Optional Features Guide

TCP/IP for VSE

Version 2 Release 2

TCP/IP is a communications facility that permits bi-directional communication between VSE-based software and software running on other platforms equipped with TCP/IP.

This manual describes the optional features available with TCP/IP FOR VSE.

Published October 2017

CSI INTERNATIONAL

"Delivering what the competition can only promise"

www.csi-international.com • info@csi-international.com • 800.795.4914

TCP/IP FOR VSE Optional Features Guide
Version 2 Release 2
October 2017

Published by CSI International

Phone:	800-795-4914
Fax:	740-986-6022
Internet:	http://www.csi-international.com
Product questions:	info@csi-international.com
Technical support:	support@csi-international.com
Review comments:	documentation@csi-international.com

CSI International Technical Support

During Business Hours

Monday through Friday, 9:00 A.M. through 5:00 P.M. EST/EDT.

Telephone:	Toll Free in the USA	800-795-4914
	Worldwide	740-420-5400
Email:	support@csi-international.com	
Web:	http://csi-international.com/problemreport_vse.htm	

Emergency Service 24/7

After business hours and 24 hours on Saturday and Sunday:

Telephone:	Toll Free in the USA	800-795-4914
	Worldwide	740-420-5400

CSI International provides support to address each issue according to its severity.

Updates to This Manual

The following table describes updates to this manual. Updates may be associated with an incident in CSI International's support database.

October 2017

ID	Change Description	Page
	Ch. 2, TLS/SSL for VSE:	
	Updated the application-setup procedure.	25
	DEFINE TLSD command parameters:	
	CIPHER: Added cipher suites for the TLS 1.2 protocol.	38
	MINVERS: Added the TLS 1.2 protocol version.	39
	Ch. 3, SecureFTP for VSE:	
	FTPBATCH server—SET command: Added the TLS 1.2 protocol version.	56
	DEFINE FTPD command parameters:	
	SSLCIPHER: Added cipher suites for the TLS 1.2 protocol	59
	SSLVERSION: Added the TLS 1.2 protocol version.	60
	SSLMODE: New parameter.	60
	FTPBATCH client—OPEN command: Added the TLS 1.2 protocol version.	63
	Ch. 5, Firewall Shield	
	FIREWALL command:	79
	Added the ALLOWED and BLOCKED operands.	
	Updated the FIREWALL REPORT description.	

Table of Contents

To return from a hyperlink jump, press <Alt> <◄>

1

General Print Server

Overview

The General Print Server (GPS) optional feature enables your VTAM-based applications to print on TCP/IP-based printers without modifying the applications. The applications may run under CICS or they may be stand-alone VTAM applications.

How It Works

Each GPS daemon identifies itself to VTAM as a logical unit with the characteristics of a locally attached 3287 printer. Either GPS or an application then initiates a BIND to establish communication. Once connected, the GPS daemon receives data from the application through VTAM, just like a physical printer. Each buffer is reformatted with ASA carriage control and is written to a staging member in a library. At some predetermined point, based on amount of data or idle time, the accumulated data is passed to the standard TCP/IP LPR client. The client, in turn, establishes a TCP connection with a remote Line Printer Daemon (LPD) and transfers the file. The LPD then physically prints the file or disposes of it in some predetermined manner.

Setup

If you licensed TCP/IP FOR VSE from Connectivity Systems, Inc., or from one of CSI's distributors, GPS is included in your distribution. This section describes how to set up your system to use GPS.

Product Key

To enable the GPS feature, you need to apply a GPS product key. To check whether a valid GPS key is installed, see the section "Product Key," in chapter 2, "TLS/SSL for VSE," on page 20.

See the *TCP/IP FOR VSE Installation Guide*, chapter 3, "Installation," for information on installing product codes.

Defining to VTAM

Each GPS daemon requires a VTAM application ID. The application ID appears to the printing application as an LU. An acceptable definition of application IDs for two GPS daemons is shown in the following example:

```
TCPPRT   VBUILD TYPE=APPL
GPS1     APPL  AUTH=(ACQ),DLOGMOD=DSC2K
GPS2     APPL  AUTH=(ACQ),DLOGMOD=DSC2K
```

In this example, LOGMODE DSC2K is supplied by IBM and is suitable for BIND negotiation.

Defining to CICS

Although GPS works with any VTAM application, the majority of printing is expected to come from CICS. To use a GPS printer under CICS, you must define it to CICS as a locally attached non-SNA device. You can use the CEDA transaction to do this.

There are many combinations of parameters that you can specify for both TERMINAL and TYPETERM definitions.

Typical parameter values follow. These values are known to work.

TERMINAL Definitions. The following parameter settings are valid.

```
Terminal Characteristics:

  OBJECT CHARACTERISTICS
   CEDA  View
    TErminal       : GPS1
    Group : VSETERM1
    AUTINSTModel   : No        No | Yes | Only
    AUTINSTName    :
   TERMINAL IDENTIFIERS
    TYpeterm       : GPSPRT
    NEtname        : GPS1
    Console        : No        No | 0-99
    REMOTESystem   :
    REMOTEName     :
    Modename       :
   ASSOCIATED PRINTERS
    PRINTER        :
    PRINTERCopy    : No        No | Yes
    ALTPRINTEr     :
    ALTPRINTCopy   : No        No | Yes
    SPOOLTo        :
   PIPELINE PROPERTIES
    POol  :
    TAsklimit      : No        No | 1-32767
   OPERATOR DEFAULTS
    OPERId         :
    OPERPriority   : 000       0-255
    OPERRsl        : 1-24      0-24,...
    OPERSecurity   : 1-64      1-64,...
    Userid         :
    NAtlang        :
   TERMINAL USAGES
    TRansaction    :
    TErmpriority   : 000       0-255
    Inservice      : Yes       Yes | No
   PRINTER DATA
    SPOOLDest      :
    SPOOLPRTRsl    : Public    0-24 | Public
    SPOOLPRTTo     : 00        0-59
    PRINTEDmsg     : No        No | Yes
    PRINTImmed     : No        No | Yes
   SESSION SECURITY
    SEcurityname   :
    ATtachsec      : Local     Local | Identify | Verify
    Bindpassword   : PASSWORD NOT SPECIFIED
```

TYPETERM Definitions. The following parameter settings are valid.

```
Typeterm Characteristics:

  OBJECT CHARACTERISTICS
   CEDA  View
    TYpeterm       : GPSPRT
    Group : VSETERM1
   RESOURCE TYPE
    DEVice         : 3270P
    TERmmodel      : 2
    SESsiontype    :
    PRINTErtype    : 3287
    LDclist        :
    SHippable      : No      No | Yes
   MAPPING PROPERTIES
    PAGesize       : 024 , 080        0-999
    ALTPage        : 000 , 000        0-999
    ALTSUffix      :
    FMhparm        : No      No | Yes
    OBOperid       : No      No | Yes
   PAGING PROPERTIES
    AUTOPage       : Yes     No | Yes
   DEVICE PROPERTIES
    DEFscreen      : 024 , 080        0-999
    ALTSCreen      :    ,   0-999
    APLKybd        : No      No | Yes
    APLText        : No      No | Yes
    AUDiblealarm   : No      No | Yes
    COLor : No      No | Yes
    COPy  : No      No | Yes
    DUalcasekybd   : No      No | Yes
    EXtendedds     : Yes     No | Yes
    HIlight        : No      No | Yes
    Katakana       : No      No | Yes
    LIghtpen       : No      No | Yes
    Msrcontrol     : No      No | Yes
    OBFormat       : No      No | Yes
    PARtitions     : No      No | Yes
    PRINTAdapter   : No      No | Yes
    PROgsymbols    : No      No | Yes
    VAlidation     : No      No | Yes
    FOrmfeed       : Yes     No | Yes
    HOrizform      : No      No | Yes
    VErticalform   : No      No | Yes
    TEXTKybd       : No      No | Yes
    TEXTPrint      : No      No | Yes
    Query : No      No | Cold | All
    OUtline        : No      No | Yes
    SOsi  : No      No | Yes
    BAcktrans      : No      No | Yes
    CGcsgid        : 00000 , 00000    0-65535
   SESSION PROPERTIES
    AScii : No      No | 7 | 8
    SENdsize       : 00000   0-30720
    RECEivesize    : 00256   0-30720
    BRacket        : Yes     Yes | No
    LOGMode        :
```

```
DIAGNOSTIC DISPLAY
    ERRLastline    : No      No | Yes
    ERRIntensify   : No      No | Yes
    ERRColor       : NO      NO | Blue | Red | Pink | Green|
           Turquoise | Yellow | NEutral
    ERRHilight     : No      No | Blink | Reverse |
Underline
    AUTOConnect    : No      No | Yes | All
    ATi   : Yes     No | Yes
    TTi   : No      Yes | No
    CReatesess     : Yes     No | Yes
    RELreq         : Yes     No | Yes
    DIscreq        : Yes     Yes | No
    Nepclass       : 000     0-255
    SIgnoff        : Yes     Yes | No | Logoff
   MESSAGE RECEIVING PROPERTIES
    ROutedmsgs     : All     All | None | Specific
    LOGOnmsg       : No      No | Yes
   APPLICATION FEATURES
    BUildchain     : No      No | Yes
    USerarealen    : 100     0-255
    Ioarealen      : 00256 , 00000    0-32767
    UCtran         : No      No | Yes | Tranid
   RECOVERY
    RECOvoption    : Sysdefault       Sysdefault | None
```

Storage Library

The LPR protocol works only at a report level, which means that an entire report must be ready for processing before it is shipped to an LPD. For this reason, GPS sends the printed data to a member of the VSE library defined with the STORAGE parameter on the DEFINE GPS command. After a sufficient number of pages accumulate, an LPR operation is initiated for the data accumulated in the library member. GPS waits for acknowledgment of a successful LPR operation and then deletes the storage file. Processing resumes with the next incoming data from the VTAM connection.

The VSE library that contains these files can be shared by multiple GPS daemons because the member name is unique for each daemon. The member name is the name of the VTAM application ID (printer LU name) and the extension is PRINT.

If the LPR operation fails, the GPS daemon immediately shuts down and does not delete the storage member. When the GPS daemon restarts, its first action is to retry the failed operation. Using this method, the report is not lost.

Operation

Segmenting Reports

When you print to a real printer, report segmentation is not an issue. As each page of data is sent, it is printed. The operator who tears the paper off the printer segments the report.

When you print with GPS, segmentation is important. It would be very inefficient to package each page as an LPR segment and transmit it to the remote daemon. On the other hand, we do not want to stage data endlessly. For efficiency and expediency, GPS uses a variety of criteria to segment data.

Whenever the VTAM application releases its bind on the printer, GPS assumes that the current report is complete. The accumulated data is immediately sent to the LPD regardless of its size. The user has no control over this process.

The user may establish segmentation points using the following criteria:

- Page count. This is the preferred method because it does not introduce extraneous breaks in the middle of a report page.
- Line count. This is intended as a backup method and is supposed to keep the storage library from accidentally filling up. This value should be large enough that it does not trigger segmentation under normal conditions.
- Character count. This is intended as a backup method and is supposed to keep the storage library from accidentally filling up. This value should be large enough that it does not trigger segmentation under normal conditions.
- Idle period. When the VTAM connection is idle (that is, presents no new data) during the specified interval, GPS segments the current report. This feature is useful where the printer is not busy 100 percent of the time. When a report is finished, it is sent to the LPD without waiting for another report to flush it through. The value you specify should be large enough that occasional delays in CICS (or other VTAM applications) do not cause a false end-of-report condition. A good value to start with is 10 seconds.

Controlling the Remote Printer

Desktop printers generally have more features than the VTAM printers they replace. In most cases, you need to set operation modes before you print your reports. For example, you might want to specify landscape mode or a mono-spaced font.

Fortunately, GPS has a facility that permits you to include control data that is automatically merged with the print stream. Control data can precede the report, precede each page of the report, and follow the last page of the report. To accomplish this, you can identify an INSERTS phase on the DEFINE GPSD command for each GPS daemon.

The INSERTS phase is syntactically identical to the INSERTS phase documented in the LPR chapter of the *TCP/IP FOR VSE User Guide*, but the data is processed by GPS and not LPR.

Directing Reports

In the VTAM/SNA world, a printer is a physical unit. It has one name and is owned by one application at a time. When two CICS partitions share a printer, for example, one CICS acquires the printer, prints its report, and then releases it. The other CICS is then free to acquire the printer and begin its print operation.

Because LPR/LPD is a spooling protocol, this restriction does not apply. If you have two CICS partitions and one remote printer, simply define two GPS daemons (with different APPL IDs, of course) and assign one to each CICS. You can assign both daemons to one remote LPD. As reports are produced, each GPS daemon collects its data in its staging file and spools the data to the remote LPD as required. The LPD that owns the printer respools the reports and prints them on the same physical printer. Most LPDs support printers with multiple names, so you can use this mechanism to assign priorities and other properties to the reports.

HP JetDirect

Many installations use print-serving devices. Each device can be a card installed in a laser printer or a small gray box into which you can plug any serial printer. In general, these cards turn each printer into a miniature special-purpose TCP/IP running the LPD application. If you use these devices, you need to be aware of some limitations.

The JetDirect card or box attaches directly to your Ethernet. The first limitation is that the JetDirect device has no disk. This is relevant because an LPD is required to receive and spool an entire report before processing it. The LPR/LPD protocol includes a data file and a control file. The control file specifies how the data is to be processed, including such information as the number of copies, which named printer to use, whether to print and delete or to print and save, and so on. Some of these LPD implementations completely print the report before they receive (and ignore) the control file, so none of these features work. Another problem is that in the LPR/LPD protocol, error recovery consists of restarting the transmission. This works perfectly with a spooled LPD, but it results in extraneous pages on these types of devices.

The second limitation is that if the printer is busy (for example, printing a document from a Windows client using IPX), it simply ignores any GPS attempt to connect to it. Unfortunately, this means that it appears to be down. GPS is forced to signal the error to the operator and then shut down. This limitation is annoying, but can be minimized if other use of the printer or the attached card is limited. You can also specify that GPS retry the failing LPR after an interval.

Logging

You probably are not concerned with every event that occurs in the GPS daemon. When things are not working and you need information, you can start a GPS daemon with the parameter LOG=YES. When you do that, the GPS daemon creates an additional member in the storage library. The member name is the same as the daemon's terminal name (as identified by the TERMNAME= parameter). The extension is LOG.

The log file contains printable EBCDIC and has RECFM=F and LRECL=80. Different types of information are recorded, including dumps of VTAM control blocks.

You cannot access the log file until you stop the GPS daemon. When you restart the daemon, it overwrites any existing log file.

Debugging

When GPS is active, it receives a 3270 job stream from your application, converts it to lines of data with ASA carriage control, and ships it to the LPR client. The LPR client converts it to an ASCII data stream with embedded line and form control. Needless to say, this is a complex task, and not all reports appear as intended (although most do).

Normally, the GPS daemon uses one member to stage the data (such as GPS1.PRINT, where GPS1 is the value assigned to the TERMNAME= parameter). After LPR processes the report, GPS reuses the member.

When you specify DEBUG=YES, GPS uses a different approach. Instead of one member, it uses two. The first member contains the standard data that is transmitted to the LPR client and the second contains a copy of the data exactly as it is received from VTAM. If the GPS terminal name is GPS1, the two members are named *GPS1.PRTnnnnn* and *GPS1.RAWnnnnn.* Variable *nnnnn* is a five-digit number that begins with 00001 when the daemon starts and increments by one for each report segment.

Although GPS always starts with number 00001 and overwrites any existing members, it does not delete members. You must perform the delete function manually. The amount of data logged is voluminous and your staging library can fill rapidly. You should use DEBUG=YES only in a controlled environment and only to solve formatting problems.

The RAW files contain raw 3270 data streams as sent from VTAM. They are stream mode members with RECFM of *S*. If you FTP a RAW file to Connectivity Systems for problem analysis, use binary mode, RECFM=S. In basic mode, the PRT files are printable EBCDIC and have a RECFM of SV (a special format for variable-length record library members). The TCP/IP FOR VSE FTP daemon supports this record format, and you can FTP the data to your PC for viewing and analysis. Specify "SITE RECFM SV" and ASCII for the transfer.

The PRT files contain the ASCII data stream that is sent directly to the LP daemon without modification. Its RECFM is S. If you download this file to your PC in binary mode, you can read it. This file also contains INSERTS data, if specified.

The RAW files allow CSI technical support to replicate and analyze problems.

