graphical client, such as FileZilla, does not work with this setting, such as ftp.exe on Windows, as shown in Example 6-1.

Example 6-1 Using the MS Windows batch FTP client

```
D:\>ftp vssl
Connected to vssl.boeblingen.de.ibm.com.
220-TCP/IP for VSE Internal FTPDAEMN 01.05 F 20120717 12.01
    Copyright (c) 1995,2006 Connectivity Systems Incorporated
220 Ready for new user
User (vssl.boeblingen.de.ibm.com:(none)): jsch
331 User name okay, need password
Password:
230 User logged in, proceed
ftp> dir
200 Command okay
150 File status okay; about to open data connection
drw-rw-rw-  1 vse direct      0 MAY 02 13:11 ICCF
drw-rw-rw-  1 vse direct      0 MAY 02 13:11 POWER
drw-rw-rw-  1 vse direct      0 MAY 02 13:11 PRD1
drw-rw-rw-  1 vse direct      0 MAY 02 13:11 PRD2
...
```

With UNIX=YES or UNIX=BIN, FileZilla displays the VSE file systems as defined by using DEFINE FILE and DEFINE FILESYS.

► FTPBATCH

This daemon runs in a separate partition. FTPBATCH can be run as a client or as a server. As with FTPD, FTPBATCH has a UNIX parameter that must be set to YES when a graphical FTP client is used.

Depending on the file definitions in the IPINIT member, DTPD and FTPBATCH provide access to all VSE file systems: Librarian, POWER, and VSAM.

Typical FTPBATCH JCL is shown in the following example:

```
// LIBDEF *,SEARCH=(PRD2.TCPIPC,PRD1.BASE)
// EXEC FTPBATCH,SIZE=FTPBATCH,PARM='UNIX=YES,FTPDPOR=21'
/*
```

Figure 6-1 shows the FileZilla client window that is connected to FTPBATCH.

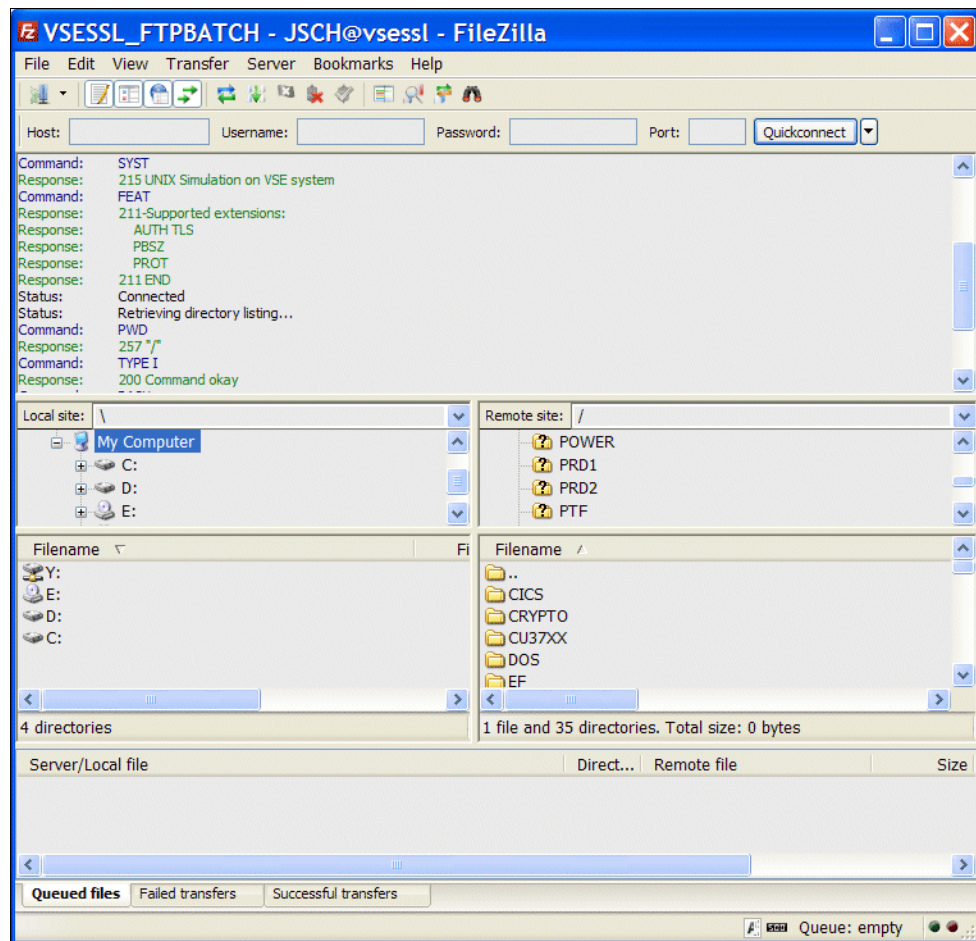


Figure 6-1 The use of FileZilla client with FTPBATCH

IPv6/VSE

The main difference between the FTP server that is provided with IPv6/VSE and the CSI-provided FTP servers is the display of VSE file systems. Although CSI displays the file systems as defined by using DEFINE FILE and DEFINE FILESYS, the BSI-provided FTP server displays only one mount point. Only one mount point can be defined for one given BSTTFTPS instance.

Example 6-2 shows the JCL to connect to BSTTFTPS.

Example 6-2 JCL for a BSTTFTPS showing the PRD2 library

```
// EXEC BSTTFTPS,SIZE=BSTTFTPS,OS390
ID 02
*
OPEN 9.152.131.189 21
*
* USE THE IBM PROVIDED BSM SECURITY EXIT
BSSTISX
*
* DEFINE THE DEFAULT FILE SYSTEM TO USE
*
SMNT LIBRARY PRD2
*
ATTACH SERVER-1
ATTACH SERVER-2
ATTACH SERVER-3
*
/*
```

When you are connecting to BSTTFTPS with the definition that is shown in Example 6-2, FileZilla shows VSE library PRD2 and its sublibraries, as shown in Figure 6-2.

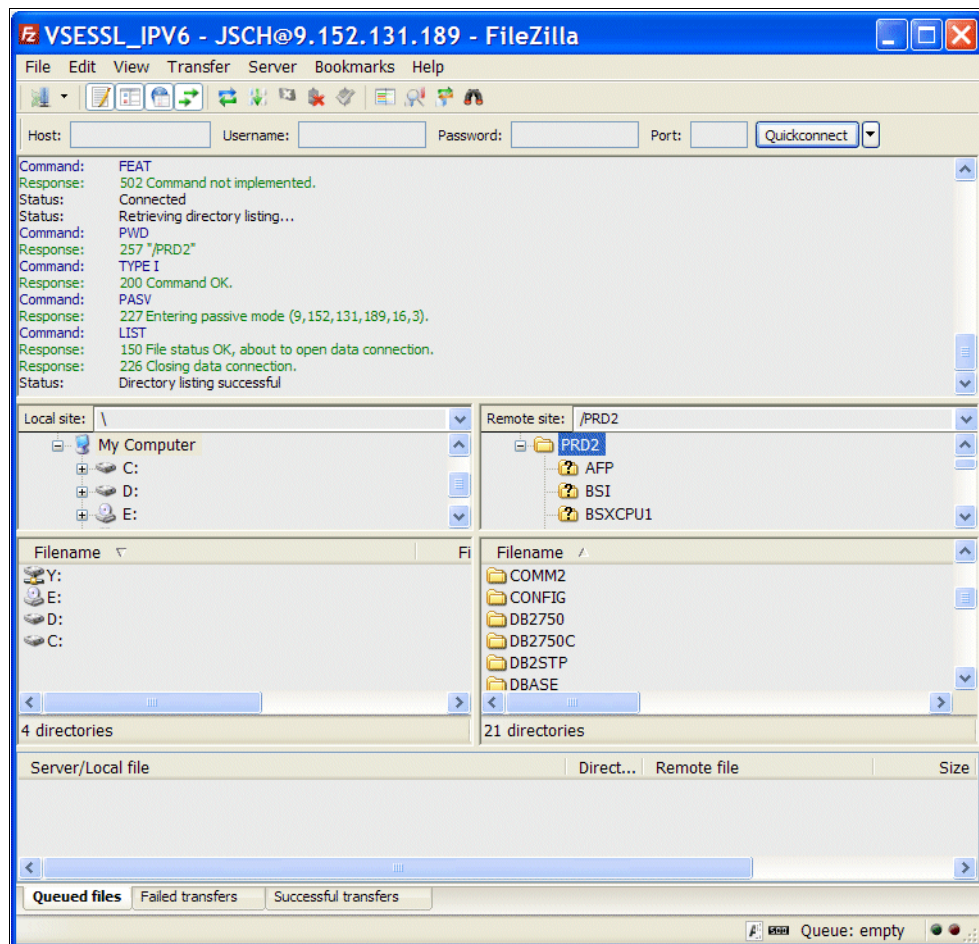


Figure 6-2 FileZilla client that shows the PRD2 library

The following methods can be used to establish a mount point:

- ▶ The BSTTFTPS startup commands specify a default file system to be mounted when an FTP client logs in to the FTP server. There can be only one default file system; if you specify multiple default file systems in the BSTTFTPS start commands, the last one is used.
- ▶ The FTP-USER command that is specified in the BSTTSCTY.T security table defines users. You can optionally define a file system to be mounted for each user on the FTP-USER command, as shown in the following example:

```
FTP-USER PWRTEST BSI          SMNT-POWER/LST
```

- ▶ The FTP client can issue a command to mount a file system by using the SMNT (structure mount) command, as shown in the following examples:

```
smnt power  
smnt vsam vsam.catalog.name
```

When an **smnt** command is sent from the FTP client, spaces are used between each parameter. If the FTP client does not support the **smnt** command, use the **quote** command to issue the **smnt** command, as shown in the following example:

```
quote smnt power
```

- ▶ The FTP client can also mount a file system by using a change directory (**cd**) command, as shown in the following examples:

```
cd smnt-power/lst  
cd smnt-vsam-vsam.catalog.file  
cd smnt-library-prd2/config
```

Because the **cd** command does not allow the use of spaces, dashes are used between parameters. Internally, the FTP server changes the dashes to spaces before the embedded **smnt** command is processed.

- ▶ Many PC FTP clients allow you to define connections and save them for future use. Most of these FTP clients also allow you to specify initial command (or commands) that are automatically issued after the login is complete. These initial commands provide an easy way to mount a file system.

