IPv6/VSE IPv6

Installation Guide

Build 247 and Higher

Table of Contents

Preface

About this Publication

This is the **IPv6/VSE Installation Guide**. The manual will introduce you to IPv6/VSE and provide you with the information necessary to install and startup the IPv6/VSE IPv6 TCP/IP stack.

Trademarks

The following are lists of the trademark and products referenced in this manual. Symbols for trademarks and registered trademarks do not appear in subsequent references.

Barnard Software, Inc.

TCP/IP-TOOLS is a registered trademark of Barnard Software, Inc.
IPv6/VSE is a registered trademark of Barnard Software, Inc.

International Business Machines Corporation

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

Copyrights

zlib-1.2.5 Compression Library

Technical Support

IBM Customers

IBM IPv6/VSE customers should contact IBM for support.

BSI Customers

Technical Support is available from Barnard Software, Inc. by phone, mail or email:

Barnard Software, Inc.
806 Silk Oak Terrace
Lake Mary, FL 32746

Phone: 1-407-323-4773

Support: bsiopti@bsiopti.com

Sales: bsisales@bsiopti.com

Support is available from 9:00 a.m. through 5:00 p.m. EST, Monday through Friday.

If a TSR (Technical Support Representative) is not available at the time of your call, please leave a message and a TSR will return your call as soon as possible. Please provide the following information: name, company, phone number, product name, product release level, and a short description of the problem.

BSIUsers Announcement List Server

When new releases of IPv6/VSE are available BSI will post an announcement on its BSIUsers announcement list.

To subscribe to the BSIUsers announcement list send an email to this email address

BSIUsers-subscribe@yahoogroups.com

To unsubscribe to the BSIUsers announcement list send an email to this email address

BSIUsers-unsubscribe@yahoogroups.com

Problem Determination

If you have a problem using a IPv6/VSE application always check the SYSLST output for additional information and messages. Most messages are written to SYSLST and not to the VSE/ESA system console.

When contacting BSI for technical support always have the applications JCL/commands, console and SYSLST output available for problem determination. The SYSLST output is very important.

While a IPv6/VSE application is running, you can issue the **AR CANCEL XX,PARTDUMP** command to terminate IPv6/VSE application and dump the partition to SYSLST. Using the VSE/POWER Flush (F) command cancels the IPv6/VSE application partition without a dump.

If the IPv6/VSE application partition stops responding to its console interface, use the **AR DUMP XX** command to obtain a dump of the partition.

Chapter 1

IPv6/VSE Features

Internet Protocol version 6 (IPv6) is the next-generation Internet Protocol version. IPv6 has been designated as the successor to IPv4. IPv4 was the first implementation used in the Internet that is still in main protocol in use today. The primary reason for the redesign of IP was the foreseeable IPv4 address exhaustion. IPv6 was defined in December 1998 by the Internet Engineering Task Force (IETF) with the publication of an Internet standard specification, RFC 2460.

IPv6 has a dramatically larger address space than IPv4. This results from the use of a 128-bit address, whereas IPv4 uses only 32 bits. The new address space thus supports about 3.4×1038 addresses. This expansion provides flexibility in allocating addresses and routing traffic and eliminates the primary need for network address translation (NAT), which gained widespread deployment as an effort to alleviate IPv4 address exhaustion.

IPv6 also implements new features that simplify aspects of address assignment (stateless address auto-configuration) and network renumbering (prefix and router announcements) when changing Internet connectivity providers. The IPv6 subnet size has been standardized by fixing the size of the host identifier portion of an address to 64 bits to facilitate an automatic mechanism for forming the host identifier from Link Layer media addressing information (MAC address).

The first publicly used version of the Internet Protocol, Version 4 (IPv4), provides an addressing capability of about 4 billion addresses. This was deemed sufficient in the early design stages of the Internet when the explosive growth and worldwide expansion of networks was not anticipated.

By the beginning of 1992, several proposed systems were being circulated, and by the end of 1992, the IETF announced a call for white papers (RFC 1550) and the creation of the "IP Next Generation" (IPng) area of working groups. The Internet Engineering Task Force adopted IPng on July 25, 1994, with the formation of several IPng working groups. By 1996, a series of RFCs were released defining Internet Protocol Version 6 (IPv6), starting with RFC 2460.

Incidentally, the IPng architects could not use version number 5 as a successor to IPv4, because it had been assigned to an experimental flow-oriented streaming protocol (Internet Stream Protocol), similar to IPv4, intended to support video and audio.

It is widely expected that IPv4 will be supported alongside IPv6 for the foreseeable future. IPv4-only nodes are not able to communicate directly with IPv6 nodes, and will need assistance from an intermediary.

Estimates of the time frame until complete exhaustion of IPv4 addresses used to vary widely. As of May 2009, a daily updated report projected that the IANA pool of unallocated addresses would be exhausted in June 2011, with the various Regional Internet Registries using up their allocations from IANA in March 2012. There is now consensus among Regional Internet Registries that final milestones of the exhaustion process will be passed in 2010 or 2011 at the latest, and a policy process has started for the end-game and post-exhaustion era.

In most regards, IPv6 is a conservative extension of IPv4. Most transport and application-layer protocols need little or no change to operate over IPv6; exceptions are application protocols that embed network-layer addresses, such as FTP or NTPv3.