DEFINE GPSD Command

The DEFINE GPSD command initiates a single instance of a GPS daemon (server). You must define one daemon for each VTAM logical unit that you want to emulate. There is no MODIFY GPSD command. To change a specification, you must delete and redefine the daemon. The syntax is as follows.

```
DEFINE GPSD,ID=name,STORAGE='pubname',IPADDR=dest,TERMNAME=lu, -
PRINTER=pname[,LOG={YES|NO}][,TRANSLATE=xlate][,DEBUG={YES|NO}]
   [,INSESS={YES|NO}][,USER=user][,PASSWORD=pswd][,TARGET=appl]
   [,LOGMODE=mode][,MAXPAGES=pp][,MAXLINES=ll][,MAXCHARS=cc]
   [,MAXIDLE=tt][,NETWORK_RETRY_COUNT={nn|3}]
   [,NETWORK_RETRY_TIME={rr|18000}][,VTAM_RETRY_COUNT={nn|10}]
   [,VTAM_RETRY_TIME={nn|18000}][,INSERTS=ins][,LINELEN=len]
   [,NOEJECT={YES|NO}][,QUEUING={MEMORY|DISK}]
   [,ALTLEN=length][,OUTPUT={LPR|DIRECT}][,PORT=num]
   [,EMULATE={3287|TRANSPARENT}]
```

The parameters are as follows:

Parameter	Description
ID=	Identifies the individual daemon. The value is a 1- to 16-character alphanumeric name. The first character must be alphabetic. This is a required parameter.
STORAGE='*pubname*'	Specifies the library to be used for staging LPR data and for the optional logging file. Variable *pubname* is enclosed in single quotes and is the public name of a library and sublibrary to be used by the GPS daemon. This is a required parameter.
IPADDR=	Specifies the IP address of the remote host that owns the LPD. The value is a numeric or symbolic IP address. This is a required parameter.
TERMNAME=	Specifies the VTAM LU name that identifies GPS to VTAM applications. The value you specify must be defined to VTAM as an application ID. This is a required parameter.

Parameter	Description
PRINTER=	Specifies the name of a print queue. This name is sent to the LPD on the remote host to identify the target printer. You must know this name and specify it here. Otherwise, the LPD rejects GPS's attempts to send data. This is a required parameter.
LOG=[YES | NO]	Specify YES to direct the GPS daemon to create a log file. The log file name is the value specified for TERMNAME with an extension of LOG. This file is overwritten each time the daemon restarts. The file is a simple text file, and you can use standard VSE facilities to view or print it. The default is NO.
TRANSLATE=	Specifies a translate table name that the LPR client is to use when it converts the printed data to ASCII. **Note**: If this parameter is set on the DEFINE FILE command when you define the STORAGE file, that setting takes precedence over the value set on DEFINE GPSD. If TRANSLATE= is not set, the system default table is used.
DEBUG=[YES | NO]	Specify YES to direct the GPS daemon to run in debug mode. In this mode, raw VTAM transmissions and all staged data are saved in the staging file. You can review the data to ensure a correct conversion. When you are finished, you must manually delete these files. The debugging mode may require considerable library space, depending on the amount of data being printed. The default is NO.
INSESS=[YES | NO]	Specify YES to direct the GPS daemon to attempt to bind with the application specified in the TARGET= parameter immediately at startup. Specify NO to direct the GPS daemon to wait for the application to initiate the bind request. If the application releases the bind, the daemon makes no additional bind attempts even if you specify YES. The default is NO.
USER=	Sets the user ID to be used when initiating an LPR request. This is required if you are running TCP/IP with security on and your staging library is security protected. The default is AUTOLPR.

Parameter	Description
PASSWORD=	Specifies the password to be used when initiating an LPR request. This may be important if your staging library is security protected. There is no default. The QUERY GPSD command does not display the password.
TARGET=	Identifies the VTAM application that you want GPS to bind with immediately at startup. The value is the application ID. This parameter is effective only if you also specify INSESS=YES. If you specify INSESS=NO or if you omit this parameter, GPS does not attempt a bind and instead waits for an application to initiate the bind.
LOGMODE=	Specifies the VTAM LOGMODE name that is to be used to negotiate a bind with the VTAM application. The default (and recommended) value is DSC2K. This is an IBM-supplied, non-SNA LOGMODE for printers. This parameter is effective only if you also specify INSESS=YES and TARGET=appl.
MAXPAGES=	Specifies how many pages must accumulate before an LPR operation begins. Because LPR is a spooled protocol and GPS emulates a serial device, GPS can use the MAXPAGES parameter to determine when to segment the report and transmit it using LPR. One method is to count pages and begin the LPR operation when the page count is reached. The advantage of triggering by page count is that no extraneous page breaks are introduced. The default is 100 pages.
MAXLINES=	Specifies how many lines must accumulate before an LPR operation begins. Because LPR is a spooled protocol and GPS emulates a serial device, GPS can use the MAXLINES parameter to determine when to segment the report and transmit it using LPR. The method is to count lines and begin the LPR operation when the line count is reached. The disadvantage of triggering by line count is that extraneous page breaks are introduced. The default is 10000 (10K lines).

Parameter	Description
MAXCHARS=	Specifies how many characters must accumulate before an LPR operation begins. Because LPR is a spooled protocol and GPS emulates a serial device, GPS can use the MAXCHARS parameter to determine when to segment the report and transmit it using LPR. The method is to count characters and begin the LPR operation when the character count is reached. The disadvantage of triggering by character count is that extraneous page breaks are introduced. The default is 1000000 (1M characters).
MAXIDLE=	Specifies how much idle time needs to pass before an LPR operation begins. The value is the number of 1/300th-second intervals. Because LPR is a spooled protocol and we are emulating a serial device, GPS can use the MAXIDLE parameter to determine when to segment the report and transmit it using LPR. We also need a procedure for transmitting accumulated leftover data. One method is to begin the LPR operation when no data is received over the VTAM connection for *tt*/300 seconds. The value must be set high enough so that normal processing delays do not cause premature transmission of the report and extraneous page breaks. The default is 3000 (10 seconds).
NETWORK_RETRY_ COUNT=	Specifies how many times GPS should retry when it cannot connect with the network at startup. The value is the number of times GPS attempts to retry before it shuts down. The default is 3. This value also controls the retries of failing LPR operations.
NETWORK_RETRY_ TIME=	Specifies the interval at which GPS should retry when it cannot connect with the network at startup. The value is the number of 300th-second intervals between retries. The default is 18000 (1 minute).
VTAM_RETRY_ COUNT=	Specifies how many times GPS should retry when it cannot connect with VTAM at startup. The value is the number of times GPS attempts to retry before it shuts down. The default is 10. This value also controls the retries of failing LPR operations.

Parameter	Description
VTAM_RETRY_ TIME=	Specifies the interval at which GPS should retry when it cannot connect with VTAM at startup. The value is the number of 300th-second intervals between retries. The default is 18000 (1 minute).
INSERTS=	Specifies data to be included with each report. The value is the name of a phase containing inserts data. The INSERTS phase contains data to be transmitted before the report, before each page, and after the report. For details on using an INSERTS phase, see the *TCP/IP FOR VSE User Guide*.
LINELEN=	Specifies the maximum line length for your application. The value is the maximum line length. The 3287 emulation provides for a maximum line length of 132. When the line is filled, the printer forces a newline operation. If your application needs to print longer lines, you can increase the printer's maximum line length to as many as 255 characters.
NOEJECT= [YES | NO]	Specify YES to suppress the initial form feed character at the beginning of a listing. Many print files begin with a page-eject character. TCP/IP FOR VSE normally translates this character into a form feed; however, this creates a blank page on some printers. Specify NO (the default) if you do not want to suppress the initial form-feed character.
QUEUING= [MEMORY | DISK]	Specify MEMORY to direct GPS to queue printouts in 31-bit memory instead of using a VSE library member. Specify DISK (or allow it to default) to direct GPS to queue printouts to the VSE library defined with the STORAGE= parameter. Using memory-based queuing can reduce the 24-bit storage requirements for each daemon by approximately 100K.
ALTLEN=	Specifies the line length to be used if an application uses the ERASE WRITE ALTERNATE command. If an application issues this command, the 3270 printer is set to its alternate characteristics. The default is 80. See the LINELEN= parameter description for more information about GPS and line lengths.

Parameter	Description
OUTPUT= [LPR | DIRECT]	Specify LPR to transmit GPS output as LPR/LPD data to the Line Printer Daemon at the other end. Specify DIRECT to transmit GPS output as a print data stream with no LPR/LPD negotiations or framing. DIRECT is suitable for the direct socket connection of HP or a custom application that needs to receive the data. The default is LPR. For more information about these options, see the discussion following this table.
PORT=	Specifies the port number for the GPS daemon. This applies only when you specify OUTPUT=DIRECT.
EMULATE= [3287 | TRANSPARENT]	Specify 3287 to indicate that processing is to emulate a 3287 printer. In this case, printable data is converted to ASCII, and carriage control characters are converted to those required by common ASCII-mode printers (for example, CR, LF, and FF characters are used). This is the default. Specify TRANSPARENT to indicate that the data received over the VTAM connection is transmitted directly without translation or interpretation. The 3270 commands (for example, WRITE and ERASE WRITE) and WCC bytes are removed, leaving only 3270 orders and data bytes. INSERTS data is added, when specified, only for start of report and end of report. Because the data itself is not processed, top-of-form INSERTS data cannot be supplied. Also, for report segmentation, each VTAM transmission is considered to be one page.

Using the OUTPUT Parameter

The following applies when you specify OUTPUT=DIRECT:

- The connection with the remote printer/host opens as soon as the first buffer is received from VTAM. When you specify OUTPUT=LPR, the remote connection opens when the LPR transmission is ready to begin.
- The connection is held open until the VTAM interface stops sending data for an interval that exceeds the value specified for MAXIDLE.

- Print data is NOT buffered. The VTAM interface transmits a series of 3270 orders and data. This information is used to construct an image buffer (that is, a 3270 screen/printer buffer). Following a physical VTAM transmission that has the start printer WCC bit set, the image buffer is interpreted and transmitted across the TCP/IP connection.

  This contrasts with OUTPUT=LPR. Instead of transmitting the data at this point, LPR processing dictates that the data (which is buffered) be added to a queue (disk or memory) for later batch transmission.

- The INSERTS phase, if any, is added to the data stream at the appropriate place.

- Although disk/memory buffering is not used, its specification is still required. The loss of the TCP/IP connection during processing can result in the loss of up to one screen of data. This contrasts with LPR processing, which queues undelivered data for retransmission at a later time.

Example A DEFINE GPSD command is shown in the following example:

```
msg f8
AR 015 1I40I  READY
F8 043 IPN300I Enter TCP/IP Command
F8-043
43 define gpsd,id=gps1,storage='gps.save', -
43 ipaddr=rmt1,printer=prt1,termname=gps1
F8 043 GPS900I GPS1 GPS Daemon Starting
F8 043 GPS927I GPS1 GPS Version dated 9/26/15
F8 043 FPS917I GPS1 Waiting for BIND
F8 043 IPN300I Enter TCP/IP Command
F8-048
```

DELETE GPSD Command

The DELETE GPSD command terminates an active GPS daemon. When you issue this command, processing stops immediately. The syntax is as follows:

```
DELETE GPSD,ID=name
```

The parameter is as follows:

Parameter	Description
name	Identifies the daemon to be terminated. The value is a 1- to 16-character alphanumeric name.

QUERY GPSD Command

The QUERY GPSD command displays the status of a specific GPS daemon or all GPS daemons. The syntax is as follows:

```
QUERY GPSD,ID=name
QUERY GPSDs
```

The parameter is as follows:

Parameter	Description
name	Identifies the daemon to be displayed. The value is a 1- to 16-character alphanumeric name.

Problem Solving

Overview

This section describes solutions for common GPS problems in these areas:

- VTAM Problems
- LPR Failure
- Delayed Reports
- GPS Termination
- Formatting Problems

VTAM Problems

Is the VTAM APPL properly defined, and did you vary it active? If REQSESS is failing, then CICS (or another application) is refusing a bind. Make sure that CICS has the appropriate printer's LU marked in service. The CICS log may also help determine why a bind is rejected. You can use IUI menu option 4.2 to access the CICS log.

LPR Failure

Check the IP address and LPD printer name. They must correspond with a reachable LPD and a valid printer queue.

Delayed Reports

GPS can only segment and transmit reports using criteria that you specify. Delays are generally caused when you set too large a value for the MAXIDLE= parameter. MAXIDLE is the maximum idle time that must pass between CICS and GPS before GPS forces an LPR of any leftover data. We do not advise setting MAXIDLE to less than 600 (2 seconds).

GPS Termination

A GPS daemon terminates when the following events occur:

- Someone issues a DELETE GPSD command.
- An error occurs that is not potentially recoverable. In this case, correct the error and restart the daemon. GPS recovers and retransmits the stranded report data.
- An error occurs that is potentially recoverable, but the specified number of retries is exhausted. If this occurs, consider increasing the number of retry operations that may be attempted. To do this, use the NETWORK_RETRY_COUNT= or the VTAM_RETRY_COUNT= parameter.

Use the DIAGNOSE GPS command to obtain more diagnostics.

Formatting Problems

Data is too far left/right/up/down. You have not set proper margins on your printer. You can set the margins manually, or you can provide an LPR INSERTS phase to set them electronically. See the *TCP/IP FOR VSE User Guide* for information about using the LPR INSERTS phase.

Characters do not line up properly. This usually means that you are printing with a proportional font. Manually set the printer to a mono-spaced font (such as Courier New), or include an LPR INSERTS phase to set the font electronically.

Report pages split across physical pages. For LPR to paginate properly, LPR transmittal must occur only on logical page boundaries. This means that only MAXPAGES= and correctly interpreted MAXIDLE= values may cause LPR transmittal. Physical release of the LU by CICS is also acceptable. If transmittal is forced by line count (MAXLINES=) or character count (MAXCHARS=), unintended page skips are introduced. Either increase MAXLINES= and/or MAXCHARS=, decrease MAXPAGES=, or increase MAXIDLE=, as appropriate.

Extra random characters appear in the reports. This is usually caused when an application sends control sequences that are not recognized by GPS and are simply passed through to LPR. Ensure that CICS and applications understand that this is a 3287 device.

Extra blank lines appear in the report. GPS emulates a 3287 printer, so a line containing 132 characters followed by an NL actually prints as a data line followed by a blank line. This is actual 3287 behavior. If CICS or your application does not conform to 3287 standards, contact CSI International to see if other emulations are possible.

Bar codes, graphics, and printer control strings do not work. Remember that GPS emulates an IBM 3287 printer and not a third-party device that looks like a 3287 printer with extensions. A necessary part of LPR is EBCDIC-to-ASCII translation. Binary/image data is not preserved. You may be able to circumvent this by translating the image data from ASCII to EBCDIC before transmission to GPS. GPS's translation then simply restores the data.

Another consideration is that the image data, as an emulated printer, occupies physical locations on the page. Pagination leaves space for all data regardless of how the remote printer handles it. If image data exceeds the 3287 line length (maximum 132), CR/LFs are inserted. A suggested workaround is to break image data into segments of less than 132 characters, place the segments on lines that don't contain printable data, and terminate each segment with a carriage return (X'0D'). This forces a logical line overstrike and does not affect the vertical spacing of the final report.

Other problems. It is impossible to anticipate all possible uses of GPS. In addition, there are inherent problems in emulating one form of hardware on another unit that is basically incompatible. For these reasons, we have included a number of diagnostic aids in GPS.

If you have problems with report formats that cannot be adjusted using the parameters provided, see the next section, "Obtaining Support."

Obtaining Support

If you have an improperly formatted report and have determined that it is an issue with the product, please do the following before you call for support:

1. Prepare to generate a small portion of the report, such as one or two pages.
2. Print a correct copy of the report. We may ask you to transmit it to us.
3. Start your GPS daemon with the DEBUG=YES and LOG=YES options set.
4. Print the bad report.
5. Stop the daemon.
6. Using FTP, transfer the GPS members from the staging library to a PC. You need the following two files:

File Name	Description
luname.LOG	This is the log file. It should be transmitted as an ASCII file using RECFM=F.
luname.RAW00001	This is the most important file. It contains the data as it is sent from VTAM. We use this data to simulate a CICS session and recreate the formatting. This file must be sent by FTP in binary mode using RECFM=S. If you understand GUI FTP clients and UNIX mode, you can proceed. If you are using a point-and-click FTP client, we suggest that you first rename the member to *luname*.BIN. This filename extension prevents UNIX mode from forcing an ASCII transfer.

7. Zip the two files and contact CSI Technical Support. They will give you instructions for transmitting the diagnostic information for analysis.

2

TLS/SSL for VSE

Overview

TLS/SSL for VSE is an optional feature that is included with TCP/IP FOR VSE. It provides security for TCP/IP applications by implementing the Transport Layer Security (TLS)/Secure Sockets Layer (SSL) protocol, as defined by the Internet Engineering Task Force (IETF). The IETF has enhanced and replaced the SSL 3.0 specification with the TLS protocol specification.

When a client wants to establish a secure TLS/SSL connection, it proposes a set of cryptographic algorithms it can support. The TLS-enabled server application examines this cipher suite list and tells the client which one will be used. The algorithms manage authentication, data encryption, and message integrity. These concepts are explained in the section "Protocol Introduction" on page 22.

Product Key

You must have a valid SSL product key to use TLS/SSL for VSE, the SHA-1 Phase Verification procedure, or any of the cryptographic functions.

To check whether a key is installed, issue a "Q PRODKEYS" command in the TCP/IP FOR VSE partition. This command returns information on installed keys, such as the following example:

```
IPN886I SSL.........           (CSI) Expires on 2018/12/31
```

If the display indicates that the key is missing or expired, contact CSI International at sales@csi-international.com to request a key. There is no cost to obtain an SSL key.