For example, by using Core FTP LE, you can specify predefined commands to be issued at various points in the connection process. In the Site Manager window, click **Advanced**. Then, in the Advanced Site Settings box, select **Scripts/Cmds**, as shown in Figure 6-3.

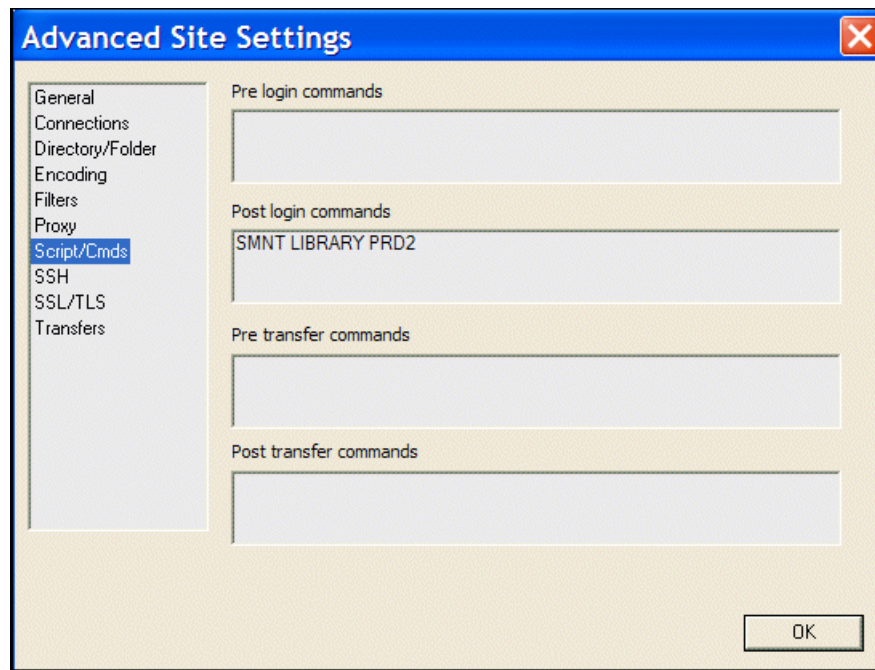


Figure 6-3 Specifying predefined commands in Core FTP LE

This example causes the PRD2 library to be displayed when this connection is used. In this way, you can define one connection per file system.

Figure 6-4 shows the Site Manager window with different connection definitions.



Figure 6-4 Displaying the PRD2 library in Core FTP LE

LFP

When the Virtual z/VSE FTP Daemon with Linux Fast Path is used, your FTP client displays the file system as provided by the VSE Connector Server. After the VSE Connector Server is started on VSE and the z/VSE Virtual FTP daemon (for example, on your workstation), FileZilla shows the default file systems that are using the VSE Connector Server.

ICCF skeleton SKVCSLIB in ICCF library 59 allows the user to add more VSE libraries. For VSAM, the VSE Connector Server reads the standard labels and displays all defined files, as shown in Example 6-3.

Example 6-3 Specifying VSE libraries in skeleton SKVCSLIB

```
* *****
* LIBRARIAN CONFIGURATION MEMBER FOR VSE CONNECTOR SERVER
* *****
*
* ADD THE NAME OF YOUR LIBRARIES TO THIS MEMBER IF YOU WANT TO
* HAVE ACCESS TO THEM WITH THE VSE CONNECTOR SERVER.
*
* NOTE: EACH LINE SPECIFIES ONLY ONE LIBRARY
*
* *****
CRYPTO
PRD1
PRD2
```

When the default configuration for VSE Connector Server is used, the FileZilla client displays the default file systems, as shown in Figure 6-5.

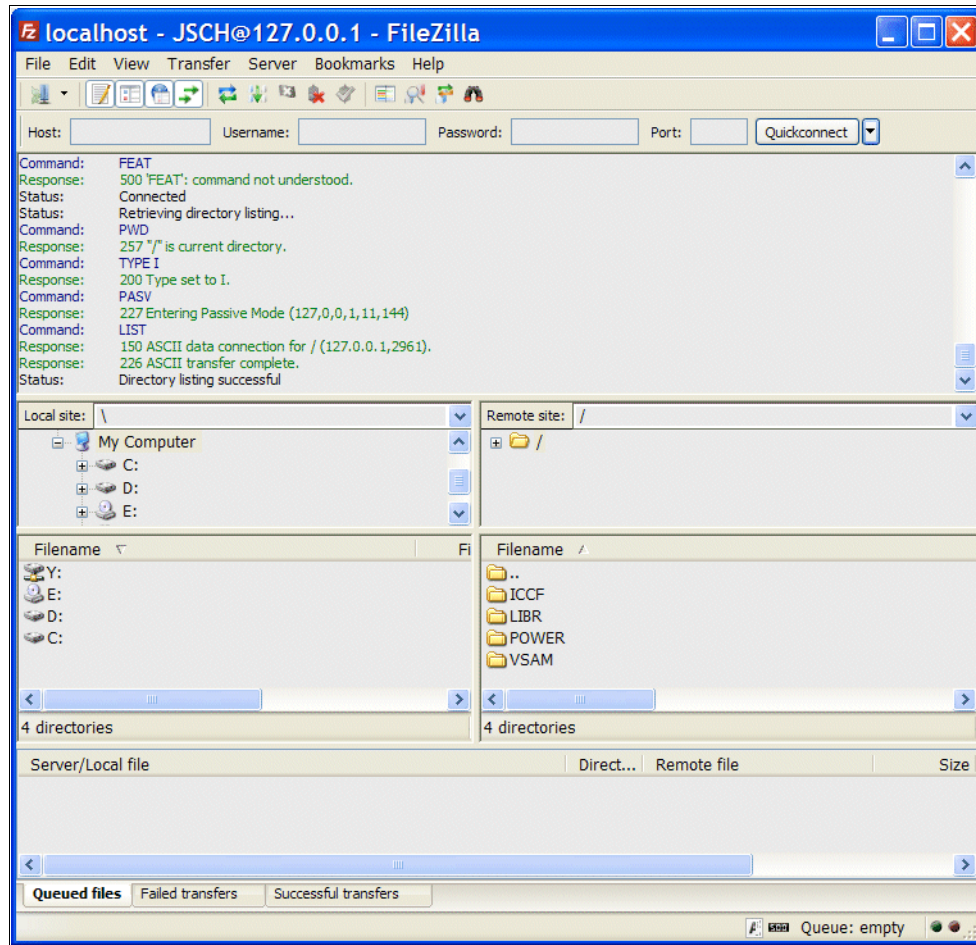


Figure 6-5 Using FileZilla client with LFP and z/VSE Virtual FTP Daemon

6.2.2 FTP clients

The behavior of the FTP clients that are provided by the different stacks is similar. However, there are differences in JCL.

A typical problem when mainframe files are sent to non-mainframe platforms is handling the record format and record length. Because record-based file systems exist only on mainframes, FTP servers, such as FileZilla server, do not accept any site commands setting RECFM or LRECL.

To overcome this problem, TCP/IP for VSE/ESA and IPv6/VSE provide functionality that allows their FTP clients to use this setting at the client side.

TCP/IP for VSE/ESA

Example 6-4 shows how to send a VSAM file to an external FTP server by using a fixed record length of 128 bytes. You specify these values as parameters of the **PUT** command.

Example 6-4 TCP/IP for VSE/ESA FTP client

```
// DLBL INFILE,'BF.WORK.H0590A.#05',,VSAM,CAT=CSYSWK6
// EXEC FTPBATCH,SIZE=FTPBATCH
LOPEN
EXECUTE 139LOGIN.L
CD REG\OP\SHARED\DOWN
PUT %INFILE,ESDS,F,0128 FMASTCR.TXT
QUIT
/*
```

IPv6/VSE

Example 6-5 shows how BSTTFTPC sends a VSAM file to a non-mainframe FTP server by using a fixed record length of 128 bytes. The **OUTPUT** command is used to specify the values.

Example 6-5 IPv6/VSE FTP client

```
// DLBL INFILE,'BF.WORK.H0590A.#05',,VSAM,CAT=CSYSWK6
// EXEC BSTTFTPC,SIZE=BSTTFTPC
* $$ SLI MEM=139LOGIN.L
CWD REG\OP\SHARED\DOWN
INPUT VSAM INFILE
TYPE I
STOR FMASTCR.TXT
QUIT
/*
```

LFP

As of this writing, LFP does not provide its own FTP client. However, you can use BSTTFTPC from the IPv6/VSE stack.

6.2.3 Uploading a virtual tape into VSAM

Sometimes it is useful to upload a virtual tape file (.aws) into VSAM for further processing. For example, performing a Fast Service Upgrade (FSU), or installing products from an optional product tape. This section shows how to do this using the different IP stacks. In all examples, we use the interactive FTP batch client (ftp.exe) in MS Windows. We assume that the VSE.VTAPE.FILE has been defined by using ICCF skeleton SKVTAPE.

TCP/IP for VSE/ESA

When using TCP/IP for VSE/ESA you first navigate to the data part of the VSAM file, then specify record length and format. Finally, you do the FTP put into the data part of the file as shown in Example 6-6.

Example 6-6 Uploading a vtape by using TCP/IP for VSE/ESA

```
D:\>ftp vsessl
Connected to vsessl.boeblingen.de.ibm.com.
220-TCP/IP for VSE Internal FTPDAEMN 01.05 F 20120717 12.01
    Copyright (c) 1995,2006 Connectivity Systems Incorporated
```

```

220 Ready for new user
User (vsessl.boeblingen.de.ibm.com:(none)): jsch
331 User name okay, need password
Password:
230 User logged in, proceed
ftp> cd vse.vtape                <-- change directory
250 Requested file action okay, completed
ftp> quote site lrecl 32758        <-- specify record length
200 Command processed
ftp> quote site recfm v            <-- specify record format
200 Command processed
ftp> bin
200 Command okay
ftp> put mytape.aws file          <-- do the FTP put into the file
200 Command okay
150-About to open active data connection
File:VSE/VTAPE/FILE
Type:Binary Recfm:V Lrecl: 32758
CC=ON UNIX=ON RECLF=OFF TRCC=OFF CRLF=ON NAT=NO CONT=OFF
MODE=Stream STRU=File
150 File status okay; about to open data connection
226-Bytes received: 560,819
Records received: 18
Transfer Seconds:          .06 ( 9128K per second)
File I/O Seconds:         .06 ( 9128K per second)
226 Closing data connection
ftp: 560819 bytes sent in 0,04Seconds 14379,97Kbytes/sec.
ftp>

```

IPv6/VSE

When using the FTP server (BSTTFTPS) of IPv6/VSE, first mount the VSAM user catalog with the **smnt** command, if not already mounted by the server. Then, specify the DLBL name of the VSAM file and transfer type binary as shown in Example 6-7.