IPv6 specifies a new packet format, designed to minimize packet-header processing. Since the headers of IPv4 packets and IPv6 packets are significantly different, the two protocols are not compatible.

The most important feature of IPv6 is a much larger address space than that of IPv4: addresses in IPv6 are 128 bits long, compared to 32-bit addresses in IPv4. The very large IPv6 address space supports a total of about 3.4×1038 addresses—or about 4.5×1015 addresses for every observable star in the known universe.

While these numbers are impressive, it was not the intent of the designersof the IPv6 address space to assure geographical saturation with usable addresses. Rather, the longer addresses allow a better, systematic, hierarchical allocation of addresses and efficient route aggregation. With IPv4, complex Classless Inter-Domain Routing (CIDR) techniques were developed to make the best use of the small address space. Renumbering an existing network for a new connectivity provider with different routing prefixes is a major effort with IPv4, as discussed in RFC 2071 and RFC 2072. With IPv6, however, changing the prefix announced by a few routers can in principle renumber an entire network since the host identifiers (the least-significant 64 bits of an address) can be independently self-configured by a host.

IPv6 hosts can configure themselves automatically when connected to a routed IPv6 network using ICMPv6 router discovery messages. When first connected to a network, a host sends a link-local multicast router solicitation request for its configuration parameters; if configured suitably, routers respond to such a request with a router advertisement packet that contains network-layer configuration parameters.[12]

If IPv6 stateless address auto-configuration is unsuitable for an application, a network may use stateful configuration with the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) or hosts may be configured statically.

Multicast, the ability to send a single packet to multiple destinations, is part of the base specification in IPv6. This is unlike IPv4, where it is optional (although usually implemented).

IPv6 does not implement broadcast, which is the ability to send a packet to all hosts on the attached link. The same effect can be achieved by sending a packet to the link-local all hosts multicast group. It therefore lacks the notion of a broadcast address—the highest address in a subnet (the broadcast address for that subnet in IPv4) is considered a normal address in IPv6.

Internet Protocol Security (IPsec), the protocol for IP encryption and authentication, forms an integral part of the base protocol suite in IPv6. IPsec, however, is not widely used at present except for securing traffic between IPv6 Border Gateway Protocol routers.

IPv4 limits packets to 64 KB of payload. IPv6 has optional support for packets over this limit, referred to as jumbograms, which can be as large as 4 GB. The use of jumbograms may improve performance over high-MTU networks. The use of jumbograms is indicated by the Jumbo Payload Option header.

Fragmentation is handled only in the sending host in IPv6: routers never fragment a packet, and hosts are expected to use Path MTU discovery.

IPv6 addresses are normally written with hexadecimal digits and colon separators like 2001:db8:85a3::8a2e:370:7334, as opposed to the dot-decimal notation of the 32 bit IPv4 addresses. IPv6 addresses are typically composed of two logical parts: a 64-bit (sub-)network prefix, and a 64-bit host part.

IPv6 addresses are classified into three types: unicast addresses which uniquely identify network interfaces, anycast addresses which identify a group of interfaces—mostly at different locations—for which traffic flows to the nearest one, and multicast addresses which are used to deliver one packet to many interfaces. Broadcast addresses are not used in IPv6. Each IPv6 address also has a 'scope', which specifies in which part of the network it is valid and unique. Some addresses have node scope or link

scope; most addresses have global scope (i.e. they are unique globally).

Some IPv6 addresses are used for special purposes, like the loopback address. Also, some address ranges are considered special, like link-local addresses (for use in the local network only) and solicited-node multicast addresses (used in the Neighbor Discovery Protocol).

A quad-A record (AAAA) is defined in the DNS for returning IPv6 addresses to forward queries; a new format of PTR record is also defined for reverse queries.

Until IPv6 completely supplants IPv4, a number of transition mechanisms are needed to enable IPv6-only hosts to reach IPv4 services and to allow isolated IPv6 hosts and networks to reach the IPv6 Internet over the IPv4 infrastructure.

A fundamental IPv4-to-IPv6 transition technology involves the presence of two Internet Protocol software implementations in an operating system, one for IPv4 and another for IPv6. Such dual-stack IP hosts may run IPv4 and IPv6 completely independently, or they may use a hybrid implementation.

Dual-stack IPv6/IPv4 implementations typically support a special class of addresses, the IPv4-mapped addresses. This address type has its first 80 bits set to zero and the next 16 set to one while its last 32 bits are filled with the IPv4 address. These addresses are commonly represented in the standard IPv6 format, but having the last 32 bits written in the customary dot-decimal notation of IPv4; for example, ::ffff:192.0.2.128 is the IPv4-mapped IPv6 address for IPv4 address 192.0.2.128.

In order to reach the IPv6 Internet, an isolated host or network must use the existing IPv4 infrastructure to carry IPv6 packets. This is done using a technique known as tunneling which consists of encapsulating IPv6 packets within IPv4, in effect using IPv4 as a link layer for IPv6.

The direct encapsulation of IPv6 datagrams within IPv4 packets is indicated by IP protocol number 41. IPv6 can also be encapsulated within UDP packets e.g. in order to cross a router or NAT device that blocks protocol 41 traffic. Other encapsulation schemes, such as used in AYIYA or GRE, are also popular.

Barnard Software, Inc.'s IPv6/VSE product is the first IPv6 TCP/IP product for the z/VSE customer. BSI's IPv6/VSE provides an IPv6 TCP/IP stack, IPv6 API routines and a full set of IPv6 enabled applications for the z/VSE user.