Important:

Export and re-export controls on commercial encryption products are administered by the Bureau of Industry and Security (BIS) in the U.S. Department of Commerce. This encryption feature is not available in countries restricted by U.S. Department of Commerce export regulations.

More information is available at this site:

www.bis.doc.gov/index.php/policy-guidance/country-guidance

Protocol Introduction

Authentication

The TLS/SSL protocol uses certificates to authenticate servers and clients. It is easiest to think of certificates as a driver's license or passport. When you are pulled over for speeding, or if you enter a different country, the first item you are asked for is either a driver's license or passport that identifies you. You can be in serious trouble if you cannot produce the required documents.

The authentication of a certificate is exactly like that. It contains a signature, expiration date, and other important information. The signature on the certificate is provided by a certificate authority, and authorizing agencies provide services for creating and revoking certificates they issue. The system of managing these certificates is known as *Public Key Infrastructure* (PKI).

The server always sends back a certificate to the client that is digitally signed by a certificate authority. This certificate contains the public part of a RSA key that the client then uses to encrypt a random value. The server also has the private part of the RSA key and uses it to decrypt the encrypted random value from the client. This is referred to as public/private key encryption. It is important to know that the private part of the RSA key is never revealed nor sent out over the network. It is kept completely secret in the server system.

Optionally, the server can also request a certificate from the client application, and the server can then verify the client's identify. This would be the same as asking the officer to see his driver's license after he has asked you for yours. It would allow you to guarantee that he really is a police officer, although this is not normally done. In the TLS protocol, this transaction is referred to as client authentication. Just remember that server authentication is always required, but client authentication is optional (and not normally used).

A site that wants to use a TLS/SSL server application on VSE must first obtain a certificate from an authority such as Network Solutions, GoDaddy, VeriSign, Thawte, or OpenSSL. The certificate contains the public part of the server application's RSA key. The client uses the public key to securely encrypt a random value. This random value is used to create keys for encrypting/decrypting and for guaranteeing the integrity of the data flowing over the connection. No new connections ever reuse the same key values.

Data Encryption

Once a server and client are authenticated, the random value is used to create keys for encrypting/decrypting the data that is sent and received over the secure connection. The keys, which are randomly generated and unique to the one session, are used for the life of the connection and then destroyed when the TLS connection is terminated. TLS/SSL for VSE can use the Single-DES, Triple-DES, AES-128, and AES-256 algorithms for data encryption and decryption.

Message Integrity

We can authenticate servers and clients and keep data confidential by encrypting it, but what about guaranteeing the integrity of the data as it traverses the network? Isn't it possible to just randomly change bits of data to cause data corruption? Without TLS/SSL, the answer is absolutely yes!

But, what about the TCP checksum—doesn't that keep my data safe and secure?

The answer is NO. A hacker could change the data, recreate the checksum, and the TCP protocol would pass it onto the application as good data.

With TLS/SSL, a secure message authentication code (MAC) for the data is created before data encryption and stored at the end of each data block sent out on the network. The MAC is created using a secure hash, such as SHA-1, combined with some of the secret key data. The result is that any changes caused by a hacker or a hardware failure are detected.

Note:

See "References" on page 52 for more information on cryptography and computer security.

Feature Components

TLS/SSL for VSE implements the cryptography algorithms required for the key exchange, data encryption, and message authentication. It also provides utilities to install certificates, a daemon to transparently enable existing applications to TLS/SSL, and APIs to natively implement TLS/SSL or cryptography into your applications. TLS/SSL for VSE relies on a number of integrated components, including:

- PKI for identification
- RSA for key exchange
- AES, Single-DES, or Triple-DES for data encryption
- MD5, SHA-1, or SHA-2 for message hashing
- HMAC for message authentication.

Industry Standards

TLS/SSL for VSE is the implementation of numerous industry-standard algorithms and is based on definitions published by IETF's Transport Layer Security (TLS) working group. More information on TLS standards is at http://datatracker.ietf.org/wg/tls/charter. The associated Request for Change (RFC) documents are available at http://ietf.org. See also "Published Standards" on page 51.

By using an open, industry-standard protocol and associated algorithms, you are assured of compatibility with a wide variety of platform applications that are TLS/SSL enabled.

TLS/SSL on TCP/IP FOR VSE

TLS/SSL for VSE is integrated into the TELNETD, HTTPD, FTPD, and Entrée servers to provide security for these applications from remote TLS/SSL clients. In addition, the FTP client can be TLS-enabled on VSE to securely communicate with a TLS/SSL-enabled remote FTP server. TLS/SSL for VSE also provides security for other TCP applications that use the TLS/SSL for VSE APIs.

Setting Up Your Application

Overview

This section describes how to set up your application to use TLS/SSL for VSE. A PC must be connected to the VSE mainframe that is the target of the TLS/SSL setup. You must use the free IBM Keyman/VSE utility to perform some of the tasks in the following procedure.

For details on setting up and using the IBM Keyman/VSE utility, see the IBM z/VSE website. Since the links below are owned and maintained by IBM, they may change. Please contact IBM z/VSE technical support if the following links are not valid.

http://www-03.ibm.com/systems/z/os/zvse/downloads/

http://www-03.ibm.com/systems/z/os/zvse/documentation/security.html#howto

ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/How_to_setup_and_use_KeymanVSE.pdf

Sample Jobs

TLS/SSL for VSE is integrated into and distributed with the TCP/IP FOR VSE product. The table below contains brief descriptions of the sample jobs in this chapter that must be created on your system. Simply copy them from the text and paste them into your editor (ICCF, BIM-EDIT, VM-XEDIT, or similar tool).

Sample Job	Description
CIALSRVR.JCL	Runs the RSA key-generation server
CIALPRVK.JCL	Catalogs an RSA private key file
CIALCERT.JCL	Catalogs a server or client certificate file
CIALROOT.JCL	Catalogs a certificate authority root certificate file
CIALSIGV.JCL	Verifies the certificate signature
CIALGPRV.JCL	Backs up an RSA private key file

Setup Summary

The setup procedure consists of the following tasks. Each task is explained in detail below.

- Create a key sublibrary
- Create sample files from the text examples below.
- Create custom files.

Step 1: Create a Key Sublibrary

Designate a separate VSE sublibrary for storing the private key and certificate files. The library.sublib should be unique and NOT defined to the TCP/IP FOR VSE file system to protect these sensitive key files.

Defining a separate sublib allows each TLS/SSL application running on VSE to use a unique private key, server certificate, and root certificate. The following job can be used to create this sublibrary. In this example, the sublibrary PRD2.SSLKEYS is created.

```
// JOB LIBRDEFS
// EXEC LIBR,SIZE=256K
DEFINE SUBL=PRD2.SSLKEYS REUSE=IMMED
/*
/&
```

Step 2: Create Sample Files

This step consists of procedures 2-1 through 2-4. Use these procedures to create RSA key and certificate files for testing.

Note 1: Replace lib.sublib in all the sample jobs with the library and sublibrary used to install TCP/IP FOR VSE.

Note 2: The jobs in these procedures contain a PARM='*library.sublib.member*' definition. Specify the VSE sublibrary you created in Step 1 above. It will contain the RSA private key. Choose the member name carefully ("SAMPLE03" in the examples). It must be unique, and you must use the same name in each job. Each job creates a library member using this member name. For example, if the member name is SAMPLE03, the job CIALPRVK creates "SAMPLE03.PRVK." These members are stored in the sublibrary.

Also, you must use these library, sublib, and member names when you configure a TLS application. For example, you will need them when you

- Define a TLS daemon. See the section "Defining the TLS/SSL Daemon" on page 36.
- Define a SecureFTP server/client. See chapter 3, "SecureFTP for VSE" on page 53.

The procedures below refer to jobs in the following files.

Proc.	File Name	Job Function
2-1	CIALPRVK.JCL	Catalogs a sample RSA private key file
2-2	CIALCERT.JCL	Catalogs a sample server certificate file
2-3	CIALROOT.JCL	Catalogs a sample root certificate file
2-4	CIALSIGV.JCL	Verifies that the sample RSA private key and certificates are synchronized

Procedure 2-1: Catalog the Sample RSA Private Key File

Modify and run the CIALPRVK sample job. This job contains a sample RSA private key in the job stream. You can use this job, suitably modified, for testing. The key data is not humanly readable. The binary data is represented with base64 printable-character encoding. Note that the key data you actually use must be kept private. IBM maintains other sample jobs in ICCF MEMBER SKSSLKEY SAMPLE.

```
// JOB CIALPRVK
// OPTION SYSPARM='00'          SysId of main TCP/IP partition
// LIBDEF PHASE,SEARCH= lib.sublib
// EXEC CIALPRVK,SIZE=CIALPRVK,PARM='PRD2.SSLKEYS.SAMPLE03'
BwIAAAAApABSU0EyAAAEAAABAAHNVqgad4F3Nt0tp923uGuRavlUaTYQ2ZTd/bh
ES5R0FdEhpOV3OjqxAZ54Zbi17w88ngKxGxWrJ2N7UcARR8NP3fll6kDuzJbPLr
ulwbLR9ah/iem6XWr6dw9P1m6nU/MMlbLGcfv1GjLy2Fb5evRy81iLnyA+F3ANF
oXMPrADqfOS3eGEApwVhZ/yGKebMVANr2yHrZ33Z+NnUBDBn0daFahPn8C6Lrc3
hUqVRC1CM4tbQCMtz9iVHkZF6APeMqfX0G3BqGIZLDS6vt95P2c6EFVRkf1nv33
e4dDxzipO5Cqcx+7H5XFwoW6q8KTXQwW8sg1PiZ3fKH/vG1X1T3Eve574ufJ0GJ
/bt/f5h2d+ubCYbi7WcSwWMBZJTpfWrTINd+UhoYFdIGQvZFEpzwagfoQLo08nB
Dd6rBPi08O8VXfSpo3GUqkNTT1+Ko+kL0kitmwytVJ8FIPCNwN6Zy0kK7CepSnh
b2mMh+hq8er+sFnDNfa6PqgfUWDNeZ1Y/UtH9PJC2G7gJhOs/9O9g7x0NlTNcAR
DW8TMfLGvhBL8zJoNT77G6S/6eN+pZaqfR6PeW4Oi9za8/B8P7Te4S4p+ZyZzB+
iktthuv3KmZJZWIxTT711y9X/30pSULmqxRqd3CSmmxAASX7zqeJzQ70QKKW9O9
t0eQ35j8R0eGRAtwlh/4u92F/CkovPp8VLti9rEE++ysyJhqk0tiXSRbl7QJhsP
/kRSR7H/F5WlZFm6ggPUpoT0mkiYgC7FiFrOpYk=
/*
/&
```

Procedure 2-2: Catalog the Sample Server Certificate File

Modify and run the CIALCERT sample job below. This job contains a sample X.509v3 server certificate. This is an alternative to creating a certificate using the Keyman/VSE utility or obtaining a certificate from a provider such as VeriSign or Thawte. You can use this certificate for initial testing. The binary data is represented with base64 printable-character encoding.

```
// JOB CIALCERT
// OPTION SYSPARM='00'          SysId of main TCP/IP partition
// LIBDEF PHASE,SEARCH=lib.sublib
// EXEC CIALCERT,SIZE=CIALCERT,PARM='PRD2.SSLKEYS.SAMPLE03'
-----BEGIN CERTIFICATE-----
MIICozCCAgygAwIBAgIDQHoaMA0GCSqGSIb3DQEBBAUAMIGHMQswCQYDVQQGEwJ
aQTEiMCAGA1UECBMZRk9SIFRFU1RJTkcgUFVSUE9TRVMgT05MWTEdMBsGA1UECh
MUVGhhd3RlIENlcnRpZmljYXRpb24xFzAVBgNVBAsTDlRFU1QgVEVTVCBURVNUM
RwwGgYDVQQDExNUaGF3dGUgVGVzdCBDQSBSb290MB4XDTAwMDkyNTAwNTcwNFoX
DTAxMDkyNTAwNTcwNFowfjELMAkGA1UEBhMCVVMxDTALBgNVBAgTBE9oaW8xETA
PBgNVBAcTCENvbHVtYnVzMR0wGwYDVQQKExRDb25uZWN0aXZpdHkgU3lzdGVtcz
EUMBIGA1UECxMLRGV2ZWxvcG1lbnQxGDAWBgNVBAMTD3d3dy5zc2w0dnNlLmNvb
TCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAzVaoGneBdzbdLafdt7hrkWr5
VGk2ENmU3f24REuUdBXRIaTldzo6sQGeeGW4te8PPJ4CsRsVqydje1HAEUfDT93
5ZepA7syWzy67pcGy0fWof4npul1q+ncPT9Zup1PzDJWyxnH79Roy8thW+Xr0cv
NYi58gPhdwDRaFzD6wA6kCAwEAAaMlMCMwEwYDVR0lBAwwCgYIKwYBBQUHAwEwD
AYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQQFAAOBgQBeKDqSmeCYyL/T2pMSM03X
22NwGLyh3tbXZbyCVkfPEiTjygf5vpg1Bx8yEOxMP4nGlcZaSMlIEbue6FZAARI
cBtI6X1mtws9THbXo4xZpgectvhdA9wCWmszDLZcjai417K6oZYwYjsIPMwmi/7
Vl/0RUuDj6YnHQli6x1BYXAw==
-----END CERTIFICATE-----
/*
/&
```

Procedure 2-3: Catalog the Sample Root Certificate File

Modify and run the CIALROOT job stream. This job contains a sample X.509v3 certificate authority root certificate you can use for initial testing. Again, base64-encoded printable characters represent the binary data.

```
// JOB CIALROOT
// OPTION SYSPARM='00'          SysId of main TCP/IP partition
// LIBDEF PHASE,SEARCH=lib.sublib
// EXEC CIALROOT,SIZE=CIALROOT,PARM='PRD2.SSLKEYS.SAMPLE03'
-----BEGIN CERTIFICATE-----
MIICmTCCAgKgAwIBAgIBADANBgkqhkiG9w0BAQQFADCBhzELMAkGA1UEBhMCWkE
xIjAgBgNVBAgTGUZPUiBURVNUSU5HIFBVUlBPU0VTIE9OTFkxHTAbBgNVBAoTFF
RoYXd0ZSBDZXJ0aWZpY2F0aW9uMRcwFQYDVQQLEw5URVNUIFRFU1QgVEVTVDEcM
BoGA1UEAxMTVGhhd3RlIFRlc3QgQ0EgUm9vdDAeFw05NjA4MDEwMDAwMDBaFw0y
MDEyMzEyMTU5NTlaMIGHMQswCQYDVQQGEwJaQTEiMCAGA1UECBMZRk9SIFRFU1R
JTkcgUFVSUE9TRVMgT05MWTEdMBsGA1UEChMUVGhhd3RlIENlcnRpZmljYXRpb2
4xFzAVBgNVBAsTDlRFU1QgVEVTVCBURVNUMRwwGgYDVQQDExNUaGF3dGUgVGVzd
CBDQSBSb290MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC1fZBvjrOsfwzo
ZvrSlEH81TFhoRPebBZhLZDDE19mYuJ+ougb86EXieZ487dSxXKruBFJPSYttHo
Cin5qkc5kBSz+/tZ4knXyRFBO3CmONEKCPfdu9D06y4yXmjHApfgGJfpA/kS+Qb
biilNz7q2HLArK3umk74zHKqUyThnkjwIDAQABoxMwETAPBgNVHRMBAf8EBTADA
QH/MA0GCSqGSIb3DQEBBAUAA4GBAIKM4+wZA/TvLItldL/hGf7exH8/ywvMupg+
yAVM4h8uf+d8phgBi7coVx71/lCBOlFmx66NyKlZK5mObgvd2dlnsAP+nnStyhV
HFIpKy3nsDO4JqrIgEhCsdpikSpbtdo18jUubV6z1kQ71CrRQtbi/WtdqxQEEtg
ZCJO2lPoIW
-----END CERTIFICATE-----
/*
/&
```

Procedure 2-4: Verify the Sample Private Key and Certificates

Modify and run the CIALSIGV job to verify that the RSA private key and certificates are synchronized.

```
// JOB CIALSIGV
// OPTION SYSPARM='00'      SysId of main TCP/IP partition
// LIBDEF *,SEARCH=lib.sublib
// EXEC CIALSIGV,SIZE=CIALSIGV,PARM='PRD2.SSLKEYS.SAMPLE03'
/*
/&
```

Step 3: Create Custom Files

This step consists of Procedures 3-1 through 3-6. Use these procedures to create a custom RSA private key file and certificate files for your system. The CIALSRVR, CIALCERT, CIALROOT, CIALSIGV, and CIALGPRV utilities catalog and access the private key, server certificate, and root certificate files. Procedure 3-6 creates a backup of the RSA private key file.

Store the private key and certificate files you create in the VSE library you named in "Step 1: Create a Key Sublibrary." The library, sublibrary, and member names for the private key and certificates you use must be specified with PARM= on the // EXEC statement in each utility.

Note:

The RSA private key file is encrypted using a default key when it is stored on the VSE system (Procedure 3-1). To increase the security of the private key file, create a unique encryption key for each VSE installation. See the section "Protecting the RSA Private Key File," page 33, for information on creating a unique key.