Example 6-7 Uploading a vtape by using IPv6/VSE

```

D:\>ftp vsessl
Connected to vsessl.boeblingen.de.ibm.com.
220 IPv6/VSE FTP Server Ready.
User (vsessl.boeblingen.de.ibm.com:(none)): jsch
331 User name OK, need password.
Password:
230 User JSCH logged in, proceed.
ftp> quote smnt vsam vsesp.user.catalog    <-- mounts the user catalog
200 Command OK.
ftp> quote site output esds vtape1 recsz 32758 <-- vtape file DLBL name
200 Command OK.
ftp> bin
200 Command OK.
ftp> put mytape.raws
200 Command OK.
150 File status OK, about to open data connection.
250 Requested file action OK, completed.

```

```
ftp: 327240 bytes sent in 0,02Seconds 19249,41Kbytes/sec.  
ftp>
```

The record format (variable or fixed) does not need to be specified when using IPv6/VSE.

Linux Fast Path

With LFP, you will most likely use the Virtual z/VSE FTP Server. In Example 6-8 we started the Virtual z/VSE FTP Daemon on our local workstation, so we connect to 127.0.0.1 (the localhost). The FTP server is configured to connect to the same VSE system that is used in the other examples.

Example 6-8 Uploading a vtape using LFP

```
D:\>ftp 127.0.0.1  
Connected to 127.0.0.1.  
220 IBM Virtual z/VSE FTP Server on BR9WG7D4 (version 1.1) ready to serve.  
User (127.0.0.1:(none)): jsch  
331 Password required for jsch.  
Password:  
230 User jsch logged in. Idle timeout is 15 minutes.  
ftp> cd vsam  
250 CWD command successful, current directory is "/VSAM".  
ftp> cd vsesp.user.catalog  
250 CWD command successful, current directory is "/VSAM/VSESP.USER.CATALOG".  
ftp> bin  
200 Type set to I.  
ftp> put VSE5200P4_20140225.RAWS vse.vtape.file  
200 PORT command successful.  
150 Binary data connection for /VSAM/VSESP.USER.CATALOG/VSE.VTAPE.FILE (127.0.0.  
1,57428).  
ftp: 39138734 bytes sent in 1,74Seconds 22545,35Kbytes/sec.  
ftp>
```

The Virtual z/VSE FTP Server allows you to specify transfer mode S (stream, which is the default) or B (block). When using stream mode, the target file is filled record by record according to the VSAM cluster definition. It is not needed to specify a specific record length or format.

Note: This support requires updates in the VSE Connector Server and Virtual z/VSE FTP Server from September 2014 or later.

6.2.4 AutoFTP

AutoFTP means that an FTP client is started automatically when a predefined event occurs. Only one type of an event is supported by all stacks: the appearance of a new entry on the VSE/POWER LST or PUN queue in a specified class.

TCP/IP for VSE/ESA

Automatic FTP is configured with the DEFINE EVENT command. When you run this command, you must complete the following tasks:

- ▶ Specify class to monitor
- ▶ Use autoFTP scripts to control an FTP client session

Example 6-9 shows an example of an autoFTP script that contains variables for the VSE/POWER job name, number, and form number.

Example 6-9 Sample autoFTP script that contains variables

```
LOPEN 10.215.4.14
LUSER CMS15
LPASS CMS15
LCD POWER.LST.0
OPEN 10.215.100.12
USER archive
PASS mypasswd
SETVAR &LFN = &PWRNAME + "." + &PWRNUMB
SETVAR &ONAME = &PWRNAME + "." + &PWRNUMB + "." + &PWRFORM + "." + "1"
SETVAR &NNAME = &PWRNAME + "." + &PWRNUMB + "." + &PWRFORM
cd /arsload/Prod1
LSITE CRLF ON
ASCII
PUT &LFN &ONAME
RENAME &ONAME &NNAME
CLOSE
```

This allows sending many differently named reports out of one class no matter what the job name is.

IPv6/VSE

IPv6/VSE provides a sample REXX program BSTTECSI.PROC that can be used to monitor a VSE/POWER class. This program performs the following tasks:

- ▶ Creates appropriate JCL that is submitted to run in another partition.
- ▶ Reproduces the reading of the autoftp script and generates the proper FTP commands.

Another REXX sample, BSTTAFTP.PROC, shows how to autoFTP VSE/POWER queue entries. The BSTTAFTP.PROC scans the POWER LST queue looking for member to automatically FTP. By default it looks for members in class X. When a member appears, the class is changed to Y and a BSTTFTPC job is submitted to process the member. The BSTTAFTP.PROC REXX EXEC handles jobnames, numbers, classes, and so on.

If you want to use extensive symbolic variable support you should look at using the BSIREXXC.PROC to invoke the BSTTFTPC, BSTTMTPC, BSTTREXC, and BSTTLPRC utilities. The BSIREXXC.PROC supports many symbolic variables.

See the *IPv6/VSE Migration Guide* for more information about the BSIREXXC.PROC.

6.2.5 TN3270

In this topic, we describe an example of supporting multiple CICS partitions.

TCP/IP for VSE/ESA

With TCP/IP for VSE/ESA, you can use multiple port numbers to control target CICS partitions by using the terminal ID prefix, as shown in the following example:

```
DEFINE TELNETD,ID=H0,COUNT=10,PORT=23,TARGET=PRODCICS
DEFINE TELNETD,ID=H5,COUNT=10,PORT=5000,TARGET=PRODICCF
DEFINE TELNETD,ID=FS,COUNT=10,PORT=6000,TARGET=PRODCICS
```

Port number was specified in the client TN3270 definition.

IPv6/VSE

With IPv6/VSE, the following options are available:

- ▶ One TN3270 partition for each port, which might result in the use many partitions.
- ▶ One TN3270 partition (such as T2) at port 23 with multiple pools defined, which are provided the IP stack for TN3270 runs in T1.

Example 6-10 shows how to use TN3270E LU/Resource names to specify the proper pool.

Example 6-10 The use of terminal pools

```
// EXEC BSTTVNET,SIZE=BSTTVNET,DSPACE=3M
ID 01
OPEN 10.28.209.28 23
APPLID PRODICCF VSE1 ICCF
APPLID PRODCICS VSE1 CICS
TERMINAL H001 GENERIC PRODCICS DEDICATE POOL H0CICS
TERMINAL H002 GENERIC PRODCICS DEDICATE POOL H0CICS
TERMINAL H501 GENERIC PRODICCF DEDICATE POOL H5ICCF
TERMINAL H502 GENERIC PRODICCF DEDICATE POOL H5ICCF
TERMINAL FS01 GENERIC PRODCICS DEDICATE POOL FSCICS
TERMINAL FS02 GENERIC PRODCICS DEDICATE POOL FSCICS
ATTACH TN3270E
/*
```

6.2.6 Printing

There are several options available for setting up printing in TCP/IP for VSE/ESA and IPv6/VSE.

TCP/IP for VSE/ESA

TCP/IP for VSE/ESA provides the following alternatives for printing:

- ▶ LPR (Batch/Auto)
- ▶ LPD (LPR data to POWER)
- ▶ GPS (LPR)

The Generalized Print Server (GPS) is an optional feature that takes 3270 a data stream, converts EBCDIC to ASCII, and uses LPR to send print data to a network printer. The GPS daemon is defined in the TCP/IP partition with the DEFINE GPSPD command, as shown in the following example:

```
DEFINE GPSPD,ID=HSP1,STORAGE='PRD2.TEMP', -
      IPADDR=HS139,TERMNAME=HSP1,PRINTER=DEFLJ3, -
      INSESS=YES,TARGET=PRODCICS,INSERTS=L681
```

IPv6/VSE

IPv6/VSE provides the following alternatives for printing:

- ▶ LPR (batch/auto)
- ▶ FTP (instead of LPD)
- ▶ TN3270E with a client for SNA or non-SNA printer emulation
- ▶ TN3270E without a client (DIRECT, LPR, FTP)
- ▶ VSE2PDF

TN3270 Printing is included in the base product. Printers are defined in the TN3270 partition with commands that are similar to the following example:

```
* 1u applid ip addr port inserts queue
DIRECT HSP1 VIACICS1 HS139 515 L681 DEFLJ3
```

6.2.7 AutoLPR

AutoLPR provides functionality for printing z/VSE data on printers that are connected to remote TCP/IP hosts.

TCP/IP for VSE/ESA

TCP/IP for VSE/ESA provides the **DEFINE EVENT** command for establishing actions that are triggered when specified events occur on your VSE system. Consider the following points:

- ▶ It allows you to specify the class to monitor.
- ▶ It uses AUTOLPR scripts to control printing.
- ▶ The script name is specified in the USER parameter of the * \$\$ LST card.

IPv6/VSE

IPv6/VSE provides a sample REXX program, BSTTALPR.PROC, for monitoring a VSE/POWER LST class. Consider the following points:

- ▶ It creates appropriate JCL that is submitted in another partition to print LST a queue entry.
- ▶ It can read an AUTOLPR script and generate related IPv6/VSE LPR commands.
- ▶ The LST card in the original job must specify the INSERTS name on the UCS parameter.

6.2.8 Inserts coding

INSERTS coding is used for further controlling the print output. By using this coding, you can include printer control data before the file, after the file, and after each form feed.

TCP/IP for VSE/ESA

TCP/IP for VSE/ESA uses an INSERTS phase that contains printer control commands. The commands are specified in the Printer Command Language (PCL) that was originally created by Hewlett-Packard.

The *TCP/IP for VSE 1.5F User's Guide* provides the example that is shown in Example 6-11.

Example 6-11 INSERTS phase for TCP/IP for VSE

```
* $$ JOB JNM=INSCOP2,CLASS=4,DISP=D
// JOB INSCOP2
// LIBDEF *,SEARCH=(PRD2.TCPIPC)
// LIBDEF PHASE,CATALOG=PRD2.TCPIPC
// OPTION CATAL,LIST
// EXEC ASMA90
*****
*
* This inserts phase causes two copies of the output to be printed. *
* *
* 1B45 = Printer Reset *
* 1B266C3258 = Two Copies of Output (1B266C is the category ) *
* (32 is Ascii for 2 copies ) *
* (58 is the command code ) *
```

```

*                                                    *
*                                                    *
* A trailer of 1B45 resets the printer.            *
*                                                    *
*****
INSCOPY2 INSERTS DEFINE,                            X
                HEADER=1B451B266C3258,              X
                TRAILER=1B45
                END
/*
// EXEC LNKEDT
/&
* $$ E0J

```

All command codes must be specified in hexadecimal.