IPv6/VSE provides a new IPv6 TCP/IP stack. The IPv6 stack is run in a separate partition using a different stack ID. This allows you to run dual TCP/IP stacks on your z/VSE system. Running separate dual IPv4 and IPv6 stack partitions provides the best possible performance and reliability. Existing applications continue to run unchanged. New IPv6 enabled applications can gradually be introduced that use the new IPv6 TCP/IP stack.

Applications using the BSI IPv6/VSE stack run outside the IPv6/VSE stack partition. The BSI IPv6/VSE applications do not run in the TCP/IP stack partition. Running applications external to the IPv6/VSE TCP/IP stack partition provides greater stability and better performance.

This does mean that a few additional partitions will be used. For example, a TN3270E server partition, FTP Server partition, etc. will be used in addition to the IPv6/VSE TCP/IP stack partition.

IPv6/VSE supports 6in4 Static Tunneling ...

IPv6/VSE includes 6in4 tunneling. Want to test IPv6? Think you will have ISP or hardware issues? No problem! IPv6/VSE's 6in4 static tunnel driver allows you to pass IPv6 packets to a IPv6 tunnel broker over

an IPv4 connection. Best of all, most IPv6 tunnel brokers are free! Now you can test and develop IPv6 in house using your existing IPv4 infrastructure.

6in4 is an Internet transition mechanism for migrating from Internet Protocol version 4 (IPv4) to IPv6. 6in4 uses tunneling to encapsulate IPv6 traffic over explicitly-configured IPv4 links. The 6in4 traffic is sent over the IPv4 Internet inside IPv4 packets whose IP headers have the IP protocol number set to 41. This protocol number is specifically designated as IPv6-in-IPv4. In 6in4, the IPv4 packet header is immediately followed by the IPv6 packet being carried. This means that the encapsulation overhead is simply the size of the IPv4 header of 20 bytes. With an Ethernet Maximum Transmission Unit (MTU) of 1500 bytes, one can thus send IPv6 packets of 1480 bytes without fragmentation. 6in4 tunneling is also referred to as proto-41 static because the endpoints are configured statically.

BSI IPv6 Enabled TCP/IP Applications

FTP server. The FTP server supports access to z/VSE resources (VSE/POWER queues, VSAM catalogs, SAM file, z/VSE libraries, etc.) by remote host FTP clients.

Batch FTP client, The batch FTP client runs as a z/VSE batch job providing access to remote host FTP servers. Data can be sent to or received from these remote FTP servers.

TN3270E server. The IPv6/VSE TN3270E server supports TN3270/TN3270E terminal sessions and TN3270E printer sessions. In addition, DIRECT, LPR and FTP printer sessions are supported.

NTP server. The NTP server is a Network Time Protocol server that allows remote hosts (PC's, servers, etc) to ask z/VSE the time of day. This allows remote hosts to 'sync' their clocks to z/VSE's clock.

NTP client. The NTP client allow z/VSE to set its TOD clock to an external source. Typically this is done only during time changes.

System Logger client. This application is used to log selected z/VSE console message to a remote Linux syslog-ng daemon. Once sent to the syslog-ng daemon Linux automation processing can be used to trigger events.

Batch Email client. The batch Email Client is used to send an email to an SMTP server. In turn, the SMTP server will send the email to a destination user. Any number of recipients are permitted and files can be attached to an outgoing email message.