Procedure 3-1: Create a Unique RSA Private Key File

Choose one of the tools in the table below to generate the RSA private key, then go to the listed procedure.

Tool	Notes	Go to...
IBM's Keyman/VSE Java utility	For information about using this utility, contact IBM, or refer to IBM's z/VSE downloads page. See the web links in "Overview" on page 25.	"Keyman/VSE Utility" (page 30)
Crypto coprocessor (CC) card	A CC card allows you to create an RSA private key and write it to z/VSE directly without using Keyman/VSE. Supported key sizes: 1024, 2048, 4096.	"Crypto Coprocessor (CC) Card" (page 30)

Keyman/VSE Utility. To use this utility, do the following:

1. Modify the sample job CIALSRVR.JCL and run it on the target VSE system. This job stores the generated RSA public/private key structure that is sent to VSE.

```
// JOB CIALSRVR
// OPTION SYSPARM='00'   SysId of main TCP/IP partition
// LIBDEF PHASE,SEARCH=lib.sublib
// EXEC CIALSRVR,SIZE=CIALSRVR,PARM='LIB.SUBLIB.MEMNAME'
SETPORT 6045
/*
/&
```

2. Use the Keyman/VSE utility to generate an RSA key. It connects to VSE, generates the RSA public/private key, and sends the key to the target VSE system. The CIALSRVR program that is running on VSE then validates and writes the key.

   The CIALSRVR program issues a message every 30 seconds while it waits for the PC to generate and send the key. The program shuts down automatically after it receives the key. You can also force the CIALSRVR process to stop by issuing the command "MSG *xx*,DATA=SHUTDOWN" to the partition in which the process is running.

   It is important to keep the private key file secret because it contains the RSA private key data.

Crypto Coprocessor (CC) Card. To use a CC card, modify the sample job below and run it on the target VSE system. The card generates a key and stores it on the VSE system. To verify that a CC card is available, see the section "Verifying the Crypto Express Hardware Card" on page 43.

```
// JOB CIALSRVR
// OPTION SYSPARM='00'      SysId of main TCP/IP partition
// LIBDEF PHASE,SEARCH=lib.sublib
// EXEC CIALSRVR,SIZE=CIALSRVR,PARM='LIB.SUBLIB.MEMNAME'
GENRSAPK keysize
/*
/&
```

The *keysize* is the bit size of the private key. The valid values are 1024, 2048, and 4096.

Procedure 3-2: Create a Certificate Request

1. Use the IBM Keyman/VSE utility to generate a certificate request.

2. Submit the request to any certificate authority (CA) to obtain a digital certificate. When you request the certificate, make sure that it is a base64-encoded, X.509v3 SSL server certificate. The CA then issues a certificate for the TLS/SSL for VSE server and signs the certificate with its RSA private key.

   Important:

   The certificate request contains the RSA public key that VSE uses during encryption negotiations. The RSA private key is used to sign the certificate request, but the contents of the VSE private key are *not* contained in it and must be kept secret. The CA verifies the signature in the request and issues a certificate containing the VSE's RSA public key. Then it signs the issued certificate with its (the CA's) private key. The CA guarantees the authenticity of the certificate it issues that contains VSE's RSA public key. All certificate profiles and key and cryptographic formats are defined by the IETF's PKIX working group.

Procedure 3-3: Install the Issued Certificate

1. After you obtain the certificate from the CA, use a PC text editor (such as NOTEPAD.EXE on Windows® operating system) to look at it. If it is ASCII text that you can display, and it looks similar to the certificate in the sample job CIALCERT.JCL, then you have a base64-encoded certificate. (See the sample job in Procedure 2-2.)

   You can also display its contents from a PC. It should have a .CRT or .CER file extension, and on Windows you can double-click on it to open it and examine its contents.

   If the certificate is in binary format, you must convert it to base64 encoding with a utility available on Windows or Linux. Refer to Linux or Windows documentation for details on converting a binary certificate to a base64-encoded certificate.

2. Modify the sample job CIALCERT.JCL to contain the issued certificate. The certificate data should replace the text between the "BEGIN CERTIFICATE" line and the "END CERTIFICATE" line in the file.

3. Run your modified CIALCERT.JCL job on the target VSE system.

Procedure 3-4: Install the Certificate Authority's Root Certificate

1. Obtain the root certificate in text format from the CA. You must use the same CA that issued the production server certificate. The CA root certificate is installed and used by the VSE system to verify the signature in the issued certificate.

2. Use a PC text editor to view the certificate file. If the certificate is in binary format, you must convert it to base64 encoding with a utility available on Windows or Linux. Refer to Linux or Windows documentation for details. Verify that the certificate looks similar to the one in the job CIALROOT.JCL. (See the sample job in Procedure 2-3.)

3. Modify the sample job CIALROOT.JCL to contain the root certificate. The certificate data should replace the text between the "BEGIN CERTIFICATE" line and the "END CERTIFICATE" line in the file.

4. Run your modified CIALROOT.JCL job on the target VSE system.

Procedure 3-5: Verify the Certificate

1. Modify the sample job CIALSIGV.JCL with your information. (See the sample job in Procedure 2-4.)

2. Run your modified job on the target VSE system. The signature contained in the VSE certificate is validated using the CA's public key. This step verifies that the associated CA issued the certificate.

Your VSE system is now ready to use the following components:

- TLSD daemon; see "Defining the TLS/SSL Daemon," page 36
- TLS/SSL for VSE Application Programming Interface (API)
- CryptoVSE API
- Common Encryption Cipher Interface

The APIs listed above are documented in the *TCP/IP FOR VSE Programmer's Guide*.

Procedure 3-6: Back Up the RSA Private Key File

Use this optional procedure to back up an RSA private key file. This procedure allows you to securely transport the encrypted key to another VSE system.

1. Modify the sample job CIALGPRV.JCL with your system's information.

2. Run the modified CIALGPRV job on the VSE system that contains the key you want to copy. The encrypted RSA private key file is written to SYSLST using base64 printable character encoding. This encrypted data can then be used as input to the CIALPRVK utility.

3. To install the key on another VSE system, modify the CIALPRVK sample job with information for that system. (See the sample job in Procedure 2-1.) Specify the *LIB.SUBLIB.MEMNAME* in which to install the key, and replace the sample private key data with the data generated by the CIALGPRV utility in step 2 above.

4. Run the modified CIALPRVK job on the target VSE system.

Protecting the RSA Private Key File

The security and integrity of the TLS and SSL protocols depend on keeping the contents of the RSA private key private! If the RSA private key is compromised, then all communications using TLS/SSL could also be compromised. A separate lib.sublib must be created as described in "Step 1: Create a Key Sublibrary," page 26. This lib.sublib should not be used for any other purpose.

Also, the RSA private key is encrypted automatically when it is stored on the VSE system. A default internal key is used to encrypt the stored key, but we recommend that you create a unique key value for each installation. When CIALSRVR starts up (see the jobs in Procedure 3-1), it attempts to CDLOAD a CIALEXIT.phase. If this phase does not exist, the program uses a default password phrase to encrypt the RSA private key file. You optionally can create a unique CIALEXIT.phase to protect the contents of the RSA private key file.

To create a unique CIALEXIT.phase for an installation, modify and run the following sample job. Then, perform the procedures in "Step 3: Create Custom Files," page 29, to create the custom files.

Note:

The Keyman/VSE tool supports uploading a private key using a CIALEXIT phase. It will prompt for the CIALEXIT passphrase when uploading the key into a PRVK.

```
// JOB CIALEXIT SAMPLE
// OPTION CATAL
// LIBDEF *,CATALOG=CSILIB.DRSTEST
// EXEC ASMA90,SIZE=ASMA90
         PUNCH    ' PHASE CIALEXIT,* '
CIALEXIT CSECT
CIALEXIT AMODE    31
CIALEXIT RMODE    ANY
         PRINT    GEN
         STM      R14,R12,12(R13)          Save callers regs
         LR       R12,R15                  Set up base reg
         USING    CIALEXIT,R12             Establish addressability
         ST       R13,SAVEAREA+4           Store callers R13
         LA       R13,SAVEAREA             Provide a save area
         C        R0,=F'1'                 Is this a get pass phrase?
         BE       GETP1000                 bif yes
         C        R0,=F'2'                 Is this get key for rsa?
         BE       GETK1000                 bif yes
         LA       R15,BADREQST             Tell caller we failed
         B        MAIN9000                 Exit now
*
* * Get the private key file pass-phrase...
GETP1000 DS       0H
         L        R0,GETPHRSL              Length of phrase
         LA       R1,GETPHRST              Address of phrase
         B        MAIN8000                 Return to caller
*
* * Get key used to encrypt the RSA private key...
GETK1000 DS       0H
         LA       R0,AES256KL              Length of key (AES-256)
         LA       R1,AES256KY              Address of key (AES-256)
         B        MAIN8000                 Return to caller
*
MAIN8000 DS       0H
         LA       R15,GOODNEWS             Teller called success
MAIN9000 DS       0H
         L        R13,SAVEAREA+4
         LM       R2,R12,28(R13)           RESTORE CALLERS R2-R12
         L        R14,12(R13)              RESTORE R14
         BR       R14                      RETURN TO CALLER
*
SAVEAREA DS       18F
```

(continued next page)

```
*
* * The PC client generating the RSA private key
* * must enter this phrase before he or she is
* * allowed to send a RSA private key to the VSE CIALSRVR server.
* * * The SHA-1 hash on it is also stored with the RSA private key
* * * and is checked whenever the private key is read...
GETPHRSL DC       F'28'                    Length of phrase(max 64)
GETPHRST DC       C'The cow jumped over the moon'
*
* * The below is a sample AES-256-CBC key value used to encrypt
* * the RSA private key data that is stored on the VSE system.
AES256KY DC       XL16'1234567890ABCDEF0123456789ABCDEF'
         DC       XL16'A123456789ABCDEFB123456789ABCDEF'
         DC       XL16'7123456789ABCDEFA123456789ABCD0F'
AES256KL EQU      *-AES256KY
*
         LTORG    ,
*
* * EQUATES
GOODNEWS EQU      0
BADREQST EQU      1
*
R0       EQU      0
R1       EQU      1
R2       EQU      2
R3       EQU      3
R4       EQU      4
R5       EQU      5
R6       EQU      6
R7       EQU      7
R8       EQU      8
R9       EQU      9
R10      EQU      10
R11      EQU      11
R12      EQU      12
R13      EQU      13
R14      EQU      14
R15      EQU      15
         END      CIALEXIT
/*
// EXEC LNKEDT,SIZE=512K
/*
/&
```

Defining the TLS/SSL Daemon

Using the TLS/SSL daemon allows you to implement the TLS/SSL protocol quickly and easily. You do not need to modify existing applications running on the VSE system. To facilitate this implementation, processing is added at the front end of the application. Instead of allowing straight communication between the client software and the server running on VSE, TLS/SSL for VSE intercepts the communication on a secured port and manages the session directly with the client.

There are advantages and disadvantages to this approach. The biggest advantage is that no modifications are required to existing applications. The biggest disadvantage is that the approach may not work with some types of applications. For those applications, TLS/SSL for VSE provides an API that can be used to SSL-enable an application. See the *TCP/IP FOR VSE Programmer's Guide* for information on using TCP/IP FOR VSE APIS.

In this section, we present four methods of implementing a TLS/SSL daemon:

- Implementing TLS/SSL in pass-through mode
- Implementing a TLS/SSL-enabled HTTPD server
- Implementing a TLS/SSL-enabled HTTPD server with client authentication
- Implementing a TLS/SSL-enabled TELNETD server

Implementing TLS/SSL in Pass-Through Mode

To initialize pass-through mode, you need to define a TLS/SSL daemon for each TCP port that you want to secure. To do this, use the following two steps:

1. Define and initialize the TLS/SSL daemon. The following sample command defines a daemon that communicates with a TLS/SSL-enabled client. Each line is labeled. Use the same library, sublibrary, and member names you defined in "Step 1: Create a Key Sublibrary, page 26."

```
DEFINE TLSD,ID=TLSDTELNET, -     Id of this TLS/SSL daemon
           PORT=2020, -          Secure port
           PASSPORT=3020, -      Port the data is passed to
           CIPHER=0A0908, -      Allowed cipher suites
           CERTLIB=PRD2, -       Library name
           CERTSUB=SSLKEYS, -    Sublibrary name
           CERTMEM=SAMPLE03, -   Member name
           TYPE=1, -             SSL server application
           MINVERS=0300, -       Minimum version required
           DRIVER=SSLD           Driver phase name
```

The keyword parameters are as follows:

Parameter	Description
ID=	An arbitrary 1- to 16-character identifier for this SSL daemon.
PORT=	Port on which TLS/SSL for VSE listens for all TCP traffic.
PASSPORT=	The TCP port to which the TLS/SSL for VSE daemon passes traffic. If it is the same as PORT= value, then the defined TLS/SSL daemon passes the security parameters to a daemon that has directly implemented the TLS/SSL for VSE API. The corresponding daemon uses the same port as the TLS/SSL daemon.

Parameter	Description
CIPHER=	A value corresponding to a supported cipher suite. The valid values for the **SSL30**, **TLS 1.0**, and **TLS 1.1** protocol versions are as follows: 01 for RSA-NULL-MD5 02 for RSA-NULL-SHA 08 for RSA-SDES040-SHA 09 for RSA-SDES056-SHA 0A for RSA-TDES168-SHA 2F for RSA-AES128-SHA 35 for RSA-AES256-SHA When **TLS 1.2** is used, the supported cipher suites are listed below. These suites do not support the DES algorithm. TLS 1.2 support also requires IBM's hardware CP Assist for Cryptographic Function (CPACF) feature. 2F for RSA_AES128CBC_SHA160 35 for RSA_AES256CBC_SHA160 3C for RSA_AES128CBC_SHA256 3D for RSA_AES256CBC_SHA256 You can specify multiple cipher suites by concatenating values. Clients must use one or more of the specified cipher suites. This enforces the desired level of security in a critical or sensitive application. The TLS/SSL protocol allows the client to send a list of preferred cipher suites. The server selects one from the list and replies with the cipher to be used. TDES (for other than TLS 1.2) or AES128 is sufficient for most applications, but the AES256 provides the strongest encryption for more sensitive applications.
CERTLIB= CERTSUB= CERTMEM=	The library, sublibrary, and member names for the private key and certificates to be used by the application defined on this port. If these names are omitted, the default sequential disk files are used.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>TYPE=</td><td>The TLS/SSL application type. Valid values are:
0 for a client with authentication
1 for a server with no client authentication
2 for a server with client authentication
3 for a client with no authentication.</td></tr>
<tr><td>MINVERS=</td><td>A hexadecimal value that sets the minimum version of the TLS/SSL protocol that will be used for the HTTPD, TELNETD, and TLS/SSL pass-through daemons.
The valid values are as follows.
<table>
<tr><th>Value</th><th>Notes</th></tr>
<tr><td>0303</td><td>This is the TLS 1.2 standard documented in RFC5246. It is the most secure, but clients must support it for this to work.</td></tr>
<tr><td>0302</td><td>This is the TLS 1.1 standard documented in RFC4346.</td></tr>
<tr><td>0301</td><td>This is the TLS 1.0 standard documented in RFC2246. This version is more secure than SSL30, and most clients support it. It should be the preferred setting.</td></tr>
<tr><td>0300</td><td>This is the older, less secure version of the original SSL protocol (SSL30), but it provides the most flexibility for clients to connect in to this server.</td></tr>
</table>
Note: The MINVERS value you choose should be based on your installation’s security policies. The above recommendations are meant to be informative to assist, but ultimately it is your choice based on your environment.</td></tr>
<tr><td>DRIVER=</td><td>The phase name of the TLS/SSL daemon.</td></tr>
</table>

2. Install and test with the client software. You must follow the client’s installation instructions, which usually include configuring the port that corresponds with the PORT= parameter specified on the TCP/IP DEFINE TLSD command.

Implementing a TLS/SSL-Enabled HTTPD Server

The TCP/IP FOR VSE HTTP daemon has implemented the TLS/SSL for VSE API with native-mode support. The definition is as follows. Each line is labeled. Use the same library, sublibrary, and member names you defined in "Step 1: Create a Key Sublibrary" on page 26.

```
DEFINE TLSD,ID=HTTPDSSL, -          ID of this TLS/SSL daemon
            PORT=443, -             Default HTTPS port
            PASSPORT=443, -         Native support
            CIPHER=0A0908, -        Allowed cipher suites
            CERTLIB=PRD2, -         Library name
            CERTSUB=SSLKEYS, -      Sublibrary name
            CERTMEM=SAMPLE03, -     Member name
            MINVERS=0300, -         Minimum version required
            TYPE=1, -               SSL server application
            DRIVER=SSLD             Driver phase name

DEFINE HTTPD,ID=HTTPDSSL, -         ID of this HTTP daemon
            PORT=443, -             Default HTTPS port
            CONFINE=YES, -          Confine to a specific lib
            DRIVER=HTTPD            Driver phase name
```

Note:

- The DEFINE TLSD PORT= parameter, the DEFINE TLSD PASSPORT= parameter, and the DEFINE HTTPD PORT= parameter all specify the same port number.
- The defined HTTPD uses the CIPHER and MINVERS values specified in the corresponding DEFINE TLSD command, and only one port is used.
- The browser uses HTTPS by default, and then it connects directly to port 443 as defined in the RFC2818 "HTTP over TLS" standard.