IPv6/VSE

For IPv6/VSE, an INSERTS member contains printer control commands in decimal. Example 6-12 shows various printer control commands.

Example 6-12 INSERTS member for IPv6/VSE

```

CATALOG L681.I REPLACE=YES
* L681
* RESET to printer defaults
HEADER 027 069
* 13 CPI 13 = ASCII X'3133' = 049 051
HEADER 027 040 115 049 051 072
* LANDSCAPE
HEADER 027 038 108 049 079
* 68 LINES PER PAGE 68 = ASCII X'3638' = 054 056
HEADER 027 038 108 054 056 080
* 8 LINES PER INCH 8 = ASCII X'38' = 056
HEADER 027 038 108 056 068
* TOP MARGIN 4 LINES 4 = ASCII X'34' = 052
HEADER 027 038 108 052 069
* LEFT MARGIN 3 COLUMNS 3 = ASCII X'33' = 051
HEADER 027 038 097 051 076
* RESET to printer defaults
TRAILER 027 069
/+

```

6.2.9 Email

Sending emails (and optionally attaching files) is supported by TCP/IP for VSE/ESA and IPv6/VSE. However, there are differences in how file attachments are defined. In this topic, we describe these differences in an example that sends the subcapacity measurement report to IBM. When you run the IBM SCRTTOOL, the output goes into a VSAM ESDS file.

TCP/IP for VSE/ESA

Example 6-13 shows the JCL for sending the subcapacity measurement report to IBM. The VSAM DLBL name (REPORT) is changed to a PC file name (report.csv) when the report is attached to the email.

Example 6-13 Sending the CRT report to IBM by using TCP/IP for VSE

```
* $$ JOB JNM=EMAIL,CLASS=Y,DISP=D
// JOB EMAIL
// DLBL REPORT,'DBGREP',,VSAM,CAT=DATACAT
// EXEC EMAIL,SIZE=EMAIL,PARM='ID=00'
SET HOST=d06av01.portsmouth.uk.ibm.com
SET RPORT=12345
SET FROM=jremus@de.ibm.com
SET TO=jremus@de.ibm.com
SET Subject=SCRT Sample Report <HOLD>
TEXT
Please find attached my SCRT sample report generated on SCRTTEST.
/+
SEND REPORT AS REPORT.CSV
QUIT
/*
/&
* $$ EOJ
```

IPv6/VSE

Example 6-14 shows how to send the subcapacity measurement report to IBM by using IPv6/VSE. By using the BSTTMTPC program, you can send an email from a z/VSE batch partition and optionally attach any type of file.

Example 6-14 Sending the CRT report to IBM by using IPv6/VSE

```
// DLBL REPORT,'file.name',,VSAM,CAT=.....
// LIBDEF PHASE,SEARCH=ipv6lib.slib
// LIBDEF SOURCE,SEARCH=(PRD2.CONFIG,ipv6lib.slib)
// EXEC BSTTMTPC,SIZE=BSTTMTPC
ID 00
OPEN 10.150.10.65 25
HELO name.com
MAIL FROM: ...
RCPT TO: SCRTLMS@DK.IBM.COM
SUBJ SUBJECT: <AUTO>
ORGA ORGANIZATION: your_id
*
DATA
*
INPUT VSAM REPORT
TYPE A
DISP ATTACH
INCLUDE SUBCAP.CSV
*
QUIT
/*          EOF for BSTTMTPC commands
/*          EOF for email text read by DATA command
```

6.2.10 Creating PDF documents

CSI and BSI provide a means for converting z/VSE data into PDF documents.

TCP/IP for VSE/ESA

The PDF Conversion Facility is part of TCP/IP for VSE/ESA and can generate a PDF from Power, LIBR, VSAM, or CICS, if you have the CICS Access Facility (CAF) add-on component. The PDF Conversion Facility can be started from the FTPD, the mail client, and the HTTPd web server.

For more information about configuring and the use of the PDF Conversion Facility, see the *TCP/IP for VSE User's Guide*, SC33-6763.

IPv6/VSE

Any licensee of IPv6/VSE also is granted a license available at no extra cost to run VSE2PDF/Lite, which can be used to convert basic reports to simple PDF files. Output from the conversion process can be emailed to users.

VSE2PDF/Lite is equivalent to the CSI PDF Conversion Facility; the full VSE2PDF product is a complete PDF report distribution facility.

When VSE2PDF/Lite is started, it appears to VSE/POWER as a printer. When output appears in the VSE/POWER LST queue of the correct class and destination, VSE/POWER prints the LST queue member to VSE2PDF/Lite. As VSE/POWER output is received, VSE2PDF/Lite converts it into a PDF document and emails the PDF document to a single user. Optionally, the PDF document can be transferred to an FTP server.

Because VSE2PDF/Lite uses the VSE/POWER device driver interface, no scanning of VSE/POWER queues is required. System resources are used only while output is received from VSE/POWER and sends the PDF output to the wanted destination.

For more information about VSE2PDF/Lite, see *IPv6/VSE User's Guide*, SC34-2618.

6.2.11 Remote EXEC client

By using a Remote Execution Protocol (REXEC) client, you can issue a command to a separate TCP/IP system that is running an REXEC daemon.

TCP/IP for VSE/ESA

TCP/IP for VSE/ESA provides an interactive REXEC client that can be started from a CICS terminal and a batch client. *TCP/IP for VSE 1.5F User's Guide*, SC33-6763 provides the example that is shown in Example 6-15 for a batch client.

Example 6-15 TCP/IP for VSE batch REXEC client

```
* $$ JOB JNM=REXEC,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB REXEC
// EXEC CLIENT,PARM='APPL=REXEC,ID=nn,QUIET=YES'
REXEC command 1
REXEC command 2
REXEC command n
/*
```

```
/&  
* $$ E0J
```

The ID parameter specifies the system ID of the TCP/IP partition to which you want to connect.

IPv6/VSE

IPv6/VSE provides a REXEC client through the BSTTREXC utility. BSTTREXC runs a single command or shell script. Example 6-16 shows the JCL and a tip for the use of a command continuation line. In BSTTREXC, the EXE2 command can be used to continue an EXEC statement. EXE2 must precede the EXEC command.

Example 6-16 IPv6/VSE batch REXEC client

```
// EXEC BSTTREXC,SIZE=BSTTREXC  
ID 00  
*  
OPEN ...  
*  
USER ...  
PASS ...  
CODE *  
EXE2 continued-text-goes-here  
EXEC gzip -v  
/ftphome/cabs/bhn/emi/arc/long-file-name-here-that-i-need-to-continue  
QUIT  
/*
```

In the EXEC line, the text must go out to column 80. This text is then continued on the EXE2 line. You can have one EXE2 command only. If this is not enough, use a script and run the script.

6.3 Performance

In this section, we compare TCP/IP for VSE/ESA to LFP in a z/VM environment, as shown in Table 6-1.

Table 6-1 Performance overview

Workload	TCP/IP for VSE/ESA	Linux Fast Path	Difference
FTP (BSI FTP server) VSE ? Linux (1 GB) (NULL file, no I/O)	19 MBps 29% CPU (5% App + 24% TCP/IP)	72 MBps 20% CPU (App)	3.7 times faster 9% less CPU
Linux ? VSE (1 GB) (NULL file, no I/O)	21 MBps 55% CPU (11% App + 44% TCP/IP)	70 MBps 20% CPU (App)	3.3 times faster 35% less CPU

Workload	TCP/IP for VSE/ESA	Linux Fast Path	Difference
Socket Application (running 3 times) VSE ? Linux (100 MB)	4.6 MBps (*3 = 13.8 MBps)	14.6 MBps (*3 = 43.8 MBps)	3.2 times faster
Linux ? VSE (100 MB)	9.7 MBps (*3 = 29.1 MBps) 26% CPU (3*1% App + 23% TCP/IP)	16.2 MBps (*3 = 48.6 MBps) 9 % CPU (3*3% App)	1.7 times faster 17% less CPU

The environment that was used is IBM System z10 EC (2097-722). The TCP/IP connection is through a shared OSA adapter.

The results show significant benefits in transfer rate and CPU usage, which results in reduced subcapacity Cost.

Note: For more information about IPv6/VSE throughput rates, see this website:

<http://bsitcpip.blogspot.com/2010/09/zvse-tcpip-throughput-rates.html>

6.4 SSL

Because TCP/IP for VSE/ESA and IPv6/VSE are using different SSL implementations, there are significant differences in using SSL.

TCP/IP for VSE/ESA

TCP/IP for VSE/ESA provides an SSL implementation as part of the CSI stack. The SSL API is described in *TCP/IP for VSE Programmer's Guide*, SC33-6766, and is compatible with the z/OS SSL API.

Most applications are SSL-enabled. However, the following SSL modes are provided:

- SSL in native mode

SSL traffic goes directly to the SSL enabled daemon on VSE. Native mode is supported by the FTP daemons, TELNETD, and HTTPd. In addition to one of these daemons, you must define a TLS D daemon on the same port. This daemon then gets all SSL-related settings from the related TLS D daemon.

- SSL in pass-through mode

This mode must be used for secure Telnet, but also can be used for HTTP. Also, here we must define a TLS D daemon. The difference from native mode is that we use the PASSPORT parameter to route SSL traffic from an unsecured daemon to the SSL daemon.

The preferred method to use is native mode.

IPv6/VSE

IPv6/VSE provides SSL support by using two proxy servers that translate encrypted traffic into clear traffic and vice versa. This means that applications are not natively SSL enabled and you must configure and start one or more SSL proxy servers.

The following server options are available:

- ▶ BSTTPRXY is a simple proxy server to proxy one application. To proxy multiple connections, you must run multiple BSTTPRXY partitions. BSTTPRXY performs IPv4-to-IPv6 or IPv6-to-IPv4 translations.
- ▶ BSTTATLS (Automatic Transport Layer Security) is a facility that is similar to the z/OS AT-TLS Security facility. It allows many ATTLS definitions and monitors incoming and outgoing connections. It also intercepts and converts sockets from clear text to SSL or vice versa, as necessary. However, BSTTATLS does not perform IPv4-to-IPv6 or IPv6-to-IPv4 translations.