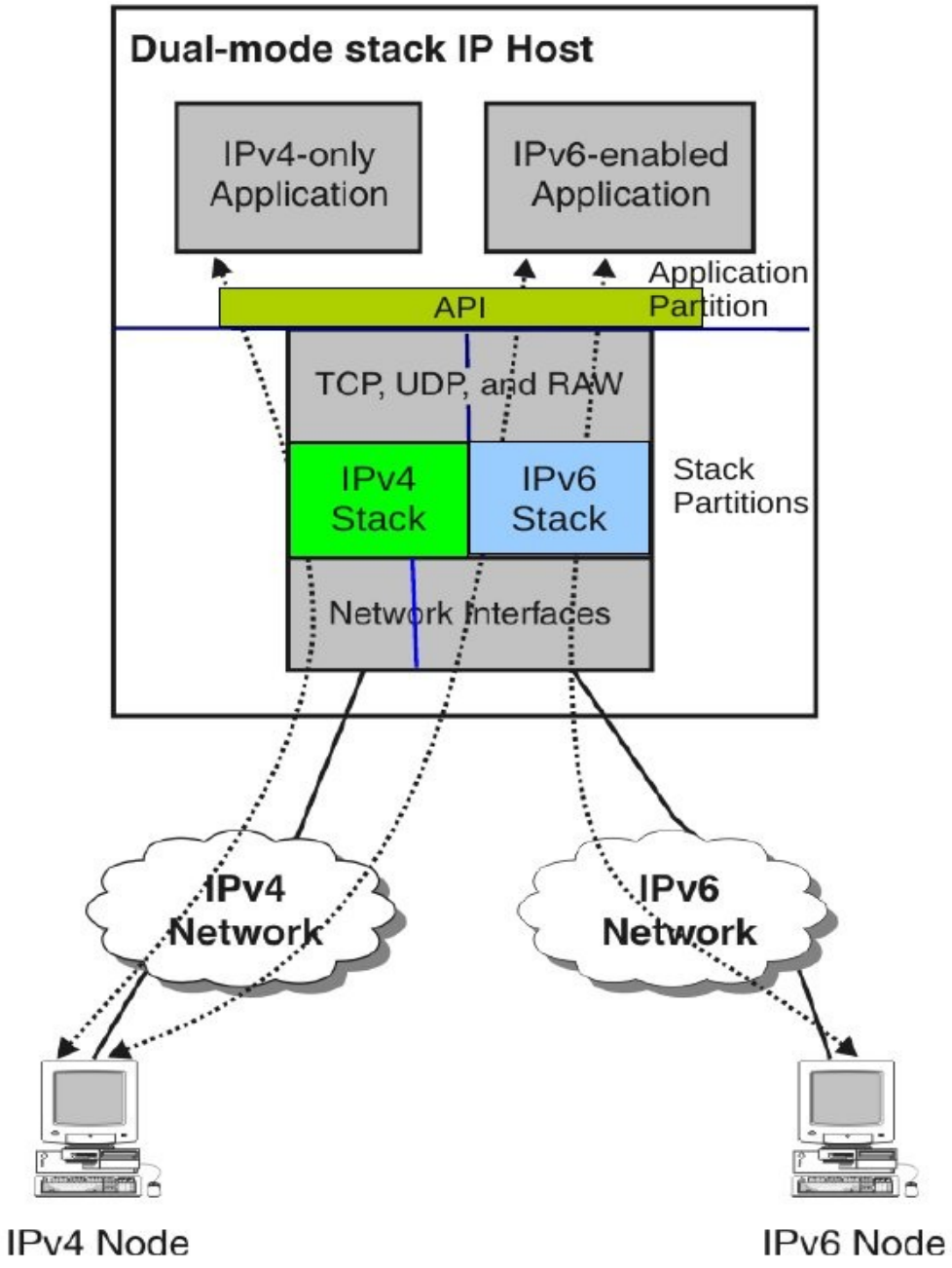
Batch LPR. The batch LPR application extracts data from the VSE/POWER queues and transfers it to a remote host LPD. The LPD can be in a printer or running as a server on a remote host.

Batch Remote Execution Client. The Remote EXEC Client allow a job running in a z/VSE partition to trigger a script to run on a remote host. Output from the script is returned to the client and scanned for completion information. The z/VSE return code is set based on the completion of the script.

Batch PING. The batch PING application is used to ping a remote host. The z/VSE return code is set based on the PING results.

GZIP data compression. IPv6/VSE provides a simple gzip data compression application. Data can be read, compressed and written to a SAM file or library member. The compressed data can then be transferred to a remote host for processing. The reverse of this process can also be done.

REXX automation. IPv6/VSE uses z/VSE REXX EXEC's for automation. Automatic FTP of data is handled using a provided sample REXX EXEC (BSTTAFTP.PROC) for example. Automatic LPR or automatic Email of data is handled in the same way. Invoking IPv6/VSE application from within a REXX EXEC allows dynamic creation of command and parameters (E.g., filenames and dates).



The above diagram shows the basic layout of a dual stack system. When an API call is made the stack ID used determine the stack to be used. Based on the type of stack either the IPv4 or the IPv6 API routines are entered.

The choice to run separate IPv4 and IPv6 TCP/IP stack partitions was made for performance, reliability and robustness.

Chapter 2

IPv6/VSE Installation

IPv6/VSE is a product made up of phases and various library members.

VSE/ESA and z/VSE Requirements

Any version and release of VSE/ESA or z/VSE running on a supported processor using a supported network interface will work.

Processor Requirements

The IPv6/VSE BSTT6NET TCP/IP stack requires an ALS 2 processor. MP-3000, 9672 G5/G6, P/390, IS-3006, FLEX-ES and all IBM zSeries processors are examples of the required processors. Older processors including 9121, 9221 and MP-2003 processors are not supported by IPv6/VSE.

Any version and release of VSE/ESA and z/VSE running on a supported processor is supported by BSI. IBM IPv6/VSE is only supported on IBM supported releases of z/VSE. Currently this is z/VSE 4.2 or higher.

Network Interface Requirements

IPv6/VSE supports CTCA, 6in4 tunnels, OSA Express and Hipersocket network interfaces.

CTCA and 6in4 tunneling network interfaces may be used with any supported version of VSE/ESA or z/VSE.

OSA Express and Hipersocket support requires z/VSE 4.2 (or higher). Only OSA Express adapters running in QDIO mode or Hipersocket network interfaces are supported. The IBM z/VSE IJBOSA driver routine must be at APAR level DY47077 or higher.

Installation of the BSI INSTTOOL.JOB File

The IPv6/VSE download is a .zip file. When the .zip file is unzipped it will contain a file called INSTTOOL.JOB. The INSTTOOL.JOB file is a BINARY file with 80 bytes fixed length records. This job will catalog the IPv6/VSE phases and library members.

Using TCP/IP

FTP the INSTTOOL.JOB file to the VSE/POWER RDR queue using a BINARY transfer. If you are using the CSI/IBM TCP/IP for VSE product will will want to rename the INSTTOOL.JOB file to INSTTOOL.BJB prior to any FTP.

Use the VSE/POWER R RDR,INSTTOOL command to release the INSTTOOL job for execution. The INSTTOOL job will pause to allow you to set various SETPARM statements.