Implementing a TLS/SSL-Enabled HTTPD Server with Client Authentication

The TCP/IP FOR VSE HTTP daemon can also be configured to require client authentication. Clients that connect to the daemon must supply a client certificate that authenticates their identity. To require authentication, specify TYPE=2 on the DEFINE TLSD command, as shown in the following definition. No other changes are needed. Each line is labeled. Use the same library, sublibrary, and member names you defined in "Step 1: Create a Key Sublibrary" on page 26.

```
DEFINE TLSD,ID=HTTPDSSL, -       ID of this TLS/SSL daemon
            PORT=443, -          Default HTTPS port
            PASSPORT=443, -      Native support
            CIPHER=0A0908, -     Allowed cipher suites
            CERTLIB=PRD2, -      Library name
            CERTSUB=SSLKEYS, -   Sublibrary name
            CERTMEM=SAMPLE03, -  Member name
            MINVERS=0300, -      Minimum version required
            TYPE=2, -            SSL w/client authentication
            DRIVER=SSLD          Driver phase name

DEFINE HTTPD,ID=HTTPDSSL, -      ID of this HTTP daemon
            PORT=443, -          Default HTTPS port
            CONFINE=YES, -       Confine to a specific lib
            DRIVER=HTTPD         Driver phase name
```

When a web browser tries to connect to VSE, it is asked to supply a client certificate. Browsers generally support client certificates and provide simple methods of obtaining and storing client certificates that are accepted by a TLS/SSL for VSE server requiring client authentication. See the documentation supplied with your browser for information on using TLS/SSL client authentication.

Implementing a TLS/SSL-Enabled Telnet Server

The TCP/IP FOR VSE Telnet daemon has implemented the TLS/SSL API with native-mode support. A sample definition follows. Use the same library, sublibrary, and member names you defined in "Step 1: Create a Key Sublibrary," page 26.

```
DEFINE  TLSD,ID=TLSDTNDS, -
              PORT=992, -
              PASSPORT=992, -
              CIPHER=090A08, -
              MINVERS=0300, -
              CERTLIB=PRD2, -
              CERTSUB=SSLKEYS, -
              CERTMEM=SAMPLE03, -
              TYPE=1, -
              DRIVER=SSLD
SET TELNETD_BUFFERS=20

DEFINE TELNETD,ID=TELNVSSL,TCPAPPL=TELNLU,TARGET=DBDCCICS, -
               COUNT=3,MENU=TCPMENU1,TYPE=VTAM,POOL=YES, -
               PORT=992,DRIVER=TELNETD
```

Note:

- You can use $SOCKOPT settings to override the ciphers specified here. See the *TCP/IP FOR VSE Programmer's Guide*, "Appendix A: $SOCKOPT Options Phase," for information on setting the SSLCIPH option in a custom options phase.
- DEFINE TLSD must precede the DEFINE TELNETD commands.
- The PORT= and PASSPORT= values on DEFINE TLSD and the PORT= value on DEFINE TELNETD must all specify the same port number. You also must specify POOL=YES on the DEFINE TELNETD command.

You can secure effector Telnet daemons with TLS/SSL, as shown below.

```
DEFINE  TLSD,ID=TLSDTNDS, -
              PORT=992, -
              PASSPORT=992, -
              CIPHER=090A, -
              MINVERS=0300, -
              CERTLIB=PRD2, -
              CERTSUB=SSLKEYS, -
              CERTMEM=SAMPLE03, -
              TYPE=1, -
              DRIVER=SSLD
SET TELNETD_BUFFERS=20

* TN3270E Listener Daemon
DEFINE TELNETD,ID=LSTNSSL,TN3270E=L,PORT=992,GROUP=ONE, -
              POOL=YES,DRIVER=TELNETD
* TN3270E Effector Daemon
DEFINE TELNETD,ID=EFFSSL,TCPAPPL=TELNLU,TARGET=DBDCCICS, -
              PORT=992,COUNT=3,MENU=TCPMENU1,TYPE=VTAM, -
              POOL=YES,TN3270E=E,GROUP=ONE,DRIVER=TELNETD
```

Verifying the Crypto Express Hardware Card

Cryptography hardware cards provide enhanced performance for asymmetric (RSA) cryptography requests. TCP/IP FOR VSE's TLS/SSL code determines whether to use hardware instructions in RSA operations by checking an indicator in an operating system control block. VSE sets this indicator when a crypto card is installed. (Some z/VSE versions do not update this indicator.)

The TLS/SSL code checks for this indicator each time an application initializes the TLS/SSL environment in the current partition. If a card is installed, TLS/SSL uses it to perform RSA operations for any TLS/SSL application that uses that partition.

By default, the TLS/SSL code chooses whether to use a crypto card automatically. You can choose to always use hardware cryptographic instructions, or never use them, by setting an option in the $SOCKOPT options phase. For more information on setting options in this phase, see the *TCP/IP FOR VSE Programmer's Guide*, "Appendix A: $SOCKOPT Options Phase."

If a Crypto Express card is installed on your system, you must ensure that it is configured to perform RSA asymmetric operations. The IBM z/VSE operating system provides the configuration and support for this device. To verify that a card is available, issue the command "MSG FB,DATA=STATUS=CR". The example output below shows that a crypto card has been configured.

```
MSG FB,DATA=STATUS=CR
 FB 0011 CRYPTO DEVICE DRIVER STATUS:
 FB 0011   AP CRYPTO SUBTASK STARTED .......... : YES
 FB 0011   MAX REQUEST QUEUE SIZE ............. : 0
 FB 0011   MAX PENDING QUEUE SIZE ............. : 0
 FB 0011   TOTAL NO. OF AP REQUESTS ........... : 0
 FB 0011   NO. OF POSTED CALLERS .............. : 0
 FB 0011   AP-QUEUE INTERRUPTS AVAILABLE ...... : NO
 FB 0011   AP-QUEUE INTERRUPTS STATUS ......... : DISABLED
 FB 0011   AP CRYPTO POLLING TIME (1/300 SEC).. : 1
 FB 0011   AP CRYPTO WAIT ON BUSY (1/300 SEC).. : 75
 FB 0011   AP CRYPTO RETRY COUNT ............... : 5
 FB 0011   AP CRYPTO TRACE LEVEL ............... : 3
 FB 0011   TOTAL NO. OF WAITS ON BUSY ......... : 0
 FB 0011   CURRENT REQUEST QUEUE SIZE ......... : 0
 FB 0011   CURRENT PENDING QUEUE SIZE ......... : 0
 FB 0011   ASSIGNED APS : PCICC / PCICA ....... : 0 / 0
 FB 0011                  CEX2C / CEX2A ....... : 1 / 0
 FB 0011                  CEX3C / CEX3A ....... : 0 / 0
 FB 0011                  PCIXCC .............. : 0
 FB 0011     AP 19 : CEX2C   - ONLINE
 FB 0011   ASSIGNED AP QUEUE (CRYPTO DOMAIN)... : 14
 FB 0011 END OF CRYPTO DEVICE DRIVER STATUS
```

See the IBM *z/VSE V*x*R*x *Administration* manual for a list of available crypto-related commands.

The IBM PCIe cryptographic coprocessor provides a high-security, high-throughput cryptographic subsystem. The IBM 4765 Cryptographic Security Module was validated to FIPS 140-2, Overall Level 4, the highest level of security. (See FIPS certification number 1505 in the module list at http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm.)

More information on IBM's 4765 module is available at http://www-03.ibm.com/security/cryptocards/pciecc/overview.shtml.

Most RSA operations occur during TLS/SSL session initialization. To change hardware crypto-coprocessor settings, you must shut down and restart all TLS/SSL applications. The hardware crypto assists that were available when the application initialized the TLS/SSL environment will continue to be used until the application is cycled.

If a hardware Crypto Express card is not available on your system, TCP/IP FOR VSE performs RSA operations using software. The following limitations apply to RSA software operations:

- TCP/IP FOR VSE supports RSA-512 and RSA-1024 only. If the application uses RSA-2048 or RSA-4096, the RSA operations will fail.
- TCP/IP FOR VSE's RSA software implementation was developed for testing and is not FIPS certified.
- RSA software operations can increase CPU utilization significantly.

TCP/IP FOR VSE does not automatically switch to using software RSA if a crypto card becomes unavailable. If you remove a crypto card and you want to use software RSA, you must cycle the partition in which the application is running.

SHA-1 Phase Verification

Overview

This section explains how to use the SHA-1 Phase Verification utility (CIALSHPH). This utility verifies the integrity of phases by reading a phase into memory, calculating a SHA-1 hash (also referred to as a fingerprint), and comparing it to an established value from the time the phase was originally created. This check provides an absolute guarantee that the phase contents match the original, distributed value.

You can check the established, correct SHA-1 values for each phase distributed with the current TCP/IP FOR VSE product.

You can also use the CIALSHPH utility to verify the integrity of non-TCP/IP phases and members such as .OBJ files.

Before You Begin

A valid SSL product key must be installed before you can use the Phase Verification utility. See the section "Product Key," page 20, for more information.

Checking TCP/IP Phase Integrity

This procedure explains how to check the integrity of downloaded TCP/IP FOR VSE phases you have installed on your system.

Step 1: Create the SHA-1 Phase Table

TCP/IP FOR VSE is distributed with a phase that contains the SHA-1 values for each phase. The SHA-1 values are contained in a sample job with the file name of T*vrm*SHA1.JCL, where *vrm* is the version, release, and modification level of TCP/IP.

Keep in mind that if any maintenance is applied, the SHA-1 values will change, and it may be necessary to create an updated table of TCP/IP phases with their corresponding SHA-1 values. You can do this by contacting technical support for an updated T*vrm*SHA1 job that will be similar to the one shown below.

Sample T*vrm*SHA1.JCL

```
// JOB TvrmSHA1
// OPTION CATAL
// LIBDEF *,CATALOG=lib.sublib
// EXEC ASMA90,SIZE=ASMA90
         PUNCH    ' PHASE TvrmSHA1,* '
*
TvrmSHA1 CIALSHAT GEN
*
 DC C'$EDCTCPV',XL20'731F6EB4A207E63EB1A3F01A751430130ADA36B1'
 DC C'$SOCKDBG',XL20'B862B55465247A7015G1589DE8E56AD34CD7303A'
 DC C'$SOCKLST',XL20'86679BDC6D5D96F146DB2FA1E12BBDFFF6F13BE3'
 DC C'$SOCKOPT',XL20'73C169E4766D16109AFB5046588F62F0628FAB22'
 DC C'ASOCKET ',XL20'207D159D973EA9A1987596CC86CEAB5C60B56E70'
 DC C'CGILOAD ',XL20'38799CE977FC4DD6D62928C612EFD81DF041ED14'
 DC C'CHECKTCP',XL20'B9C8A87358D4103C6E2BA1BC0593DBE09122E0A9'
 . . . other phases . . .
 DC C'TELNETD ',XL20'D99B98151C2E9D87A5D73187262709931EF6B0FE'
 DC C'TELNET01',XL20'AF1923FA93317AB055D5B42B842160F0B1C86869'
 DC C'TRCTAPE ',XL20'7C35A30F76177BF73281709CE7C0C0CEBA61CA5C'
 DC C'TRIBTAPE',XL20'3E4EA717EE6EEC84F7ECCD8A48300BCF9DFF041E'
 DC C'VERCHECK',XL20'8BC669B41526EC2D68E380132F270B18927DA53A'
         DC       F'-1' End of SHA-1 hash table
         END
/*
// EXEC LNKEDT,SIZE=512K
/*
/&
```

Step 2: Verify Phase Integrity

The second step is to use the SHA-1 phase table (T*vrm*SHA1.PHASE) to verify the integrity of your installed phases. Execute the CIALSHPH program using the following job as an example:

```
// JOB CIALSHPH
// OPTION LOG
// LIBDEF PHASE,SEARCH=lib.sublib
// EXEC CIAlLSHPH,SIZE=CIALSHPH
VERCHECK TvrmSHA1
/*
/&
```

Replace the *lib.sublib* in this job with the library and sublibrary you used for the TCP/IP FOR VSE installation.

Note:

See the section "CIALSHPH Program Options," page 50, for information on UPSI options you can set for the CIALSHPH utility.

If the job completes with a return code of zero, it means that all the phases match the values contained in T*vrm*SHA1.PHASE, and the integrity of the installed phases is guaranteed.

If the job completes with a non-zero return code, it means that an integrity problem was detected. You must review the SYSLST output to determine the cause of the problem. The following example shows the messages that are issued for an out-of-sync phase:

```
$SOCKDBG FOUND IN lib.sublib SIZE=00000138
$SOCKDBG SHA1 HASH=DEE6471190BB604D034991447361CF01C866381A
$SOCKDBG SHA1 HASH DOES NOT MATCH SHA1 FROM T216SHA1
```

Out-of-sync errors may be acceptable. You can correct these errors as follows:

1. Copy the HASH= value from an issued message into your copy of T*vrm*SHA1.JCL.
2. Run the T*vrm*SHA1.JCL job to re-create the T*vrm*SHA1.PHASE.
3. Rerun the CIALSHPH job.

Checking Non-TCP/IP Member Integrity

This section contains two procedures that describe how to verify the integrity of any IBM, vendor, or customer member:

- Verifying multiple phases
- Creating a SHA-1 value for any single member

Verifying Phases

Use the following four steps to verify multiple non-TCP/IP phases.

Step 1: Specify the Phases

Edit and run the JCL below. You must replace the list of phase names in the example with the ones you want to verify. The job generates the phase CUSTSHA1.

For CATALOG=, replace *lib.sublib* with the library and sublibrary into which you want to catalog this phase.

For SEARCH=, replace *csilib.sublib* with the *lib.sublib* used for the TCP/IP FOR VSE installation.

In this example, PAYROLL1, PAYROLL2, and PAYWXYZ are the phases to be verified.

```
// JOB CUSTSHA1
// OPTION CATAL
// LIBDEF *,CATALOG=lib.sublib    <--lib.sublib for CUSTSHA1
// LIBDEF *,SEARCH=csilib.sublib  <--TCP/IP lib.sublib
// EXEC ASMA90,SIZE=ASMA90
         PUNCH    ' PHASE CUSTSHA1,* '
CUSTSHA1 CIALSHAT GEN
*
 DC C'PAYROLL1',XL20'00'
 DC C'PAYROLL2',XL20'00'
 DC C'PAYWXYZ ',XL20'00'
         DC       F'-1' End of SHA-1 hash table
         END
/*
// EXEC LNKEDT,SIZE=512K
/*
/&
```

Step 2: Create SHA-1 Values

Generate SHA-1 values for the listed phases by running the following job:

```
// EXEC CIALSHPH,SIZE=CIALSHPH,PARM='GENCODE'
VERCHECK CUSTSHA1
/*
```

This job outputs the generated SHA-1 value for each phase to SYSLST.

For example, the following lines might be output for the phase

```
PAYROLL1 FOUND IN PRODLIB.PAYROLL  SIZE=00035FD8
PAYROLL1 SHA1 HASH=731F6EB4A207E63EB1A3F01A751430130ADA36B1
 DC C'PAYROLL1',XL20'731F6EB4A207E63EB1A3F01A751430130ADA36B1'
```

Similar lines would be output for the other phases.

Step 3: Edit the CUSTSHA1 Assembly

Copy the lines containing "`DC C'xxxxxxx',XL20'yyyyyyyy'`" and paste them into the CUSTSHA1 assembly to replace the phase list. (See "Step 1: Specify the Phases" on page 47.) Then, run the assembly again and link-edit to create a new CUSTSHA1.PHASE.

Step 4: Run VERCHECK

Verify the phases as follows:

A. Run the following job using the regenerated CUSTSHA1 phase:

```
// EXEC CIALSHPH,SIZE=CIALSHPH
VERCHECK CUSTSHA1
/*
```

B. Examine the output for out-of-sync errors as explained in "Step 2: Verify Phase Integrity" on page 46.

Creating a SHA-1 Value for a Single Member

You can use the following SYSIPT commands after a "// EXEC CIALSHPH" statement to create a SHA-1 value for any single member.

Command	Description
MEMBSHA1 *any-phase*	Allows you to create a SHA-1 value for any phase. The phase may be from IBM, a vendor, or other sources.
MEMBTYPE *type*	Allows you to create a SHA-1 value for any non-phase library member. The member type you specify applies to all MEMBSHA1 commands that follow.

For example, the following job can be used to create the SHA-1 value for a library object deck (.OBJ) file:

```
// EXEC CIALSHPH,SIZE=CIALSHPH
MEMBTYPE OBJ
MEMBSHA1 IPNRSTUB
/*
```

This job step reads the file IPNTSTUB.OBJ and generates the SHA-1 value for it.

CIALSHPH Program Options

The CIALSHPH program has two useful UPSI options that are described in the table below.

UPSI Byte Setting	Description
// UPSI 1XXXXXXX	DEBUG ON. This setting provides diagnostic information. Use this option only when CSI Technical Support requests that you set it to collect data for problem analysis.
// UPSI XXXXXXX1	TERSE ON. This setting suppresses all correct comparisons. Only failing comparisons are printed to SYSLST.