For more information, see 3.7.4, “Deciding whether to use the SSL proxy server or AT-TLS” on page 80.

IPv6/VSE uses phase IJBSSL (OpenSSL) that is provided with z/VSE for SSL support. OpenSSL on z/VSE provides an SSL API that is compatible to the z/OS SSL API. For more information, see “z/OS SSL API” on page 221.

LFP

LFP does not provide an SSL implementation. Instead, SSL from TCP/IP for VSE/ESA can be used (which implies a licensing issue). SSL traffic then goes from any remote platform to Linux on System z, which forwards the encrypted traffic by using IUCV or HiperSockets Completion Queue (HSCQ) to z/VSE that is running under z/VM or in an LPAR. For more information, see 4.7, “Using secure connections with SSL” on page 146.

6.5 Comparison of APIs

This section provides an overview of the different APIs in terms of protocols (IPv4, IPv6), stacks (TCP/IP for VSE, IPv6/VSE, LFP), programming languages (Assembler, C), and SSL (TCP/IP for VSE/ESA, OpenSSL).

For more information about an API reference, see Appendix A, “API reference” on page 219.

6.5.1 Socket APIs

This topic gives an overview of the APIs that are provided by the different IP stacks. A common Assembler socket API (EZASOKET) is part of the z/VSE base. EZASOKET is described in *z/VSE V5R1 TCP/IP Support*, SC34-2640.

TCP/IP for VSE/ESA

TCP/IP for VSE/ESA provides an LE/C socket API through phase \$EDCTCPV. The API functions are described in *z/VSE V5R1 TCP/IP Support*, SC34-2640.

IPv6/VSE

IPv6/VSE provides a socket API for IPv4 (phase BSTTTCPV) and IPv4 and IPv6 (phase BSTTTCP6).

LFP

LFP provides a socket API with phase IJBLFPLE.

6.5.2 SSL APIs

All SSL APIs that are available on z/VSE are derived from the z/OS SSL API that is described in Chapter 9 of *z/OS Cryptographic Services, SSL Programming*, SC24-5901. Although this API is outdated on z/OS, it is supported for old applications. On z/VSE, this is still the standard SSL API.

TCP/IP for VSE/ESA

The SSL API that is provided by TCP/IP for VSE/ESA is identical to the former z/OS SSL API. The API with its CSI-specific characteristics is described in *TCP/IP for VSE Programmer's Guide*, SC33-6766. A good source for return and reason codes is the `sslvse.a` macro that is part of TCP/IP for VSE/ESA. It is in the TCP/IP for VSE/ESA installation sublibrary.

IPv6/VSE

IPv6/VSE does not provide its own SSL API. It uses the GSK-API provided by OpenSSL.

Linux Fast Path

No SSL API is provided by LFP. Instead, LFP uses the CSI SSL implementation.

6.5.3 Crypto APIs

In this section, we describe the key components of z/VSE networking and their support of cryptography.

TCP/IP for VSE/ESA

As of this writing, TCP/IP for VSE/ESA is the only product that provides an API for encryption, decryption, signing, and message authentication. For more information about this API (CryptoVSE API), see *TCP/IP for VSE Programmer's Guide*, SC33-6766.

Another API unique to the CSI stack is the Common Encryption Cipher Interface (CECI). It was developed to meet specific auditing requirements of the VISA CISP standard. It can be called from Assembler, COBOL, or other high-level languages that use standard call/save linkage conventions. The CECI interface is also described in the *TCP/IP for VSE Programmer's Guide*, SC33-6766.

IPv6/VSE

As of this writing, IPv6/VSE does not provide any cryptographic APIs.

Linux Fast Path

As of this writing, Linux Fast Path does not provide any cryptographic APIs.

OpenSSL

There are several native OpenSSL API functions for encryption and decryption that can be used from IJBSLVSE.OBJ. This requires the caller to be a C program with an LE-C runtime environment. For a list of provided API functions, see "OpenSSL API" on page 222.

6.6 Considerations for DB2 Server for VSE interfaces

By using the DB2 Server for VSE and the DB2 Client Edition for VSE, you can choose the following API for IBM DRDA® communication socket interfaces:

- ▶ For the DB2 Application Server component, you can choose between the LE/C and EZASMI socket interfaces.
- ▶ For the DB2 Batch and online Application Requestor, you can choose between the LE/C, EZASMI, and CSI Assembler socket interfaces.

For DB2, always use the EZASMI socket interface for best compatibility with the different TCP/IP stacks.

For more information, see Chapter 3 of *z/VSE Using DB2 on Linux for System z*, SG24-7690, which is available at this website:

<http://www.redbooks.ibm.com/abstracts/sg247690.html?Open>

Depending on which socket interface that is used, you must specify the following TCP/IP stack-specific interface phases:

- ▶ LE/C: Configure the LE/C multiplexer.
- ▶ EZASMI: Specify the sockets phase in JCL.
- ▶ CSI Assembler: No further configuration is needed.

6.7 Considerations for IBM applications

In this section, we describe how the different IP stacks influence the behavior of IBM applications.

6.7.1 VSE Connector Server

When the VSE Connector Server is used with IPv6/VSE, you must add the following statement to the server JCL:

```
// SETPARM SENDALL='YES'
```

This statement is needed because the implementation of a non-blocking socket `send()` is different in TCP/IP for VSE/ESA and IPv6/VSE. The SENDALL parameter enforces the same behavior for IPv6/VSE as it is implemented in TCP/IP for VSE/ESA.

Also, ensure that you have set up the LE/C multiplexer phase for your IP stack (skeleton EDCTCPMC in ICCF library 59).

6.7.2 Virtual tape

When z/VSE vtape is used, ensure that the LE/C multiplexer phase is correctly set up for your IP stack (skeleton EDCTCPMC in ICCF library 59). For IPv6/VSE, we suggest the setup that is shown in Example 6-17.

Example 6-17 Setup of VTAPE server for IPv6/VSE

```
* $$ JOB JNM=TAPESEVR,CLASS=S,DISP=L
* $$ LST CLASS=0,PURGE=0
// JOB TAPESEVR START UP VSE TAPE SERVER
```

```
// ID USER=VCSRVR,PWD=VCSRVR
// OPTION PARTDUMP,NOSYSDUMP
// OPTION SYSPARM='00'
// SETPARM IPTRACE='NNNNNNNN'
// SETPARM SVABUF=YES
// SETPARM SENDALL=YES
// SETPARM LRGBUF=YES
// LIBDEF *,SEARCH=(PRD2.TCPIPB,PRD2.CONFIG,PRD1.BASE,PRD2.SCEEBASE)
// EXEC $VTMAIN
/*
/ &
```

The following points concerning JCL should be considered:

- ▶ Adjust the CLASS=S on the * \$\$ LST card, if needed.
- ▶ PURGE=0 automatically deletes any LST queue output if RC=0.
- ▶ // SETPARM IPTRACE='NNNNNNNN' disables any tracing.
- ▶ // SETPARM SVABUF=YES. This is needed by IPv6/VSE because the VSE vtape support uses SVA socket buffers.
- ▶ // SETPARM SENDALL=YES. This is needed by IPv6/VSE.
- ▶ // SETPARM LRGBUF=YES. This uses scaled TCP Windows.
- ▶ IPv6/VSE lib.slib must be first in // **LIBDEF PHASE,SEARCH** chain.

Note: If you enable tracing by changing the IPTRACE values, you must also enter DEBUG ON on the console to get SYSLST output.

6.7.3 CICS Web Support

When a TCP/IP service is defined for CICS Web Support (CWS), the TCP/IP host name must be made known to CICS.

TCP/IP for VSE/ESA

When TCP/IP for VSE/ESA is used, you define the TCP/IP host name in the IPINIT member, as shown in the following example:

```
DEFINE NAME,NAME=MYVSE,IPADDR=9.152.131.189
```

For more information about configuring CWS by using TCP/IP for VSE/ESA, see *Security on IBM z/VSE*, SG24-7691.

IPv6/VSE

When IPv6/VSE is used, you use a HOST command in the BSTTINET or BSTT6NET startup JCL, as shown in the following example:

```
HOST MYVSE 9.152.131.189
```

In addition to specifying the host name in the stack startup JCL, you must modify your CICS startup JCL to support CWS, as shown in the following example:

```
// JOB CICSBSI                CICS/ICCF STARTUP
// OPTION SYSPARM='nn'        <-- SYSID of BSI Stack
// SETPARM LOCALGT=YES
...
```

```
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD1.BASED,PRD2.TCIPB,PRD1.BASE, ...
```

For more information, see *IPv6/VSE Migration Guide*, SC34-2624.

The LOCALGT parameter is only used for performance. For more information, see *IPv6/VSE Programming Guide*, SC34-2617.

LFP

When LFP is used, the TCP/IP host name is the host name of the Linux system that is running the LFP daemon.

6.7.4 Encryption Facility for z/VSE

Encryption Facility for z/VSE is an optionally priced feature of z/VSE V4 and later. It provides encryption of VSE data sets and tapes and includes the following main functions:

- ▶ Password-based encryption (PBE). When PBE is used, the encryption key is derived from a user-specified secret passphrase.
- ▶ Public-key encryption (PKE). When PKE is used, the encryption key is created by using a random number generator and then protected by an RSA public key. Only the possessor of the corresponding private key can decrypt the encrypted data.

Note: As of this writing, PKE is supported by TCP/IP for VSE/ESA only. Encryption Facility uses z/VSE services for accessing RSA keys and only the CSI format (PRVK and CERT) is supported. Supporting IPv6/VSE requires enhancing the z/VSE crypto services to support PEM keystores.

For more information about Encryption Facility for z/VSE, see *z/VSE Administration*, SC34-2627.

6.7.5 Basic Security Manager

The z/VSE Basic Security Manager (BSM) provides security exit BSSTISX for accessing security-related information, such as user passwords.

TCP/IP for VSE/ESA

To enable TCP/IP for access to BSM, use the following commands:

```
DEFINE SECURITY,DIVER=BSSTISX
SET SECURITY = ON
```

You can display the current security-related settings by using the **Q SECURITY** command, as shown in the following example:

```
F7-0101 IPN300A Enter TCP/IP Command
101 q security
F7 0097 IPN253I << TCP/IP Security Settings >>
F7 0097 IPN750I Security Processing: Enabled
F7 0097 IPN750I ARP Checking: Disabled
F7 0097 IPN750I IP Address Checking: Disabled
F7 0097 IPN751I Exit data: Undefined
F7 0097 IPN750I Automatic Security: Disabled
F7 0097 IPN750I Security Exit: Enabled - BSSTISX
F7 0097 IPN750I Batch Security: Disabled
```

IPv6/VSE

The BSI FTP server security exit routine BSTTFTS1.PHASE calls the BSSTISX security exit to verify user ID and password. All other security is controlled by using the standard BSTTSCTY.T member.