Using IND$FILE

Transfer the INSTTOOL.JOB file to the VSE/POWER RDR queue. Specify BINARY and NOUC (NO UpperCase) for the transfer.

Installation of IBM IPv6/VSE

The IPv6/VSE product is distributed in the Librarian format of VSE/Advanced Functions Version 2.
To install IPv6/VSE you can use the z/VSE dialog 'Install Programs – V2 Format'.

To access the dialog, start with the Function Selection panel and select each of the two tasks
in the following sequence:

Task 1:
- **1** (Installation)
- **1** (Install Programs – V2 Format)
- **1** (Prepare for Installation)

Task 2:
- **1** (Installation)
- **1** (Install Programs – V2 Format)
- **2** (Install Program(s) from Tape)

1. Prepare for Installation

   The dialog scans the IPv6/VSE product tape, builds a list of the found programs and gathers program statistics. The list is displayed when you use the dialog 'Install Program from Tape'. When the job completes, review the scan report before you install the product.

2. Install Program from Tape

   The FULIST that was created during the 'Prepare for Installation' task, is displayed, listing all programs on the tape.

   The list in the example below shows the IPv6/VSE program with its default library PRD2.PROD. Enter '**1**' in the option field to install IPv6/VSE into the default library. The created job streams use the job manager to manage the installation.

```
INS$OPI1        INSTALL ADDITIONAL PROGRAM(S) FROM TAPE

              LIST OF PROGRAMS TO BE INSTALLED

             OPTIONS:  1 = INSTALL    2 = SKIP INSTALLATION

                                          LIBRARY   SUBLIBR.
               OPT      IDENTIFIER        NAME      NAME      SEQ.NO.  TAPE
NO.

                1       IPV6/VSE....1.1.0 PRD2      PROD        1        1
                _                         ________  ________
                _                         ________  ________
                _                         ________  ________
                _                         ________  ________
                _                         ________  ________
                _                         ________  ________
                _                         ________  ________

          PF1=HELP      2=REDISPLAY  3=END                    5=PROCESS

          ALL SCANNED PRODUCTS NEED A MINIMUM OF 12296 LIBRARY
```

For further installation instructions refer to the chapter that describes 'Installing additional VSE Programs' in the z/VSE Installation manual'.

License Parameters

The BSTTPARM.A member of the installation library contains the COMPANY and LICENSE parameters provided by Barnard Software, Inc. and required to run the TCP/IP-TOOLS products.

After customizing the BSTTPARM.A member please copy the BSTTPARM.A member to a configuration library. We recommend using PRD2.CONFIG to hold this member.

After TCP/IP-TOOLS starts executing it reads two files. The first is the BSTTPARM.A member and the second is from SYSIPT. The first file contains the TCP/IP-TOOLS COMPANY and LICENSE command statements. The second file contains TCP/IP-TOOLS command statements to be processed. TCP/IP-TOOLS considers any statement with an asterisk (*) in column one to be a comment.

Startup Commands to Customize

COMPANY

Enter your company name in the name field of the COMPANY command statement. The company name specified on this statement is passed to the LICENSE command statement processing routines to validate the verification code provided by Barnard Software, Inc.

CPUID

Enter your system's CPU serial number and the model number on the CPUID command statement. The CPU serial and model number specified on this statement are passed to the LICENSE command statement processing routines to validate the verification code provided by Barnard Software, Inc.

LICENSE

Enter the product expiration date in the EXPDATE field on the LICENSE command statement. The format of the EXPDATE field is full-year Julian (for example, 1994365). TCP/IP-TOOLS will begin issuing messages warning of product expiration 45 days prior to the actual expiration date.

Enter the verification code in the VCODE field on the LICENSE command statement. The format of this field is seven numeric digits (for example, 1234567).

The parameters on the COMPANY, CPUID and LICENSE statements are combined to validate the verification code. If an error is detected in processing these statements an error message is issued and the product continues to initialize. However, unless the verification code is validated, TCP/IP-TOOLS cannot be enabled by the TCP/IP-TOOLS ENABLE command.

After initialization completes, you can reissue the COMPANY, CPUID and LICENSE commands through the console interface, if necessary.

TCP/IP-TOOLS ENABLE

The TCP/IP-TOOLS ENABLE command should be the last startup command to be processed. Without this command TCP/IP-TOOLS will not initialize within the partition.

Trial Mode

When the IPv6/VSE TCP/IP stack is started without customizing the BSTTPARM.A member the product enters 'Trial Mode'. When trial mode is active the product will function for 30 days in a limited performance mode. Once you have received your official LICENSE information and customize the BSTTPARM.A member the product will function at full performance for the duration of the license.