Published Standards

TLS/SSL for VSE is based on published standards. You can obtain more information on these standards from the following sources:

- RSA security resources at https://www.rsa.com/en-us/perspectives/resources
- The U.S. National Institute of Standards and Technology (NIST) at http://csrc.nist.gov/groups/ST/
- RFC1321, RSA Data Security, Inc. MD5 Message-Digest Algorithm
- RFC2104, HMAC: Keyed hashing for message authentication
- RFC2202, Test Cases for HMAC-MD5 and HMAC-SHA-1
- RFC1113, Privacy enhancement for Internet electronic mail
- RFC2045, Multipurpose Internet Mail Extensions (MIME) Part One
- RFC2459, Internet X.509v3 PKI certificates
- RFC2246, TLS Standard at https://datatracker.ietf.org/wg/tls/charter/
- RFC4346, TLS 1.1 protocol standard.
- RFC5246, TLS 1.2 protocol standard.

References

For more information on TLS/SSL programming and security, see the following publications.

RSA Security's Official Guide to Cryptography

Steve Burnett and Stephen Paine
ISBN 0-07-213139-X
Osborne/McGraw-Hill

Applied Cryptography

Bruce Schneier
ISBN 0-471-11709-9
John Wiley & Sons

OS/390 V2R10.0 System Secure Sockets Layer Programming Guide and Reference

IBM document number SC24-5877-03

Security on IBM z/VSE

IBM form number SG24-7691-02
http://www.redbooks.ibm.com/redpieces/abstracts/sg247691.html

SSL and TLS: Designing and Building Secure Systems

Eric Rescorla
ISBN 0-201-61598-3
Addison-Wesley

SSL and TLS Essentials: Securing the Web

Stephen Thomas
ISBN 0-471-38354-6
John Wiley & Sons

3

SecureFTP for VSE

Overview

The standard FTP protocol is often used to transfer files over a TCP/IP network. This protocol requires one system to act as a client that issues commands to set up and control an independent data channel. This channel is then used to transfer files between the two systems. The only security provided is the optional use of a user ID and password during the initialization of the command connection. All commands, including the user ID and password, are passed across the network in the clear and are thus susceptible to abuse.

The SecureFTP for VSE feature provides user authentication, confidentiality, and data integrity by using digitally signed certificates, data encryption, and secure hash functions. These industry-standard cryptographic functions provide authentication, privacy, and integrity for commands and data transmitted using the FTP protocol by implementing Transport Layer Security (TLS) and Secure Sockets Layer (SSL) into FTP clients and servers running on the VSE platform.

The Internet Engineering Task Force (IETF) has officially replaced SSL with TLS as the protocol standard. TLS is compatible with SSL, but it contains security enhancements. SecureFTP for VSE implements the SSL 3.0, TLS 1.0, TLS 1.1, and TLS 1.2 standards for secure communications. Because some people still use the SSL acronym when discussing the TLS protocol, this chapter may use *SSL* to refer to both the SSL and TLS standards.

Protocols

SecureFTP for VSE relies on a number of integrated components, including

- PKI (Public Key Infrastructure) for identification
- RSA for key exchange algorithms
- AES, TDES, and DES for data encryption

- MD5, SHA-1, and SHA-256 for message hashing
- HMAC for message authentication

SecureFTP for VSE implements numerous industry-standard protocols, including

- RFC2246 (Transport Layer Security) for TLS 1.0
- RFC4346 for TLS 1.1
- RFC5246 for TLS 1.2
- RFC1321 (MD5 message-digest algorithm)
- RFC2104 (HMAC)
- RFC2459 (X.509v3 PKI certificates)

By using industry-standard algorithms, you are assured of compatibility with a wide variety of other platforms that also support secure TLS/SSL-enabled FTP servers and clients.

SecureFTP for VSE is based on the IETF document, “Securing FTP with TLS” (RFC4217). This document is an officially accepted standard for securely transmitting files with the FTP protocol.

Setup

The SecureFTP for VSE feature is included with the TCP/IP FOR VSE product. This section describes how to set up your system to use this feature.

Product Key

You must obtain and apply a product key to enable this feature. To check whether a valid SecureFTP key is installed, see the section "Product Key," in chapter 2, on page 20.

For instructions on applying product keys, see the *TCP/IP FOR VSE Installation Guide*, chapter 3, "Installation."

Requirements

Additional requirements for using SecureFTP for VSE are as follows:

- The TLS/SSL for VSE feature must be enabled. See chapter 2, "TLS/SSL for VSE," on page 20 for information on setting up an application to use TLS/SSL.
- To run an external SecureFTP server, the static or dynamic partition you use must have a minimum virtual storage capacity of 8 MB.
- To run a SecureFTP server or client on VSE, you must ensure that both the foreign FTP server and FTP client are TLS/SSL capable. In addition, you must determine the protocol level (TLS 1.2, TLS 1.1, TLS 1.0, or just SSL30) of the server and each client that connects to it.

Running an External SecureFTP Server

The following example job runs an external SecureFTP server on VSE.

```
// JOB FTPBSSLS
// OPTION LOG,PARTDUMP
// OPTION SYSPARM='00'
*
* * This is a SecureFTP server for VSE.
* * Any foreign clients connecting to it must be SSL enabled.
*
// LIBDEF *,SEARCH=yourlib.sublib
// EXEC FTPBATCH,SIZE=FTPBATCH,PARM='TLS=SERVER,FTPDPORT=990'
SET TLS10 PRIVATE LIB.SUBLIB.SAMP1024 NOCLAUTH
/*
/&
```

This job contains the following FTPBATCH parameters, specified by PARM=, for a TLS/SSL server.

Parameter	Description
TLS=SERVER	Required on PARM= to initialize the TLS/SSL environment.
FTPDPORT=990	Specifies the port number to which clients can connect. An *implicit* TLS/SSL client uses a port number of 990. An *explicit* TLS/SSL client uses other port numbers. An implicit client is more secure because the TLS/SSL handshake occurs just after the connection is opened, before any messages are exchanged. With explicit clients, the TLS/SSL handshake does not occur until the AUTH TLS/SSL command is received.

The SET command operands in the example job are as follows. These operands are positional.

SET Operand	Description
TLS10	The first operand sets the minimum level of the TLS/SSL protocol that clients are allowed to use to connect in to this server. TLS 1.2 has the most enhanced security features, but not all clients support it. Specifying TLS10, for example, requires clients connecting in to use the TLS 1.0 or higher protocol version. The valid values are as follows.

Value	Synonyms
TLS12	(none)
TLS11	(none)
TLS10	TLS*, TLSV1*, TLS31*
SSL30*	SSL*, SSLV3*

*May be unsupported in a future release. For compatibility with older releases, “SSL” and “TLS” (with no numbers) are still accepted.

SET Operand	Description
PRIVATE	The valid values are PRIVATE or CLEAR. This operand controls the data connection security. The command connection is always secured, but encryption is optional on the FTP data connection. You may only want to secure the user ID, passwords, and other commands and avoid the extra overhead for the bulk data transfer. Specifying PRIVATE requires clients to negotiate a secure connection for the data transfer. Specifying CLEAR causes the data connection to be sent and received without a secure connection.
LIB.SUBLIB. SAMP1024	This is the library, sublibrary, and member name for the server certificate, root certificate, and private key to be used. You can also specify SDFILES if the server/root certificate and the private key are contained in sequential disk files. See “Step 1: Create a Key Sublibrary,” page 26, for information on defining these files or a library routine.
NOCLAUTH	NOCLAUTH: No client authentication. CLNTAUTH can also be specified to force the use of client authentication. When CLNTAUTH is specified, the server requires the client to send a certificate that validates the identity of the client connecting into the server. See also the discussion of client certificates in the section “SecureFTP Certificates” on page 61.

Running an Internal SecureFTP Server

The following command defines an internal SecureFTP server on VSE.

```
DEFINE FTPD,ID=FTPDSSL,PORT=990,COUNT=1,TIMEOUT=9000,UNIX=NO, -
      DRIVER=FTPDAEMN,SSL=YES,SSLKEY=DRSLIB.SSLKEYS.SAMP1024, -
      SSLVERSION=SSL30,SSLCIPHER=ALL,SSLDATACONN=PRIVATE, -
      SSLMODE=IMPLICIT
```

DEFINE FTPD and all of its options are documented in the *TCP/IP FOR VSE Command Reference*.

The SecureFTP-related parameters are as follows.

Parameter	Description
SSL=	The valid values are as follows: NO—disables SSL for a server (the default). YES—enables SSL for this FTP server without client authentication. YESCLAUTH—enables SSL for this FTP server and requires that other clients connecting into it supply a client certificate (client authentication). With the YES option, clients are not authenticated, but with YESCLAUTH, the server requests that the client provide a certificate that it then uses to authenticate the identity of the client connecting into it. See additional information at the end of this section.
SSLKEY=	The lib.sublib.member from which SSL reads the member types (.prvk, .cert, .root).

Parameter	Description
SSLCIPHER=	For the **SSL30, TLS 1.0,** and **TLS 1.1** protocols: ALL—(the default) Use default ciphers; see note. AES—these AES cipher suites are accepted: 35 RSA_AES256CBC_SHA 2F RSA_AES128CBC_SHA DES—these DES cipher suites are accepted: 0A RSA_3DESCBC_SHA 09 RSA_DESCBC_SHA 08 RSA_DES40CBC_SHA1 STRONG—strong cipher suites are accepted: 35 RSA_AES256CBC_SHA 2F RSA_AES128CBC_SHA 0A RSA_3DESCBC_SHA MEDIUM—medium cipher suites are accepted: 2F RSA_AES128CBC_SHA 0A RSA_3DESCBC_SHA 09 RSA_DESCBC_SHA WEAK—these weak cipher suites are accepted: 09 RSA_DESCBC_SHA 08 RSA_DES40CBC_SHA1 NULL—these null cipher suites are accepted: 02 RSA_NULL_SHA1 01 RSA_NULL_MD5 HARDWARE—dynamic; based on detected hardware facilities. For **TLS 1.2:** the valid values are as follows[1]. ALL—(the default) Use default ciphers; see note. STRONG—this strong cipher suite is accepted: 3D RSA_AES256CBC_SHA256 MEDIUM—medium cipher suites are accepted: 35 RSA_AES256CBC_SHA160 3C RSA_AES128CBC_SHA256 WEAK—this weak cipher suite is accepted: 2F RSA_AES128CBC_SHA160 **Note:** The default ciphers are set by the SSLCIPH keyword in $SOCKOPT. By default, all supported cipher suites are allowed. For more information, see “Appendix A: $SOCKOPT Options Phase” in the *TCP/IP FOR VSE Programmer’s Guide*.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>SSLVERSION=</td><td>The minimum version of the SSL or TLS protocol that clients can use when connecting in to this SecureFTP daemon. The valid values are below.
<table>
<tr><th>Value</th><th>Synonyms</th></tr>
<tr><td>TLS12</td><td>0303</td></tr>
<tr><td>TLS11</td><td>0302</td></tr>
<tr><td>TLS10</td><td>0301, TLSV1*</td></tr>
<tr><td>SSL30</td><td>0300, SSLV3*</td></tr>
</table>
*Supported for compatibility with older releases.
TLS 1.2 is the most enhanced, but not all clients can support it. Each site must evaluate this requirement based on the specific environment.</td></tr>
<tr><td>SSLDATACONN=</td><td>Valid values are CLEAR and PRIVATE.
CLEAR causes file data to be sent and received without encryption. PRIVATE secures all file data sent and received with encryption.
SecureFTP server requires that all commands and responses be secured using encryption, but securing the data connection is optional. The increased overhead of encryption can adversely affect the performance of the data connection. Each site must evaluate the need to secure the data connection.</td></tr>
<tr><td>SSLMODE=</td><td>Sets the negotiation mode. The valid values are
IMPLICIT—This is the default setting. The SSL/TLS negotiation is performed immediately when the connection is established (before the 220-welcome message is sent).
EXPLICIT—This setting delays the SSL/TLS negotiation until an AUTH command is received. This is somewhat less secure because the initial 220-welcome message is sent in clear text.</td></tr>
</table>

[1] TLS 1.2 support requires IBM's hardware CP Assist for Cryptographic Function (CPACF) feature.

SecureFTP Certificates

Both external and internal SecureFTP servers require a .cert file that contains the *server certificate*. An external server uses the SET command to identify this certificate's location. An internal server uses the SSLKEY parameter to identify the location. The SSL protocol always requires the server to provide a certificate to the client connecting into it. One of the purposes of this requirement is to prevent a security exposure commonly known as *spoofing*.

The external and internal SecureFTP servers can also require clients connecting into them to provide a *client certificate* to verify each client's identity. To require clients to identify themselves with a client certificate, the external server uses the SET operand CLNTAUTH, and the internal server uses the keyword SSL=YESCLAUTH.

The server and client certificates also contain a digital signature from a certificate authority such as Verisign, Inc. The certificate is then used to verify the identity of the client or server. This process of using certificates, digital signatures, and certificate authorities is known as *Public Key Infrastructure* (PKI).

Running a SecureFTP Client

The following job shows how to run a SecureFTP client on VSE.

```
// JOB FTPBSSLC
// OPTION LOG,PARTDUMP,NOSYSDMP
// OPTION SYSPARM='00'
*
* * This is a SecureFTP client for VSE.
* * The foreign FTP server must be SSL/TLS enabled...
*
// EXEC FTPBATCH,SIZE=FTPBATCH,PARM='TLS=CLIENT'
[SET TLS10 PRIVATE LIB.SUBLIB.MEMBER]
LOPEN
LUSER local_userID
LPASS local_password
OPEN foreign_ip_address port_number [ssl_mode tls_protocol]
AUTH SSL
[CLEAR|PRIVATE]
USER foreign_userID
PASS foreign_password

<your normal FTP commands>

CLOSE
LCLOSE
QUIT
/*
```

The unique commands and operands for this TLS/SSL client are listed in the table below. The SET command and OPEN command operands are positional.

Command/Operand	Description
PARM='TLS=CLIENT'	Required on EXEC FTPBATCH to initialize the SSL/TLS environment.
SET TLS10 PRIVATE *LIB.SUBLIB.MEMBER*	This SET statement enables client authentication. The operands are positional. This command is usually used only when FTPBATCH is configured as a server. But when it is configured as a client with client_authentication, the only operand that is used is the *LIB.SUBLIB.MEMBER* to establish the client certificate that will be used if the server being connected to requests a certificate. Use this command only if the foreign FTP server requires client authentication. "TLS10" and "PRIVATE" are placeholders. *LIB.SUBLIB.MEMBER* is the library, sublibrary, and member names you used to store the RSA private key and certificate files. See chapter 2, "TLS/SSL for VSE," for details.
foreign_ip_address	Operand on the OPEN command. Specifies the foreign server's IP address.
port_number	Operand on the OPEN command. Specifies the server port to which this client connects. "990" denotes an implicit SSL client by default. Other port numbers denote an explicit SSL client by default. Implicit clients are more secure because the SSL handshake occurs just after the connection is opened. With explicit clients, the handshake does not occur until the AUTH SSL command is processed.
ssl_mode	Operand on the OPEN command. The valid values are EXPLICIT and IMPLICIT. This value overrides a port's default mode. For example, port 990 is implicit by default. Specifying EXPLICIT changes the mode from implicit to explicit. See also *port_number* above.

<table>
<tr><th>Command/Operand</th><th>Description</th></tr>
<tr><td>tls_protocol</td><td>Operand on the OPEN command. Sets the protocol version that the client proposes to the server. The valid values are as follows. The default is TLS10.
<table>
<tr><th>Value</th><th>Synonyms</th></tr>
<tr><td>TLS12</td><td>(none)</td></tr>
<tr><td>TLS11</td><td>(none)</td></tr>
<tr><td>TLS10</td><td>TLS*, TLSV1*, TLS31*</td></tr>
<tr><td>SSL30*</td><td>SSL*, SSLV3*</td></tr>
</table>
*May be unsupported in a future release. For compatibility with older releases, “SSL” and “TLS” (with no numbers) are still accepted.</td></tr>
<tr><td>AUTH SSL</td><td>The valid operands are SSL and TLS.
For explicit clients, this command must immediately follow the OPEN command.
Note: The AUTH command is not used with implicit clients (for example, port 990 by default).</td></tr>
<tr><td>PRIVATE | CLEAR</td><td>Use PRIVATE (the default) if the foreign SSL FTP server is using a private (secured) data connection.
Use CLEAR if the foreign SSL FTP server is using an unsecured data connection. These commands call PROT and LPROT with either the ‘P’ or the ‘C’ argument. See the TCP/IP FOR VSE User Guide for more information on PROT and LPROT.
You can switch between PRIVATE and CLEAR for local and foreign FTP server commands. See the client example below for more information.</td></tr>
</table>

Example The following SecureFTP client example shows how data connections can be switched between CLEAR and PRIVATE. In this example, the DIR command requests a directory listing from the foreign FTP server, and the LDIR command requests a directory listing from the local FTP server. The directory-listing data comes through the data connection.

```
// EXEC FTPBATCH,SIZE=FTPBATCH,PARM='TLS=CLIENT'
LOPEN
LUSER local_userID
LPASS local_password
LAUTH SSL
OPEN foreign_ip_address 990 IMPLICIT TLS10
USER foreign_userID
PASS foreign_password
CLEAR
LDIR
PRIVATE
DIR
PRIVATE
PUT  %SAM2,SAM,EPIC FTPBSSLC.TXT
QUIT
/*
/&
```

Notice that CLEAR is issued just before the LDIR command and PRIVATE is issued just before the DIR command. The commands in this example assume that the foreign FTP server requires a private data connection.