Note: The default FTP server security member (BSTTSCTY.T) provides open access to the FTP server, so any user ID and password are accepted.

To enable the BSTTFTS1.PHASE security exit, complete the following tasks:

- ▶ Copy the BSTTFTS1.PHASE to a configuration `lib.slib` as BSTTFTSX.PHASE.
- ▶ Use a LIBDEF statement to reference this configuration `lib.sublib` in the BSTTFTPS PHASE,SEARCH chain.
- ▶ Add the following BSSTISX command to your BSTTFTPS startup commands:

```
BSSTISX <data>
<data> is [anonym_uid] [, [anonym_pwd] [, [preproc] [, [postproc] [, [mode]]]]]
```

IPv6/VSE does not provide any command to display the settings in the BSTTSCTY.T member. Use LIBR to display its contents.

For more information about configuring security for IPv6/VSE, see the following resources:

- ▶ *IPv6/VSE IPv4 User's Guide*, which is available at this website:
<http://public.dhe.ibm.com/s390/zos/vse/pdf3/ipv6vse/iesipu41.pdf>
- ▶ *IPv6/VSE Migration Guide*, SC34-2624

LFP

As of this writing, only one IBM application is provided specially for the LFP stack: The z/VSE Virtual FTP Server. This server uses the security mechanism of the VSE Connector Server, which, in turn, accesses the BSM by using RACROUTE requests. Therefore, BSSTISX is not relevant for LFP.

6.7.6 Uploading PTF files to IJSYSPF

This section shows how to upload PTF files to the PTF service file IJSYSPF for further application from disk. You should never apply multiple PTFs by just submitting the PTF jobs to the RDR queue. In this case, MSHP would not be able to apply the PTFs in their correct order, considering all prerequisites and corequisites. In addition to that, some PTFs contain VSE/POWER JCL for generating child jobs to be placed in the RDR queue. This can only be handled correctly using the Interactive Interface apply PTF dialogs and having all PTFs available on disk.

Note: In z/VSE 5.2, the logical record length for IJSYSPF is 10320. In previous versions of z/VSE, it was 80. Be sure to use the correct LRECL value. You might redefine the PTF file by using the skeleton SKPTFILE in ICCF library 59.

TCP/IP for VSE/ESA

Uploading PTFs to IJSYSPF by using TCP/IP for VSE/ESA is described in *System Upgrade and Service*, SC34-2639. It is also described on the VSE home page in the Service &

Support section. Example 6-18 shows the FTP commands used to transfer the binary PTF file with correct record length.

Example 6-18 Upload PTF file to IJSYSPF using TCP/IP for VSE/ESA

```
C:\Temp>ftp 9.164.170.30          <-- IP address or hostname of VSE system
Connected to 9.164.170.30
220-TCP/IP for VSE -- Version 01.04.00 -- FTP Daemon
Copyright (c) 1995,2001 Connectivity Systems Incorporated
220 Service ready for new user.
User (9.164.155.2:(none)): SYSA    <-- enter your user id here
331 User name okay, need password.
Password: <-- enter your password here
230 User logged in, proceed.
ftp> binary                      <-- switch to binary mode
200 Command okay.
ftp> quote site lrec1 10320       <-- record size of your file
200 Command okay.
ftp> quote site recfm f           <-- record format of your file
200 Command okay.
ftp> put ptffile.bin PTF.FILE <-- enter your filenames
```

IPv6/VSE

Example 6-19 shows the FTP commands for uploading a PTF file to the IJSYSPF file using IPv6/VSE.

Example 6-19 Upload PTF file to IJSYSPF using IPv6/VSE

```
// EXEC BSTTFTPC,SIZE=BSTTFTPC
ID ..
USER ..
PASS ..
CWD ... if necessary ...
*
TYPE I
OUTPUT ESDS IJSYSPF RECSZ 80
RETR eptfxxxx.bin
QUIT
/*
```

The IPv6/VSE FTP server does not allow specifying a block size, so ensure that IJSYSPF is defined with the correct block size depending on your z/VSE release.

Linux Fast Path

When using LFP, you will probably use the Virtual z/VSE FTP Daemon. Example 6-20 shows how to upload a binary PTF file for this scenario.

Example 6-20 Upload PTF file to IJSYSPF using the Virtual z/VSE FTP Daemon

```
D:\ftp 127.0.0.1
Connected to 127.0.0.1.
220 IBM Virtual z/VSE FTP Server on BR9WG7D4 (version 1.1) ready to serve.
User (127.0.0.1:(none)): jsch
331 Password required for jsch.
Password:
230 User jsch logged in. Idle timeout is 15 minutes.
```

```
ftp> bin
200 Type set to I.
ftp> cd vsam
250 CWD command successful, current directory is "/VSAM".
ftp> cd VSESP.USER.CATALOG
250 CWD command successful, current directory is "/VSAM/VSESP.USER.CATALOG".
ftp> put UD54054.bin ptf.file
```

The Virtual z/VSE FTP Server allows you to specify transfer mode S (stream, which is the default) or B (block). When using stream mode, the target file is filled record by record according to the VSAM cluster definition. It is not needed to specify a specific record length or format.

Note: This support requires updates in the VSE Connector Server and Virtual z/VSE FTP Server from September 2014 or later.

6.8 Known problem: ftp.exe hangs on Windows 7

In our test setup, we found that ftp.exe hangs on Windows 7.

6.8.1 Symptom

When ftp.exe is used under Windows 7 Professional, file transfers and directory lists are not possible. Logon works fine, but there is a hang when we tried to upload or download files.

6.8.2 Solution

Complete the follow steps to correct the problem:

1. Open the Windows 7 Control Panel, then click **Windows Firewall** → **Advanced Settings** → **Inbound Rules**.
2. Scroll down until you see a rule that is called "File Transfer Program". Complete the following steps:
 - a. Double-click the rule.
 - b. On the General tab, under Action, select **Allow the connectionB**.
 - c. Click **OK**.
3. If there is no such rule, click **New rule...** and complete the following steps in the wizard:
 - a. In the first window, select **Program**.
 - b. In the second window, browse to the ftp.exe, which is in the C:\windows\system32 directory.
 - c. In the third window, select **Allow the connection**.
 - d. Keep the defaults for the remaining windows.
 - e. Click **OK**.



A

API reference

This appendix describes TCP/IP-related application programming interfaces (APIs) that can be used under z/VSE. We distinguish between the following types of APIs:

- ▶ **Socket APIs**

There are socket functions for IPv4 and for IPv6. Also, there are some stack-dependent differences.

- ▶ **Secure socket (SSL) APIs**

The z/OS compatible GSK interface is supported by CSI's SSL implementation and by OpenSSL. In addition, native OpenSSL functions can be used with the IJBSLVSE.OBJ file.

Socket APIs

z/VSE and the IP stacks that are available for z/VSE provide the following socket application programming interfaces:

- ▶ EZA interfaces

These interfaces are widely compatible with the corresponding z/OS interfaces and are supported by TCP/IP for VSE/ESA, IPv6/VSE, and Linux Fast Path (LFP). They are based on the EZASMI macro interface for HLASM programmers and the EZASOCKET call interface for COBOL, PL/I, and HLASM programmers.

Update with z/VSE 5.2: With Build 255pre02, IPv6/VSE provides an update to its EZA/GSK API so that an LE environment is automatically and dynamically established. This allows all standard types of batch and CICS applications to use the GSK API (and OpenSSL).

- ▶ TCP/IP APIs that use IBM Language Environment for z/VSE

Language Environment based interfaces include the Language Environment/VSE 1.4 C socket interface and the REXX/VSE Socket API support within REXX/VSE.

- ▶ TCP/IP for VSE/ESA native APIs

Native TCP/IP for VSE/ESA interfaces include the Assembler SOCKET macro interface, the COBOL and PL/I preprocessor interface, the BSD-C socket interface, and the REXX socket APIs.

For more information about APIs, see these resources:

- ▶ *z/VSE TCP/IP Support*, SC34-2640, which is available at this website:

<http://www.ibm.com/systems/z/os/zvse/documentation/#tcpip>

- ▶ 1.3.9, “Overview of APIs” on page 21

SSL APIs

This section, we describe the SSL-related API functions that are provided by TCP/IP for VSE/ESA and OpenSSL. Applications must link the IJBSLVSE.OBJ to get access to the API functions. The IJBSLVSE.OBJ file is part of OpenSSL on z/VSE. For more information, see 5.1.1, “What is available on z/VSE” on page 152.

z/OS SSL API

In this topic, we describe the z/OS SSL API. This API is supported by TCP/IP for VSE/ESA by using the \$EDCTCPV phase and by OpenSSL by using phase IJBSSL.

For more information about this API, see the following resources:

1. *TCP/IP for VSE Programmer's Guide*, which is provided by CSI. This book describes the API from the CSI perspective.
2. *z/OS Cryptographic Services, SSL Programming*, SC24-5901. This book describes the API from the z/OS perspective and is applicable for OpenSSL.

In this appendix, we describe only the differences of the API functions that are caused by the implementation of the OpenSSL-to-GSK layer.

gsk_free_memory

Releases storage that is allocated by the SSL run time.

No change for z/VSE compared to z/OS.

gsk_get_cipher_info

Returns the supported cipher specifications.

z/VSE features the following changes:

- The list of returned cipher suites differs from what is documented in the z/OS book because some of the ciphers that are used on z/OS are not supported by OpenSSL (for example, “00”). With APAR DY47545 OpenSSL on z/VSE returns the following strings:
"091512060201" // LOW_SECURITY
"C027C014C013C0126B67393316153D3C3B352F0A09" // HIGH_SECURITY

The first four ECDHE-RSA cipher suites have 4-digit names that start with C0. They are followed by the 2-character DHE-RSA ciphers 6B, 67, 39, 33, 16, and 15.

Cipher suites 3D, 3C, and 3B are part of the TLSv1.2 support that is included since OpenSSL 1.0.1e.

For compatibility with an earlier version reasons, the list still returns the older cipher suites 35, 2F, 0A, and 09. However, from today's perspective, at least 09 and 0A are no longer considered to provide “high security”.