Trial mode is indicated by messages appearing on the console

```
R1 0045 // JOB INETOSA6
        DATE 02/22/2010, CLOCK 15/24/59
R1 0045 1T20I  SYS000 HAS BEEN ASSIGNED TO X'FEE' (TEMP)
R1 0045 BSTT000I INITIATED  BSTT6NET Build248 02/15/10 16.24    EP=00520078
R1 0045 BSTT003I COPYRIGHT (C) 1998-2010 BARNARD SOFTWARE, INC.
R1 0045 BSTT700I IPv6/VSE BUILD 248PRE23
R1 0045 BSTT004I CB=TTLA A=0052E000 L=000013FC
R1 0045 BSTT019I VSE 8.20 MODE 31-BIT
R1 0045 BSTT004I CB=COMR A=002E44F0 L=00000108
R1 0499 BSTT000I INITIATED  BSTTX6PC Build248 01/27/10 18.36    EP=00831000
R1 0499 BSTT025W LICENSE WILL EXPIRE IN     31 DAYS
R1 0499 BSTT710I * - - - - - - - - - - - - - - - - - - - - - *
R1 0499 BSTT709I TRIAL MODE DETECTED, LIMITING PERFORMANCE
R1 0499 BSTT710I * - - - - - - - - - - - - - - - - - - - - - *
R1 0499 BSTT045I TCP/IP ID SET TO 66
R1 0497 BSTT000I INITIATED  BSTT6SRV Build248 02/20/10 07.20    EP=0084EB80
R1 0497 BSTT000I INITIATED  BSTT21EP Ver 2.46 04/01/09 11.27    EP=803279D0
R1 0497 BSTT600I     0 INITIATED  ipboot   Ver 2.48 10/12/09 08.02
```

LIBDEF's used by TCP/IP-TOOLS Programs

All of our TCP/IP applications run with our TCP/IP stack and CSI/IBM's TCP/IP stack. Basically, you must run a TCP/IP stack. You must libdef the TCP/IP-TOOLS lib.slib and the stack lib.slib. If they are the same then only one library needs to be in the search chain.

```
// LIBDEF PHASE,SEARCH=(toollib.slib,tcplib.slib)
// LIBDEF SOURCE,SEARCH=(parmlib.slib,toollib.slib)
```

Where …

Toollib.slib is the TCP/IP-TOOLS library.sublibrary.
Tcplib.slib is the TCP/IP library.sublibrary.
Parmlib.slib is the library.sublibrary containing the BSTTPARM.A member.

Upper Case SYSLOG/SYSLST Output

TCP/IP-TOOLS normally uses mixed case SYSLOG (console) and SYSLST (printer) output. This output can be converted to all upper case by using the UCMSG SETPARM. Using the SETPARM to set the variable UCMSG to 'YES' will cause output to be converted to upper case. If the SETPARM is not specified or UCMSG is set to a value other than 'YES' no conversion will occur.

```
// SETPARM UCMSG='YES'
```

Quiet Console Mode

TCP/IP-TOOLS normally writes various message to the VSE/ESA system console. This output can be disabled and all message written to SYSLST using the MSGMODE SETPARM.

```
// SETPARM MSGMODE='QUIET'
```

ASM SOCKET API Redirection

Applications using the ASM SOCKET API (E.g., vendor software) can use the // OPTION SYSPARM='xx' to direct ASM SOCKET API requests to the BSI API routines. If the ASM SOCKET request is not IPv6 enabled the API will automatically redirect the request to the API routines for the stack specified by the following SETPARM.

```
// SETPARM IPV4ID=xx
```

OS390 Execution Mode

Some of TCP/IP-TOOLS programs take advantage of the ',OS390' parameter on the // EXEC JCL statement. The ',OS390' parameter is valid and should be used on VSE/ESA 2.4 and newer systems. This parameter should be omitted from VSE/ESA 1.4, 2.1, 2.2 and 2.3 systems.

Commands Read From SYSIPT

All commands and parameters read from SYSIPT by TCP/IP-TOOLS applications are full 80 column card images. If you use an editor that uses columns 72-80 for sequence or line numbers you must disable this feature. Sequence or line numbers in column 72-80 will be treated by TCP/IP-TOOLS applications as part of the command or parameter.