For more background on local and foreign FTP server connections, see the diagrams in the introduction to the FTP chapter in the *TCP/IP FOR VSE User Guide*.

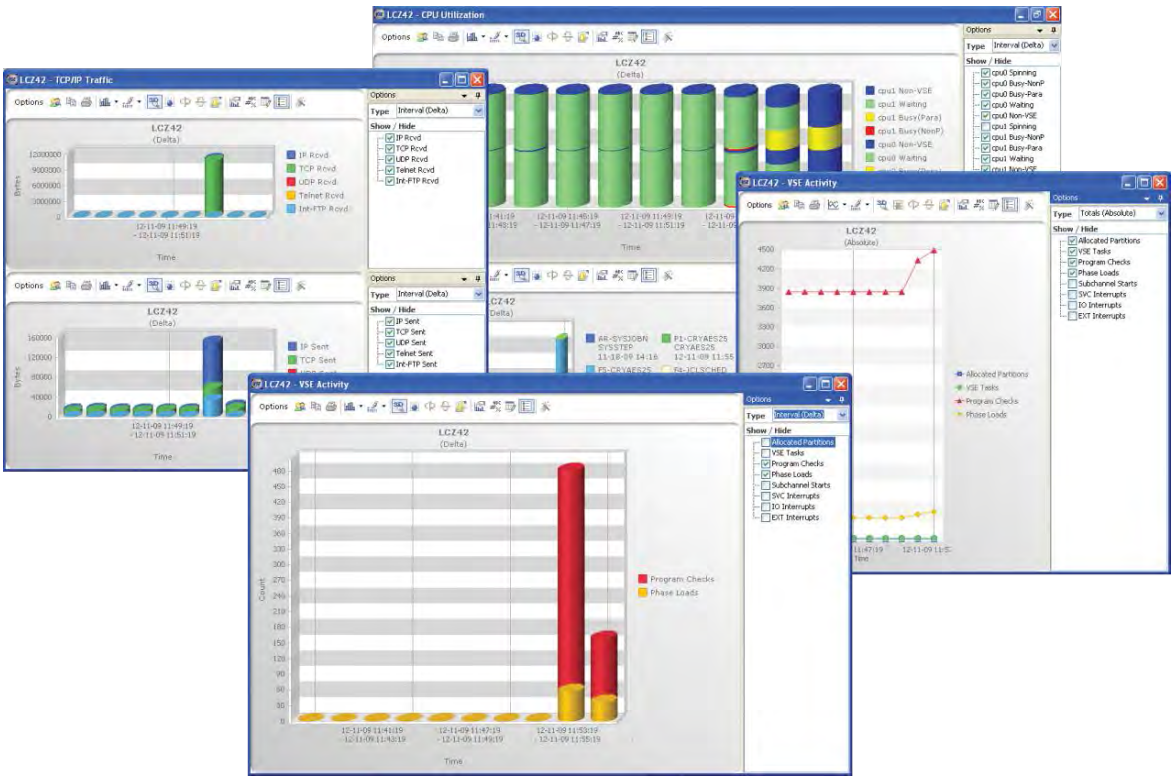
4

See-TCP/IP for VSE

Overview

See-TCP/IP for VSE is a system and network monitor that helps you identify and correct system bottlenecks to improve system performance. It consists of a VSE server and a personal computer (PC) client.

The VSE server runs on the mainframe. It listens for PC client connections and provides raw performance data related to the work done on the VSE system. The PC client collects this raw performance data and stores it in a database in a meaningful way. These data are used to monitor VSE and create charts and tables that detail system performance.



VSE Server Setup

See-TCP/IP for VSE is distributed with the TCP/IP FOR VSE product. This section explains how to set up your VSE system to use this feature.

The See-TCP/IP for VSE feature requires that you download certain components from CSI International's website and install them on a PC. This procedure is described in the section "PC Setup" on page 71.

Product Key

You must obtain and apply a See-TCP/IP for VSE product key. To check whether this key is installed, see the section "Product Key" in chapter 2, "TLS/SSL for VSE," on page 20.To obtain a See-TCP/IP for VSE product key, contact CSI International at sales@csi-international.com.

For instructions on applying a product key, see the *TCP/IP FOR VSE Installation Guide*, chapter 3, "Installation."

Procedure

Use the steps below to set up a See-TCP/IP for VSE server.

1. Add the SVSESVAP phase to the SVA. This can and should also be added to the system IPL procedure. To do this, modify and run the following job in the BG partition:

```
// JOB SVSESVAP ADD SVSESVAP TO THE SDL/SVA
// LIBDEF PHASE,SEARCH=lib.sublib
SET SDL
SVSESVAP,SVA
/*
/&
```

2. Start the See-TCP/IP for VSE server, then modify and run the following job.

```
// JOB SVSESRVR
// OPTION LOG,PARTDUMP,NOSYSDMP
// OPTION SYSPARM='00' <=SysId of TCP/IP for VSE
partition to be monitored>
// LIBDEF PHASE,SEARCH=lib.sublib Installation lib.sublib
// EXEC SVSESRVR,SIZE=SVSESRVR
SETPORT 5450
SETUSER1 $SEEVSE1 $SEEVSE1
SETUSER2 $SEEVSE2 $SEEVSE2
SETUSER3 $SEEVSE3 $SEEVSE3
OPENLOG SEEPCLOG
/*
/&
```

The SVSESRVR job will stay up, and it can be run in any static or dynamic partition with a minimum of 8 MB of virtual storage. The SVSESRVR can be shutdown with a "MSG *xx*,DATA=SHUTDOWN" command from the VSE system console, where *xx* is the partition ID running the SVSESRVR job.

The commands shown in the example job above are described in the next section.

VSE Commands

The following commands can be sent to the SVSESRVR partition using the VSE attention routine "MSG *xx*,DATA=*command*," where *xx* is the partition ID running the SVSESRVR job. You can enter most commands as card input by placing them in the SYSIPT after the // EXEC SVSESRVR JCL statement.

Command List

The VSE commands are as follows.

Command	Description
SETPORT *num*	Sets the listen port number for PC clients to connect into. The value can be any number from 1 to 65000 (default is 5450). The PC client also opens a data port whose value is one less than the SETPORT value and a command connection port that is one less than the data port. The default ports used on the VSE system are 5450, 5449, and 5448. This command may be useful when configuring firewall settings. It is valid only in SYSIPT card input.
SETUSER1 *userid passwd* SETUSER2 *userid passwd* SETUSER3 *userid passwd*	Up to three user IDs and passwords can be connected into See-TCP/IP for VSE. The SETUSER*n* command creates the user ID and password that must be entered when prompted by the PC Client. The user ID and password are not case sensitive. The user ID associated with SETUSER1 is the only session allowed to issue and receive TCP/IP FOR VSE commands and messages. Once the user ID associated with SETUSER1 is logged in, records are sent from the VSE Server to the PC and are stored in its system database. The SETUSER1's PC is used for archiving and storing the performance data. The SETUSER2 and SETUSER3 user IDs are limited to real-time activity displays.
OPENLOG SEEPCLOG	Causes the SVSESRVR to open a connection for the PC to send commands and receive TCP/IP FOR VSE messages, FTP end-of-session records, connection termination records, and other logged events. The port value for this connection is two less than the SETPORT value. Only the user ID associated with the SETUSER1 command can connect into this port.

Command	Description
SETSYSID *id*	Sets the TCP/IP FOR VSE SYSID. It overrides the // OPTION SYSPARM= setting. This command is valid only in SYSIPT card input.
CONTROL SEEVSE {ATTACH | DETACH}	Opens a control connection with the TCP/IP FOR VSE partition, sends a control command, and then receives the response from the partition. The ATTACH option attaches the SVSEMNTR phase in the TCP/IP FOR VSE partition. The DETACH option detaches the SVSEMNTR phase in the partition. Both commands are issued automatically during the SVSESRVR partition's startup and shutdown. Therefore, you should issue them only when requested by CSI technical support.
SEGMENT	Causes the VSE/POWER SYSLST output for the SVSESRVR partition to be segmented.
RESETPCX	Resets all active PC connections by abruptly closing all currently active connections. A new listen is then issued after 30 seconds. Wait about 1 minute after issuing this command before attempting to reconnect from a PC client. This command can be useful when the PC is unable to establish a connection with the SVSESRVR partition.
STATUS	Displays the PC sessions currently active with the SVSESRVR partition. MSG *xx* (with no ",DATA=") defaults to issuing the STATUS command.
MONPHASE *xxxx*	Activates phase load monitoring of phase *xxxx*.
LOCPHASE {* | *xxxxxxx*}	Displays the monitored phases. Using a '*' wildcard displays all monitored phases.
ABEND000	Causes the SVSESRVR partition to abend. Issue this command only when requested by CSI technical support to diagnose a problem.
SHUTDOWN	Terminates the SVSESRVR partition.

Debugging Options

SVSESRVR also uses the UPSI JCL statement to activate debugging options. If problems occur, CSI Technical Support may request that you add one of the following statements.

Option	Description
// UPSI 1XXXXXXX	Issue a dump for abends in SVSESRVR.
// UPSI X1XXXXXX	If TCP/IP FOR VSE is not up, wait for it to come up.
// UPSI XX1XXXXX	Have SVSEMNTR issue diagnostic messages.
// UPSI XXX1XXXX	(Not used)
// UPSI XXXX1XXX	(Not used)
// UPSI XXXXX1XX	Dump VSE control blocks during initialization.
// UPSI XXXXXX1X	Activate storage monitoring for SVSESRVR.
// UPSI XXXXXXX1	Issue lots of SDUMPs for debugging.

PC Setup

The See-TCP/IP for VSE PC setup procedure verifies and installs four components on your PC. These software components are described below.

Operating System Requirements

The See-TCP/IP for VSE PC client runs on the Windows® operating system.

Hardware Requirements

The PC must meet the following minimum requirements.

Component	Requirement
Processor	Intel® Pentium® or compatible processor; 600 MHz minimum (1 GHz or more recommended)
Memory (RAM)	256 MB minimum (512 MB+ recommended)
Networking	Must be able to access the VSE system you want to monitor
Disk Space	Each system must be able to store up to 4 GB of data on disk

Software Components to Install

The installation procedure installs the following software components:

- See-TCP/IP for VSE application. This is the desktop application used to collect and display data from your VSE system(s).
- Microsoft SQL Server® software. This is a lightweight, redistributable database server that runs as a Windows service. It is used by See-TCP/IP for VSE to store and retrieve data.
- Microsoft Data Access Components 2.8 software. This software is required by SQL Server Express.
- Windows Installer 3.1 software. This software is required by SQL Server Express.
- .NET Framework 2.0 or 3.5 software. This is a component of the Microsoft Windows® operating system and is used to build and run Windows-based applications. Microsoft packages the .NET Framework with its operating systems. The software is updated by routines built into the operating system. Therefore, the .NET framework may already be installed and up to date on your PC.

  To check whether .NET Framework is installed, go to your PC's Control Panel and choose the icon to update or remove software. Check the list of applications to see whether "Microsoft .NET Framework *x.x*" is listed. The version is indicated in the title(s).

Software Installation Procedure

See-TCP/IP for VSE is designed to run on a Windows operating system.

Use the steps below to install the client software on a PC.

1. Obtain the See-TCP/IP for VSE PC setup file from CSI international. Go to this site, http://www.csi-international.com/download.htm, and choose "See-TCP/IP" under Network Management and Monitoring. Enter the requested information and accept the user agreement to initiate the ZIP file download (seetcpipsetup.zip).

2. Extract the ZIP file's contents to a folder on your PC, then read the setup instructions in the file $README_SeeTCPIP_PC.txt.

3. Run the associated "setup.exe" file on your PC to install the .Net Framework 2.0 and the Microsoft Sql server software.

   Follow the prompts to run the corresponding installer(s).

4. Run the SetupSee.msi file.

When all of the required components have been installed and verified, the application installer will start. Follow the prompts to complete the installation. Refer to the Readme file above for more information.

PC Client Operation

Main Form

See-TCP/IP for VSE starts with a main form that allows you to define each of your VSE systems. You must specify a unique name for each system, along with the IP address and port where the server is running. Once a VSE system is defined, simply click the View System button to access it.

System Form

The System Form allows you to

- Control your connections to the server
- Initiate data polling
- View monitors and charts
- Interact with the TCP/IP FOR VSE console
- Manage each system's history

The System Form contains various panes that can be moved and resized for viewing the different data tables and charts.

Polling

See-TCP/IP for VSE uses sampling, or data polling, to collect information about your VSE system. The polling controls are located on the bottom of the System Form. To collect data, the server-side component must be running on the monitored VSE system. You then can click on the Connect button. If the connection is made, the system initializes, and a "Ready to Start" message is displayed in the status bar.

You now can start a timer to sample VSE data at a selected interval. If you click on Start Autopolling, a ticking timer is displayed, and data is collected each time the interval expires. The See-TCP/IP for VSE client takes the raw performance data and stores it in a friendly format. It then computes values using the data and displays these values in useful charts and tables.

You can also click on the Poll button anytime the client is connected to sample your VSE system immediately.

Real-Time Monitoring

See-TCP/IP for VSE allows you to create monitors for various performance-related data. Typically, a performance value from VSE holds a counter for, say, the number of interrupts that have occurred. If you want to monitor this value, you can do it using an absolute monitor or a delta monitor.

Absolute Monitors

An absolute monitor plots points showing the actual value read from VSE each time a data poll is completed. If the value read from VSE is 900, an absolute monitor plots 900 for the series at that point in time.

Delta Monitors

A delta monitor plots points that show the change in value over an interval when consecutive polls are completed. If the value read from VSE has increased from 900 to 1000, a delta monitor plots 100 for the series over that time interval.

Group Monitors

Performance data are combined into groups that relate to different aspects of VSE and TCP/IP FOR VSE. These groups are as follows:

Group	Data Monitored
VSE	Activity related to overall VSE performance.
Turbo Dispatch	CPU activity and Turbo Dispatcher cycles.
Partitions	CPU Usage and SIO counts broken down by partition, job, or step. Data are only collected for partitions that are defined during a poll. For example, if a job runs in 30 seconds and no data poll is completed during that time, the job will not be reflected in the monitor.
TCP/IP	Activity related to overall TCP/IP performance including network traffic, clients, daemons, dispatcher, errors, and other data.
Foreign IPs	Activity related to inbound and outbound traffic according to protocol, such as TCP or UDP, broken down by foreign IP address.
Connections	Activity related to the send and receive side of the current TCP/IP connections, broken down by local port, foreign IP, foreign IP:port, or partition. As with partitions, data are only collected for connections that exist during a poll.

Series Data

Series data are the actual data values that are being collected and displayed on the monitor. Some series relate to overall performance and can contain only one value for each point in time, such as the number of active tasks running under VSE. Other series can contain multiple values, such as partition CPU usage where there could be many partitions running at the same time.

For the VSE, Turbo Dispatcher, and TCP/IP groups, See-TCP/IP for VSE allows you to combine multiple series on one monitor and plot each value. For CPU Activity, the values are split into five percentages that add up to 100 percent for each CPU.

For the Partitions, Foreign IPs, and Connections groups, the monitor stacks the series values grouped according to a common value (that is, partition, job, or step). It also limits the number of values displayed to save space (the 10 partitions with the highest percentage of CPU usage). These settings can be changed using the controls to the right of the monitor.

CPU Percentages

CPU percentages are monitored in one of two ways:

- **Absolute.** Values correspond to the percentage of CPU time used since the last CPU reset.
- **Delta.** Values correspond to the percentage of CPU time used over a time interval.

History

Data are stored for each VSE system whenever a data poll is completed. If you are using the TCP/IP FOR VSE console, records are stored each time a message is received, a job step ends, a connection terminates, an FTP session ends, or the See-TCP/IP for VSE server is started. All of these data are kept on the PC and can be used to create and display historical charts and tables.

Managing Records

Each system stores its records in an SQL database. By default, the history can store up to 4 GB worth of records for each system. You can use the Manage History controls to remove unneeded records and times from your system history. A pie chart displays the amount of space being used by your system history and how much space is still available in the database.