- The version field of the `gsk_sec_level` struct returns the supported OpenSSL version (for example, 101 = 1.0.1).

gsk_get_dn_by_label

Gets the distinguished name for a certificate.

z/VSE features the following changes:

- ▶ The specified key or cert file must be a Librarian member with member type PEM or a VSAM file.
- ▶ In z/OS, they return NULL if the key database cannot be accessed. However, in VSE, we do not have enough information to access the keystore.

gsk_initialize

Starts the System SSL runtime environment.

In z/VSE, read and evaluate the JCL variables SSL\$DBG and SSL\$ICA changed.

gsk_secure_soc_close

Closes a secure socket connection. There was no change for z/VSE.

gsk_secure_soc_init

Starts a secure socket connection. There was no change for z/VSE.

gsk_secure_soc_read

Reads data by using a secure socket connection.

In z/VSE, the changes included the fact that the caller can specify `buflen = 0` to check for pending bytes. When `buflen = 0`, `SSL_pending` is called and `gsk_secure_soc_read` returns the return code of `SSL_pending`.

gsk_secure_soc_reset

Resets the session keys for a secure connection. There was no change for z/VSE.

gsk_secure_soc_write

Writes data by using a secure socket connection. There was no change for z/VSE.

gsk_uninitialize

Ends the SSL environment. There was no change for z/VSE.

gsk_user_set

Sets an application callback. As of this writing, this is not supported.

OpenSSL API

In this section, we describe the native OpenSSL API functions that are supported on z/VSE, which means that they can be used by a user application with IJBSLVSE.OBJ.

The full OpenSSL API is described at this website:

<http://www.openssl.org/docs/>

Therefore, we do not add any API description.

Note: The following VSE-specific functions allow switching between the GSK API and the OpenSSL API:

```
ssl_enable_gsk()
ssl_disable_gsk()
```

The following native OpenSSL functions are provided by z/VSE 5.1, APAR DY47499:

- ▶ AES_encrypt
- ▶ AES_set_encrypt_key
- ▶ BIO_ctrl
- ▶ BIO_ctrl_get_read_request
- ▶ BIO_ctrl_get_write_guarantee
- ▶ BIO_ctrl_pending
- ▶ BIO_f_base64
- ▶ BIO_f_ssl
- ▶ BIO_free
- ▶ BIO_free_all
- ▶ BIO_new
- ▶ BIO_new_bio_pair
- ▶ BIO_new_dgram
- ▶ BIO_new_fp
- ▶ BIO_new_mem_buf
- ▶ BIO_new_socket
- ▶ BIO_nread
- ▶ BIO_nwrite
- ▶ BIO_nwrite0
- ▶ BIO_printf
- ▶ BIO_push
- ▶ BIO_read
- ▶ BIO_s_mem
- ▶ BIO_set_flags
- ▶ BIO_snprintf
- ▶ BIO_test_flags
- ▶ BIO_write
- ▶ BN_CTX_free
- ▶ BN_CTX_get
- ▶ BN_CTX_new
- ▶ BN_CTX_start
- ▶ BN_add_word
- ▶ BN_bin2bn
- ▶ BN_bn2bin
- ▶ BN_bn2dec
- ▶ BN_clear_free
- ▶ BN_cmp
- ▶ BN_copy
- ▶ BN_dec2bn
- ▶ BN_div
- ▶ BN_dup
- ▶ BN_free
- ▶ BN_hex2bn
- ▶ BN_is_bit_set
- ▶ BN_lshift
- ▶ BN_mask_bits
- ▶ BN_new
- ▶ BN_num_bits
- ▶ BN_print_fp
- ▶ BN_rand
- ▶ BN_set_word
- ▶ BN_sub
- ▶ BN_value_one
- ▶ CRYPTO_cleanup_all_ex_data
- ▶ CRYPTO_dbg_set_options
- ▶ CRYPTO_free
- ▶ CRYPTO_lock
- ▶ CRYPTO_mem_ctrl
- ▶ CRYPTO_mem_leaks
- ▶ CRYPTO_mem_leaks_fp
- ▶ CRYPTO_set_locking_callback
- ▶ CRYPTO_set_mem_debug_functions
- ▶ CRYPTO_set_mem_debug_options
- ▶ CRYPTO_thread_id
- ▶ DES_is_weak_key
- ▶ DH_compute_key
- ▶ DH_free
- ▶ DH_generate_key
- ▶ DH_new
- ▶ DH_size
- ▶ DHparams_print_fp
- ▶ DSA_SIG_free
- ▶ DSA_SIG_new
- ▶ DSA_do_sign
- ▶ DSA_do_verify
- ▶ DSA_free
- ▶ DSA_generate_key
- ▶ DSA_generate_parameters_ex
- ▶ DSA_new
- ▶ DSA_print_fp
- ▶ ECDH_compute_key
- ▶ ECDSA_SIG_free
- ▶ ECDSA_SIG_new
- ▶ ECDSA_do_sign
- ▶ ECDSA_do_verify
- ▶ EC_GROUP_cmp
- ▶ EC_GROUP_free
- ▶ EC_GROUP_get_curve_name
- ▶ EC_GROUP_get_degree
- ▶ EC_GROUP_get_order
- ▶ EC_GROUP_method_of
- ▶ EC_GROUP_new_by_curve_name
- ▶ EC_GROUP_set_asn1_flag
- ▶ EC_KEY_free

- ▶ EC_KEY_generate_key
- ▶ EC_KEY_get0_group
- ▶ EC_KEY_get0_private_key
- ▶ EC_KEY_get0_public_key
- ▶ EC_KEY_new_by_curve_name
- ▶ EC_KEY_set_asn1_flag
- ▶ EC_KEY_set_group
- ▶ EC_KEY_set_public_key
- ▶ EC_METHOD_get_field_type
- ▶ EC_POINT_clear_free
- ▶ EC_POINT_cmp
- ▶ EC_POINT_free
- ▶ EC_POINT_get_affine_coordinates_GFp
- ▶ EC_POINT_is_at_infinity
- ▶ EC_POINT_mul
- ▶ EC_POINT_new
- ▶ EC_POINT_oct2point
- ▶ EC_POINT_point2oct
- ▶ ENGINE_load_builtin_engines
- ▶ ENGINE_register_all_complete
- ▶ ERR_clear_error
- ▶ ERR_error_string
- ▶ ERR_free_strings
- ▶ ERR_get_error
- ▶ ERR_get_error_line_data
- ▶ ERR_load_crypto_strings
- ▶ ERR_print_errors
- ▶ ERR_print_errors_fp
- ▶ ERR_remove_state
- ▶ EVP_CIPHER_CTX_cleanup
- ▶ EVP_CIPHER_CTX_get_app_data
- ▶ EVP_CIPHER_CTX_init
- ▶ EVP_CIPHER_CTX_iv_length
- ▶ EVP_CIPHER_CTX_key_length
- ▶ EVP_CIPHER_CTX_set_app_data
- ▶ EVP_CIPHER_CTX_set_key_length
- ▶ EVP_CIPHER_CTX_set_padding
- ▶ EVP_CIPHER_block_size
- ▶ EVP_CIPHER_key_length
- ▶ EVP_CIPHER_nid
- ▶ EVP_Cipher
- ▶ EVP_CipherInit
- ▶ EVP_DecryptFinal_ex
- ▶ EVP_DecryptInit_ex
- ▶ EVP_DecryptUpdate
- ▶ EVP_Digest
- ▶ EVP_DigestFinal
- ▶ EVP_DigestFinal_ex
- ▶ EVP_DigestInit
- ▶ EVP_DigestInit_ex
- ▶ EVP_DigestUpdate
- ▶ EVP_EncryptFinal_ex
- ▶ EVP_EncryptInit_ex
- ▶ EVP_EncryptUpdate
- ▶ EVP_MD_CTX_cleanup
- ▶ EVP_MD_CTX_init
- ▶ EVP_MD_size
- ▶ EVP_MD_type
- ▶ EVP_PKEY_free
- ▶ EVP_PKEY_get1_DSA
- ▶ EVP_PKEY_get1_EC_KEY
- ▶ EVP_PKEY_get1_RSA
- ▶ EVP_aes_128_cbc
- ▶ EVP_aes_192_cbc
- ▶ EVP_aes_256_cbc
- ▶ EVP_bf_cbc
- ▶ EVP_cast5_cbc
- ▶ EVP_cleanup
- ▶ EVP_des_cbc
- ▶ EVP_des_ede3_cbc
- ▶ EVP_enc_null
- ▶ EVP_get_cipherbyname
- ▶ EVP_get_digestbyname
- ▶ EVP_md5
- ▶ EVP_rc4
- ▶ EVP_sha1
- ▶ EVP_sha256
- ▶ EVP_sha512
- ▶ GENERAL_NAME_free
- ▶ HMAC
- ▶ HMAC_CTX_cleanup
- ▶ HMAC_CTX_init
- ▶ HMAC_Final
- ▶ HMAC_Init
- ▶ HMAC_Update
- ▶ MD5_Final
- ▶ MD5_Init
- ▶ MD5_Update
- ▶ OBJ_cmp
- ▶ OBJ_nid2ln
- ▶ OBJ_nid2sn
- ▶ OBJ_obj2nid
- ▶ OPENSSL_add_all_algorithms_noconf
- ▶ OpenSSL_add_all_ciphers
- ▶ OpenSSL_add_all_digests
- ▶ PEM_read_PrivateKey
- ▶ PEM_read_X509
- ▶ PEM_read_bio_PrivateKey
- ▶ PEM_write_bio_DSAPrivateKey
- ▶ PEM_write_bio_ECPrivateKey
- ▶ PEM_write_bio_RSAPrivateKey
- ▶ PKCS7_free
- ▶ RAND_bytes
- ▶ RAND_seed
- ▶ RAND_status
- ▶ RC4
- ▶ RC4_set_key
- ▶ RSA_blinding_on
- ▶ RSA_free
- ▶ RSA_generate_key_ex