Chapter 3

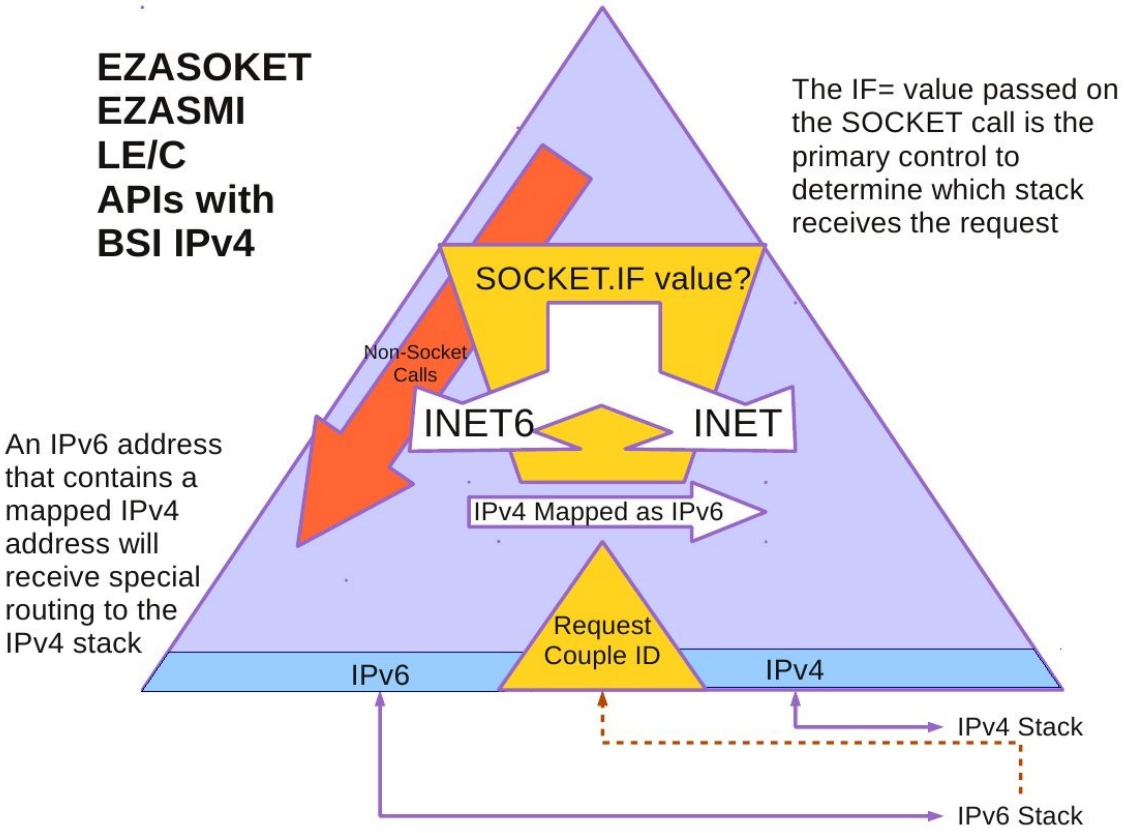
TCP/IP Communications Stack

BSI's IPv6/VSE includes a TCP/IP communication stack. The BSTT6NET TCP/IP stack runs in a stand-alone partition (static or dynamic). No TCP/IP applications (FTP, TN3270E, etc) run in the TCP/IP stack partition.

The BSTT6NET TCP/IP communications stack runs in 31-bit mode and makes extensive use of access registers. The BSTT6NET TCP/IP stack is optimized for dynamic partition usage. For the best possible performance run the BSTT6NET TCP/IP stack in a dynamic partition and run all TCP/IP applications in dynamic partitions also.

Usage of the BSTT6NET TCP/IP stack requires a verification code in the BSTTPARM.A member with feature code 6.

IPv6/VSE System Flow



IPv4 Stack Coupling

When IPv6/VSE requires IPv4 communications, an IPv4 stack is used to provide this processing. This IPv4 stack provides the IPv6/VSE customer with full dual-stack support including our Enhanced Coupling Facility and 6-in-4 tunneling . The EZASMI/EZASOKET API will fully support dual listens (listening for IPv4 and IPv6 connections at the same time) with both stacks active.

When the IPv4 stack is active it is an extension of the IPv6 stack. Or, if you prefer, the IPv6 stack is an extension of the IPv4 stack. It is a dual-stack configuration.

If you are planning to IPv6 enable your applications or plan to run 3rd party IPv6 enabled applications and want them to support connections to IPv4 hosts as well as IPv6 hosts then an IPv4 stack is needed.

Activation of an IPv4 Stack

1. Select a unique 2 digit stack ID and network interface
2. Startup a IPv6/VSE BSTTINET IPv4 stack using this ID
   Use the IPv6/VSE IPv4 Installation Guide for instructions on BSTTINET setup
3. Startup an IPv6/VSE TCP/IP stack
   Include a COUPLE XX command in the IPv6/VSE BSTT6NET stack startup
4. The BSTTINET IPv4 stack is now coupled to the BSTT6NET IPv6 stack.

ASM SOCKET API Redirection

Applications using the ASM SOCKET API (E.g., vendor software) can use the // OPTION SYSPARM='xx' to direct ASM SOCKET API requests to the BSI API routines. If the ASM SOCKET request is not IPv6 enabled the API will automatically redirect the request to the API routines for the stack specified by the following SETPARM.

```
// SETPARM IPV4ID=xx
```

Requirements

The BSTT6NET IPv6 TCP/IP stack supports OSA Express and Hipersocket network interfaces running on z/VSE 4.2+ with IJBOSA at DY47077 (or higher). BSTT6NET can also use CTCA and 6in4 tunneling devices on any version of VSE/ESA or z/VSE executing on a ALS2 processor.

Operating System

- BSI supports IPv6/VSE on all versions and releases of z/VSE 4.2 and higher
- IBM IPv6/VSE 1.1 is supported by IBM on z/VSE 5.1 and 5.2.

Compatibility

The BSI TCP/IP stack was designed to be as compatible as possible with IPv6 RFCs. Various APIs are available including ASM SOCKET, EZASOKET and EZASMI. Converting socket applications to support both IPv4 and IPv6 TCP/IP access is fairly simple and easy. Programs using the BSI TCP/IP stack must LIBDEF the IPv6/VSE installation library prior to any library containing IBM/CSI phases.

Programming

The best information on coding TCP/IP from a mainframe CICS COBOL program can be found in some of the OS/390 manuals as OS/390 has had this functionality for several years.

A "MUST" book is the Redbook "A Beginner's Guide to OS/390 TCP/IP Socket Programming", GG24-2561-00. It walks you though both batch and CICS Socket programming using the "EZASOKET" interface. Of the 350 pages, 200 pages are sample code. We have actually copied the CICS Cobol Server example (using"Cut and Paste") and after removing the RACF call, ran it in VSE using the BSI TCP/IP stack.

For a technical reference, the manual z/OS Communications Server IP Sockets Application Programming Interface Guide and Reference Version 1 Release 9 SC31-8788-06 is the best guide. BSI has implemented 99% of the EZASOKET HLL interface. The only areas not implemented are a few items in the SETSOCKOPT area that were intended for use only by IBM subsystems.