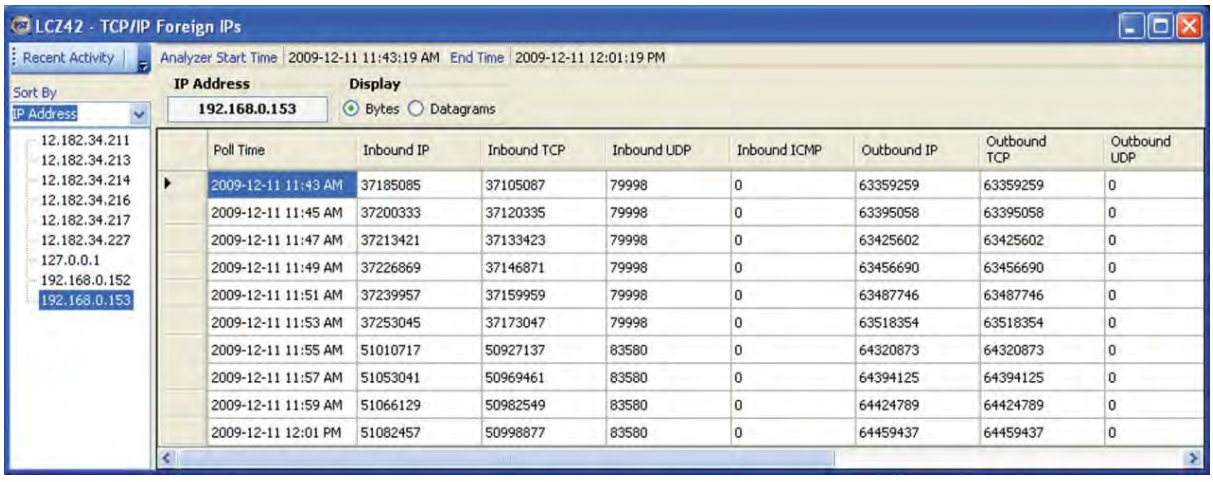
The first type of records are those collected during polls. They can be completely cleared out or removed individually according to time. The other record types are those collected by the TCP/IP FOR VSE console. Once records are removed, they can no longer be displayed in historical charts or tables.

Creating Charts

Historical charts are created in the same way as real-time monitors. Refer to "Real-Time Monitoring" above for descriptions of the available chart types, groups, and series. Of course, for historical charts, you are able to choose which dates and times you want to include.

You can right-click on any chart and select Data View to see the underlying data in tabular format. Also, you can use the toolbar on any chart to manipulate its properties and save it to disk for later use.

See-TCP/IP can create reports that show IP connections into z/VSE.



Poll Time	Inbound IP	Inbound TCP	Inbound UDP	Inbound ICMP	Outbound IP	Outbound TCP	Outbound UDP
2009-12-11 11:43 AM	37185085	37105087	79998	0	63359259	63359259	0
2009-12-11 11:45 AM	37200333	37120335	79998	0	63395058	63395058	0
2009-12-11 11:47 AM	37213421	37133423	79998	0	63425602	63425602	0
2009-12-11 11:49 AM	37226869	37146871	79998	0	63456690	63456690	0
2009-12-11 11:51 AM	37239957	37159959	79998	0	63487746	63487746	0
2009-12-11 11:53 AM	37253045	37173047	79998	0	63518354	63518354	0
2009-12-11 11:55 AM	51010717	50927137	83580	0	64320873	64320873	0
2009-12-11 11:57 AM	51053041	50969461	83580	0	64394125	64394125	0
2009-12-11 11:59 AM	51066129	50982549	83580	0	64424789	64424789	0
2009-12-11 12:01 PM	51082457	50998877	83580	0	64459437	64459437	0

TCP/IP FOR VSE Console

You use the console to issue TCP/IP FOR VSE commands and view TCP/IP FOR VSE messages. There is a command line where commands can be entered manually or with interactive help. Additionally, the console stores Messages, Job Step termination records, Connection termination records, FTP end-of-session records, and Startup records into the system history. Summaries of these records can be seen in the console logs, or complete records can be loaded into tables.

Interactive Lookup

Two controls, described below, are available on the right side of the console to look up commands and messages. They become visible when you click on them. To keep them visible, click on the thumbtack icon in their upper-right corner. A history of messages and commands you have looked up is kept in the combo-boxes:

- TCP/IP FOR VSE Commands

  Presents parameter options for available commands. Type a command name into the combo-box at the top of the help and press <Enter>.

  Shortcut: Press <F1> inside the command line and select a command name from the list presented.

- TCP/IP FOR VSE Message Codes

  Displays message descriptions and suggested actions. Type a message name into the combo-box at the top of the help and press <Enter>.

  Shortcut: Double-click a message code in the console, then right-click and select “Lookup Message.”

Console Tables

The following tables correspond to records collected whenever the TCP/IP FOR VSE console is active.

Table	Data Displayed
Job Steps	Accumulated performance data for each job step when it ends.
Connections	Accumulated performance data for each connection when it terminates.
FTP Sessions	Accumulated performance data for each FTP session when it ends.
Startups	A list of times when the See-TCP/IP for VSE server was started on VSE.

You can right click on columns that you are not interested in to hide them.

5

Firewall Shield

Overview

The Firewall Shield prevents unauthorized internet access to z/VSE systems that host the TCP/IP FOR VSE product.

Most VSE installations have a limited number of known IP addresses that are allowed to access the VSE system. Therefore, the Firewall Shield is a “white list” based firewall implementation. This approach can also be thought of as security by default, which means that access is denied unless an IP address is specifically allowed to communicate with the VSE system.

The Firewall Shield controls access in three ways:

- First, it limits access to only IP addresses within an allowed range.
- Second, for IP addresses that are allowed access, it can block ping (ICMP) attempts so that connections remain invisible to the internet.
- Third, it can control which TCP or UDP ports can be accessed by allowed addresses.

The Firewall Shield feature is included with the TCP/IP for VSE product and does not require a separate download.

Activation

The Firewall Shield is loaded early during TCP/IP startup to avoid any possibility of a remote system bypassing the shield before the firewall is initialized. The mechanism to activate the firewall is the parameter FIREWARN= or FIREWALL= that identifies the custom configuration phase to be used. The TCP/IP startup would then be similar to one of the following statements.

Activation in WARN mode:

```
// EXEC IPNET,SIZE=IPNET,PARM='ID=00,INIT=IPINIT00,FIREWARN=FIREWALL'
```

Activation in FAIL mode:

```
// EXEC IPNET,SIZE=IPNET,PARM='ID=00,INIT=IPINIT00,FIREWALL=FIREWALL'
```

With either activation statement, the configurable firewall phase will be loaded that contains the list of allowed IP addresses. See the section "Firewall Configuration," page 84, for details on the job that creates this phase. The default name of the configuration phase is FIREWALL. The same phase can be shared between stacks, or unique phase names can be created and used for allowing different settings when running with multiple stacks. Each range of IP addresses can also specify the allowed VSE ports (TCP or UDP) and whether the ICMP (Ping) is allowed from the remote IP address range. If this phase is successfully loaded, the Firewall Shield will be activated in the specified mode.

For FAIL mode, this means that any datagrams received from any remote IP address that is not defined in the firewall configuration phase will be discarded and a violation message will be issued.

For WARN mode, this means that the same violation messages will occur, but the associated datagrams will not be discarded. This mode is useful for initial setup of the firewall to identify the port numbers on VSE that remote IP addresses are attempting to access. It is strongly recommended to initially use and test with the WARN mode.

Commands and Messages

FIREWALL Command

The FIREWALL command is used to control and monitor the Firewall Shield. The syntax is

```
FIREWALL {ON|OFF|LOAD [PHASE=phase-name]|WARN|FAIL|MSGON|
          MSGOFF|DEBUGON|DEBUGOFF|REPORT|ALLOWED|BLOCKED}
```

where

ON

Is the default setting if the FIREWALL= parameter is used in the EXEC IPNET statement and a firewall configuration phase is successfully loaded during TCP/IP startup. If the firewall is not initially on, or it is turned off with the FIREWALL OFF command, FIREWALL ON can be issued to activate or reactivate the firewall.

OFF

Turns off the Firewall Shield feature.

LOAD PHASE=*phase-name*

Activates a new firewall configuration. This means a new firewall configuration phase will be loaded and activated. The default phase name is FIREWALL, but the PHASE= keyword can be used to load a different configuration phase.

WARN

Sets the firewall to warn mode. Firewall violations are displayed and logged, but an attempted access is allowed when this mode is active.

When in warn mode, the initial check for an IP address that does not match a range is allowed to pass to the next layer. The TCP ports, UDP ports, and ICMP are then also checked, and those checks may display additional blocked messages at the deeper layers for the corresponding protocol.

FAIL

Sets the firewall to fail mode. Firewall violations are displayed and logged, and each associated datagram is discarded immediately.

MSGON

Displays all blocked attempts on the console and SYSLST.

MSGOFF

Displays the first occurrence of blocked attempts on the console and SYSLST, but subsequent blocks are not displayed. A counter is maintained, and FIREWALL REPORT can be used to see the number of blocks for any specific IP address and port.

DEBUGON
Turns on debugging mode, and diagnostic dumps are created during firewall processing.

Caution:

FIREWALL DEBUGON can quickly cause a lot of dumps to SYSLST in the TCP/IP partition. Use this setting only at the direction of Technical Support.

DEBUGOFF
Turns off debugging mode, and diagnostic dumps will not occur.

REPORT
Displays allowed IP addresses and IP addresses that were allowed but then blocked by TCPPORTS, UDPPORTS, or ICMP.

ALLOWED
Displays a list of allowed IP addresses and the number of times each was allowed.

BLOCKED
Displays a list of IP addresses that were blocked because they are not in the firewall table.

QUERY FIREWALL Command

The QUERY FIREWALL command displays the current firewall settings being enforced.

See also IPSTAT and QUERY IPSTAT commands in the *TCP/IP FOR VSE Command Reference*.

Messages

The messages related to the Firewall Shield are documented in the *TCP/IP FOR VSE Messages* manual. These messages provide information and statistics about the firewall's activity.

All messages are issued to SYSLST in the TCP/IP partition, and you can control which messages are displayed on the VSE system console based on the level (critical, important, warning, security, informational, response) indicated for each individual message.

See the MODIFY LOG command in the *TCP/IP FOR VSE Command Reference* for information on controlling which message levels are displayed on the VSE system console and/or logged to a sequential disk log file.

IP Address Blocking

Address Ranges

The firewall configuration contains the IP addresses that are allowed to access the VSE system. This is accomplished by specifying ranges of remote IP addresses. A range can just be a single remote IP address by simply using the same beginning and ending address.

Multiple ranges can be specified, but no sorting is performed. The first matching range, searching from top to bottom, is used from the loaded firewall configuration phase.

IP Address 127.0.0.1

IP address 127.0.0.1 is referred to as the *local loopback address*, and it is often used by applications running under the control of the same TCP/IP stack. Traffic for this type of connection never leaves the local VSE system, but is still part of the defined connections within the VSE system. The default and recommended firewall definitions are set to always allow this type of connection. But the 127.0.0.1 loopback can also be controlled or monitored in warn mode to temporarily see these connections. An example of this is an FTPBATCH LOPEN establishing a connection with its locally attached FTP daemon subtask.

TCP and UDP Port Blocking

Overview

The IP address-range check is performed first, and if the remote IP address is allowed, then optionally a group of allowed ports can be associated with the associated IP address range. The port-group keyword ALL can be specified to allow the using any ports on the VSE system for the associated IP address range. To use TCP and UDP port blocking, the allowed IP address range can be configured with a group of specific allowed ports. A TCP connection consists of two sets of components:

- Local IP address and Port

  This is the VSE's local IP address and port number.

- Remote IP address and Port

  This is the remote IP address and port number.

The firewall's TCP port blocking examines only the local port being used in a TCP connection. Note that a connection established between partitions on the same VSE system still has local and remote port numbers.

TCP and UDP Port Blocking for Clients on VSE

When a VSE application issues a connect (an active open) or a listen (a passive open), it can specify a specific local port number to be used, or it can specify zero for the port number. If zero is used, a free port is dynamically assigned for the local port number from the available free port range on the VSE system. The default local VSE free port range is 4096–65535. The default free port range can be overridden with the PORTRANGE command, as documented in the *TCP/IP FOR VSE Command Reference* manual.

The firewall will block remote servers that attempt to use a local VSE port number that is not in the port group associated with the remote IP address. The exception to this is if the application is using a free port in the currently defined free port range. In that case, the connection will be allowed or disallowed based upon the FREEPBLK=NO/YES setting. It may be difficult to identify which VSE applications are using hard-coded local port numbers as opposed to dynamic free ports. The firewall's warn mode should be used to detect the ports being used by VSE client applications

TCP Port Blocking for Servers on VSE

When a VSE server application is run, it specifies a known port number that remote clients can use to connect into the VSE system. The firewall will block remote clients that attempt to connect to a local VSE server listening on a port that is not in the port group associated with the remote IP address.

Note that because UDP is a stateless connection, this statement does not apply to UDP port blocking.

FTP Considerations

The FTP data connection uses an independent dynamic connection, and it can be difficult to configure when the firewall is active. When a data connection is established, one side enters a passive listen state and sends a message to the other side informing it of the port number to be used.

One side is referred to as being in a *passive mode*, and the other is referred to as being in *active mode*. A dynamic free port is obtained when in passive mode, and it is therefore almost impossible to configure the allowed port numbers for the remote system. One alternative is to have the VSE side be in active mode so it will not have to enter a passive listen state, but that means that the remote system must enter the passive listen state and its firewall must allow the VSE side of the connection to connect into its passive listen port.

Firewall Configuration

Sample Configuration Job

The firewall's configuration is set by an assembly/link-edit job similar to the sample job below. This job creates the FIREWALL.PHASE.

```
* $$ JOB JNM=FIREWALL,CLASS=0,DISP=D
* $$ LST CLASS=A
// JOB FIREWALL Assemble and Catalog Firewall Settings
// OPTION CATAL
// LIBDEF *,CATALOG=LIBRARY.SUBLIB
// EXEC ASMA90,SIZE=ASMA90,PARM='SZ(MAX-200K,ABOVE)'
    PUNCH    ' PHASE FIREWALL,* '
*
* * The below are sample settings for creating the FIREWALL.PHASE
    FIREWALL  BEGIN,PHASE=FIREWALL
*
    FIREWALL  ALLOW,IPV4BEG=127.000.000.001,IPV4END=127.000.000.001,   X
               TCPPORTS=ALL,UDPPORTS=ALL,ICMP=YES,FREEPBLK=NO
*
    FIREWALL  ALLOW,IPV4BEG=039.101.062.131,IPV4END=039.101.062.131,   X
               TCPPORTS=PORTGRPA,UDPPORTS=NONE,ICMP=YES,FREEPBLK=YES
*
    FIREWALL  ALLOW,IPV4BEG=039.101.062.200,IPV4END=039.101.062.255,   X
               TCPPORTS=PORTGRPB,UDPPORTS=NONE,ICMP=NO
*
    FIREWALL  ALLOW,IPV4BEG=235.100.101.200,IPV4END=235.100.101.232,   X
               TCPPORTS=PORTGRPC,UDPPORTS=PUDPGRPA,ICMP=NO
*
    FIREWALL  END
*
* * Port Group definitions begin here...
PORTBEGN DS    0D                          Beginning of port groups
*
         DS    0D
PORTGRPA DC    F'21',F'20',F'3349'          Allowed ports
         DC    X'FFFFFFFF'                 End of group indicator
*
         DS    0D                          Beginning of new group
PORTGRPB DC    F'80',F'443',F'222'          Allowed ports
         DC    F'81',F'572',F'998'          Allowed ports
         DC    X'FFFFFFFF'                 End of group indicator
*
         DS    0D                          Beginning of new group
PORTGRPC DC    F'32767',F'32768'            Allowed ports
         DC    X'FFFFFFFF'                 End of group indicator
*
         DS    0D                          Beginning of new group
PUDPGRPA DC    F'1200',F'1201',F'1202'      Allowed ports
         DC    X'FFFFFFFF'                 End of group indicator
*
PORTENDX DS    0H                          Ending of port groups
*
      END
        /*
// EXEC LNKEDT,SIZE=512K
/*
/&
* $$ EOJ
```

FIREWALL Macro Operands

The configuration job uses the following FIREWALL macro operands.

- The ALLOW operand sets the beginning and ending IP address ranges to be allowed access to the system.
- The use of ICMP protocol can be restricted with the ICMP=YES/NO. ICMP is the protocol used by network diagnostic utilities such as PING and TRACERT. Unfortunately, these utilities are also often used by malware to detect IP addresses for hacking. Specify ICMP=NO to block a range of IP addresses from pinging the VSE system. The default is YES.
- The TCP and UDP ports that the range of IP addresses can access on the VSE system can also be controlled by pointing the TCPPORTS= and/or UDPPORTS= to a group of port numbers.

  Using the ALL option for ports allows all ports of the associated protocol to be accessed on the VSE system.

  The keyword value NONE can be specified to deny all ports of either the TCP or UDP protocol.
- The use of dynamic free ports can be controlled with the FREEPBLK=YES/NO option. The default is NO, or the use of dynamic free ports is blocked.

About the ACCESS and TRUST Commands

The current ACCESS and TRUST commands that are documented in the *TCP/IP FOR VSE Command Reference* manual can also be used to control access to the VSE system. But these commands do not support a table of pre-existing IP addresses, and they require some activity for the associated IP address before the commands can be issued.

To avoid conflicts with current sites that use these commands, the firewall's enforcement is independent of these commands. The firewall has no operational effect on these commands. The use of the ACCESS and PREVENT commands in addition to the firewall is allowed, but it is discouraged because it creates a more complex layer that is not necessary.