- ▶ RSA_new
- ▶ RSA_print_fp
- ▶ RSA_private_decrypt
- ▶ RSA_private_encrypt
- ▶ RSA_public_decrypt
- ▶ RSA_public_encrypt
- ▶ RSA_sign
- ▶ RSA_size
- ▶ SHA1_Final
- ▶ SHA1_Init
- ▶ SHA1_Update
- ▶ SHA256_Final
- ▶ SHA256_Init
- ▶ SHA256_Update
- ▶ SSL_CIPHER_get_name
- ▶ SSL_CIPHER_get_version
- ▶ SSL_CTX_callback_ctrl
- ▶ SSL_CTX_ctrl
- ▶ SSL_CTX_free
- ▶ SSL_CTX_get_cert_store
- ▶ SSL_CTX_load_verify_locations
- ▶ SSL_CTX_new
- ▶ SSL_CTX_set_cert_verify_callback
- ▶ SSL_CTX_set_cipher_list
- ▶ SSL_CTX_set_default_verify_paths
- ▶ SSL_CTX_set_info_callback
- ▶ SSL_CTX_set_session_id_context
- ▶ SSL_CTX_set_tmp_rsa_callback
- ▶ SSL_CTX_set_verify
- ▶ SSL_CTX_use_PrivateKey_file
- ▶ SSL_CTX_use_certificate_file
- ▶ SSL_SESSION_get_id
- ▶ SSL_ctrl
- ▶ SSL_do_handshake
- ▶ SSL_free
- ▶ SSL_get_current_cipher
- ▶ SSL_get_error
- ▶ SSL_get_peer_certificate
- ▶ SSL_get_servername
- ▶ SSL_get_servername_type
- ▶ SSL_get_version
- ▶ SSL_library_init
- ▶ SSL_load_error_strings
- ▶ SSL_new
- ▶ SSL_pending
- ▶ SSL_read
- ▶ SSL_set_accept_state
- ▶ SSL_set_bio
- ▶ SSL_set_connect_state
- ▶ SSL_set_session
- ▶ SSL_set_verify
- ▶ SSL_state
- ▶ SSL_state_string
- ▶ SSL_state_string_long
- ▶ SSL_version
- ▶ SSL_write
- ▶ SSLeay
- ▶ SSLeay_version
- ▶ SSLv23_method
- ▶ SSLv2_method
- ▶ SSLv3_method
- ▶ TLSv1_method
- ▶ X509_LOOKUP_ctrl
- ▶ X509_LOOKUP_file
- ▶ X509_LOOKUP_hash_dir
- ▶ X509_NAME_add_entry_by_txt
- ▶ X509_NAME_entry_count
- ▶ X509_NAME_free
- ▶ X509_NAME_get_entry
- ▶ X509_NAME_new
- ▶ X509_NAME_online
- ▶ X509_NAME_print_ex
- ▶ X509_STORE_CTX_free
- ▶ X509_STORE_CTX_get_ex_data
- ▶ X509_STORE_CTX_get_ex_new_index
- ▶ X509_STORE_CTX_init
- ▶ X509_STORE_CTX_new
- ▶ X509_STORE_CTX_set_ex_data
- ▶ X509_STORE_CTX_set_flags
- ▶ X509_STORE_add_lookup
- ▶ X509_STORE_free
- ▶ X509_STORE_new
- ▶ X509_STORE_set_flags
- ▶ X509_free
- ▶ X509_get_ext_d2i
- ▶ X509_get_issuer_name
- ▶ X509_get_pubkey
- ▶ X509_get_serialNumber
- ▶ X509_get_subject_name
- ▶ X509_print
- ▶ X509_verify_cert
- ▶ X509_verify_cert_error_string
- ▶ apps_ssl_info_callback
- ▶ ascii2ebcdic
- ▶ d2i_PKCS7
- ▶ d2i_PrivateKey
- ▶ d2i_X509
- ▶ d2i_X509_NAME
- ▶ i2d_PrivateKey
- ▶ i2d_PublicKey
- ▶ i2d_X509
- ▶ i2d_X509_NAME
- ▶ i2t_ASN1_OBJECT
- ▶ load_cert
- ▶ load_key
- ▶ set_cert_key_stuff
- ▶ sk_num
- ▶ sk_pop_free
- ▶ sk_value
- ▶ verify_callback

In addition to these OpenSSL functions, the following z/VSE-specific functions are provided:

- ▶ `ssl_disable_debug`
- ▶ `ssl_disable_gsk`
- ▶ `ssl_disable_ibmca`
- ▶ `ssl_enable_debug`
- ▶ `ssl_enable_gsk`
- ▶ `ssl_enable_ibmca`

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Note that some publications that are referenced in this list might be available in softcopy only:

- ▶ *Introduction to the New Mainframe: Security*, SG24-6776
- ▶ *Introduction to the New Mainframe: z/VSE Basics*, SG24-7436
- ▶ *OSA-Express Implementation Guide*, SG24-5948
- ▶ *Personal Communications Version 4.3 for Windows 95, 98 and NT*, SG24-4689
- ▶ *Security on IBM z/VSE*, SG24-7691

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft, and other materials at this website:

<http://www.ibm.com/redbooks>

Other publications

The following publications also are relevant as other information sources:

- ▶ *Personal Communications for Windows, Administrator's Guide and Reference*, SC31-8840
- ▶ IPv6/VSE:
 - *IPv6/VSE Installation Guide*, SC34-2616
 - *IPv6/VSE IPv4 User's Guide*, which is available at this website:
<ftp://public.dhe.ibm.com/s390/zos/vse/pdf3/ipv6vse/iesipu41.pdf>
 - *IPv6/VSE Licensed Programming Specifications*, GC33-8347
 - *IPv6/VSE Messages and Codes*, SC34-2619
 - *IPv6/VSE Migration Guide*, SC34-2624
 - *IPv6/VSE Program Directory*, GI11-9702
 - *IPv6/VSE Programming Guide*, SC34-2617
 - *IPv6/VSE SSL Installation, Programming and User's Guide*, SC34-2676
 - *TCP/IP-TOOLS Installation Guide*, SC34-2620
 - *TCP/IP-TOOLS User's Guide*, SC34-2621
 - *IPv6/VSE User's Guide*, SC34-2618

- ▶ TCP/IP for VSE/ESA:
 - *TCP/IP for VSE V1R5.0 Command Reference*, SC33-6764
 - *TCP/IP for VSE V1R5.0 Installation Guide*, SC33-6762
 - *TCP/IP for VSE V1R5.0 Optional Features*, SC33-6767
 - *TCP/IP for VSE V1R5.0 Programmer's Guide*, SC33-6766
 - *TCP/IP for VSE V1R5.0 Users Guide*, SC33-6763
 - *TCP/IP for VSE/ESA - IBM Program Setup and Supplementary Information*, SC33-6601
- ▶ z/VSE:
 - *z/VSE Administration*, SC33-8304
 - *z/VSE Operation*, SC33-8309
 - *z/VSE Planning*, SC33-8301
 - *z/VSE System Upgrade and Service*, SC34-2639
 - *z/VSE V5R1 e-business Connectors User's Guide*, SC34-2629
 - *z/VSE V5R1 TCP/IP Support*, SC34-2640

Online resources

The following websites also are relevant as other information sources:

- ▶ z/VSE home page with links to other documentation:
<http://www.ibm.com/zvse/>
- ▶ Keyman/VSE tool and VSE Connector Client:
<http://www.ibm.com/systems/z/os/zvse/downloads/>
- ▶ z/VSE virtual FTP daemon:
<http://www.ibm.com/systems/z/os/zvse/downloads/#vseftp>
- ▶ TCP/IP for VSE home page:
 - <http://www.e-vse.com/>
 - <http://www.csi-international.com/products/zVSE/TCP-IP/TCP-IP.htm/>
- ▶ Barnard Software Inc. home page:
<http://www.bsiopti.com/>
- ▶ The BSI blog:
<http://bsitcpip.blogspot.com/>
- ▶ TCP/IP and IPv6 related information on the VSE home page:
<http://www.ibm.com/systems/z/os/zvse/products/tcpip.html>
- ▶ z/VSE V5R1 TCP/IP Support, SC34-2640:
<http://www.ibm.com/systems/z/os/zvse/documentation/#tcpip>
- ▶ OpenSSL website:
<http://www.openssl.org/>
- ▶ Installing OpenSSL on Windows:
<http://www.slproweb.com/products/Win32OpenSSL.html>

- ▶ Filezilla server:
<http://filezilla-project.org/>
- ▶ Personal Communications Administrator's Guide and Reference, SC31-8840:
ftp://ftp.software.ibm.com/software/network/pcomm/publications/pcomm_57/pcadmin.pdf
- ▶ Online Administration Guide for Personal Communications:
http://publib.boulder.ibm.com/infocenter/pcomhelp/v5r9/index.jsp?topic=/com.ibm.pcomm.doc/books/html/admin_guide13.htm
- ▶ Open Text website:
<http://connectivity.opentext.com/>
- ▶ Microsoft Windows netsh commands for Interface IPv6:
<http://technet.microsoft.com/en-us/library/cc740203%28WS.10%29.aspx>
- ▶ Linux man pages example for TCP forwarding:
http://www.kernel.org/doc/man-pages/online/pages/man2/select_tut.2.html
- ▶ TN3270E printing:
http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp?topic=/com.ibm.zos.s.znetwork/znetwork_268.htm
- ▶ wc3270 Terminal Emulation:
<http://x3270.bgp.nu>
- ▶ z/VM Connectivity Guide:
<http://www.vm.ibm.com/pubs/>
- ▶ *z/OS Cryptographic Services, SSL Programming*, SC24-5901:
<http://publib.boulder.ibm.com/infocenter/zos/v1r12/index.jsp?topic=%2Fcom.ibm.zos.r12.gskal00%2Fgskala80.htm>

Help from IBM

IBM Support and downloads:

<http://www.ibm.com/support>

IBM Global Services:

<http://www.ibm.com/services>



Enhanced Networking on IBM z/VSE

(0.2"spine)
0.17"<->0.473"
90<->249 pages



Redbooks®

Enhanced Networking on IBM z/VSE

Use LFP on Z/VM and in LPAR for selected applications

Experience the benefits of OpenSSL

Learn how to use IPv6 on z/VSE

The importance of modern computer networks is steadily growing as increasing amounts of data are exchanged over company intranets and the Internet. Understanding current networking technologies and communication protocols that are available for the IBM mainframe and System z operating systems is essential for setting up your network infrastructure with IBM z/VSE.

This IBM Redbooks publication helps you install, tailor, and configure new networking options for z/VSE that are available with TCP/IP for VSE/ESA, IPv6/VSE, and Fast Path to Linux on System z (Linux Fast Path). We put a strong focus on network security and describe how the new OpenSSL-based SSL runtime component can be used to enhance the security of your business.

This IBM Redpaper Redbooks publication extends the information that is provided in Security on IBM z/VSE, SG24-7691.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8091-01

ISBN 0738440345