The BSTTEZA.C member of the IPv6/VSE installation library is a COBOL copybook containing definitions of all the parameter that might be used by the EZASOKET HLL interface and for which a length has been defined within the EZASOKET specification.

Tracing

EZASOKETs Trace flag settings

```
// SETPARM IPTRACE='xxxxxxx'
'Y.......' produce call parms trace information
'.Y......' produce iprpl trace information
'..Y.....' produce console messages on entry and exit
'...Y....' produce trace information on SYSLST, not direct to LST queue
'....Y...' produce waitlist trace information
'.....Y..' produce one line entry and exit messages trace information
'......Y.' Force 'Y.......' for any call which results in an error
'.......Y' Trace full SEND/RECEIVE buffer
All flags can be used in combinations to create multiple output entries.

Normally, all trace information (except console traces '..Y....') are
written directly the VSE/POWER LST queue with a job name of 'EZALOGxx'
(xx is the partition identifier). With heavy processing, this XPCC
communication can slow the IP processing to the point that time based
failures occur. The '...Y...' flag can help as it instructs the trace
facility to use a faster write to partition SYSLST. But, by writing to
SYSLST, the trace will be interspersed with normal job output messages
and may actually cause an overprint of the program output. (The tracing
function uses "write before advancing one line'.)
```

Give and Take Socket Performance

The BSI TCP/IP stack supports both global and local Givesocket and Takesocket requests. The default operating mode is Global Givesocket and Takesocket. When global Givesocket and Takesocket mode is active, the stack handles all Give/Take requests. When Local operation mode is active, the API within the application partition, handles all Give/Take requests. Local Give/Take operation mode reduces CPU usage by about 12% (Your mileage will vary!).

The following SETPARM, in the application JCL, activates Local Give/Take operating mode ...

```
// SETPARM LOCALGT=YES
```

Large TCP Window Support

The BSI TCP/IP stack supports large TCP receive windows. The LRGBUF SETPARM provides an external method of enabling or disabling usage of this support by an application.

As a starting point you might find this helpful ...
http://bsiopti.blogspot.com/2010/09/care-and-feeding-of-your-zvse-tcpip.html

First, large TCP window support does not apply to outbound data transfers. Large TCP windows apply to inbound data transfers.

Second, not all applications can benefit from using large TCP windows. The CICS TS CWS (HTTP daemon) or a TN3270E server are examples of interactive applications. Applications that benefit from large TCP windows are applications reading large amounts of data from a remote host.

What applications do this? BSTTFTPC (batch FTP) doing a RETR operation. The IBM VTAPE sever partition reading a remote virtual tape file, etc.

So, the TCP/IP stack will not allocate large TCP window buffers unless all the conditions are correct.

The conditions are ...

```
1) z/VSE 4.2+ with LTWBUF specified -or-
   z/VSE 5.1 with 64-bit available.
2) SHIFT 4|5|6|7 specified in the stack startup.
3) The application has requested large TCP windows buffers.
    I) // SETPARM LRGBUF=YES in the JCL -or-
   II) 'L' specified in the SRBLOK+1 during SOCKET OPEN,TCP
request
4) The remote host TCP/IP stack must also agree to use large
TCP windows. The shift value used for a socket is negotiated
during the connection process. The SYN packets contain an
options header for this purpose and both the local and
remote TCP/IP stacks must agree to use large TCP windows
before they can be used.
```

Currently the only application requesting large TCP window buffers is BSTTFTPC when an RETR operation is being done.

Note: No programming changes are required when // SETPARM LRGBUF=YES is specified in the applications JCL. However, when this option is specified in the JCL, all sockets created by the application will use large TCP window buffers.

The following SETPARM, in the application JCL, enables or disables large TCP receive window support for an application by the stack.

```
// SETPARM LRGBUF=YES|NO
```

Default Maximum Sockets

The default MAXSOC value is 64 sockets on an EZA or LE/C initapi() request. Some applications, including CICS Web Services, do not specify a MAXSOC value resulting in the default maximum sockets allocated. For applications with very high transaction rates the default value of 64 sockets may not be enough. The maximum value is 1024.

The following SETPARM, in the application JCL, overrides the default value ...

```
// SETPARM MAXSOC=nnnn
```

Posting All Partition Subtasks

Normally, the BSI TCP/IP stack will only post the z/VSE subtask that issued a request upon completion of the request. If you are writing an application that issues a socket request from one z/VSE subtask and waits on the request from another z/VSE subtask, you need to inform the TCP/IP stack of this. This SETPARM is not required if you are running z/VSE 4.2 (or higher).

The following SETPARM, in the application JCL, specifies partition level posting ...

```
// SETPARM POST=PARTITION
```

External Packet Capture Facility

The CAPTURE command is used to enable and disable the external packet capture facility. When the packet capture facility is in use, packets are written to the BSTTCAP VSAM ESDS file. The DLBL for the BSTTCAP file can be added to System Standard Labels or to the JCL used to run the stack. When the CAPTURE ON command is used, the BSTTCAP file is automatically open when the stack processed the next packet. When the CAPTURE OFF command is used, the BSTTCAP file is automatically closed after the next packet is processed by the stack. The BSTTCAP file is reset each time it is opened.

After trace data has be captured in the BSTTCAP file, a data can be FTP'ed to a PC in BINARY mode and viewed using Ethereal/Wireshark.

Allocating the BSTTCAP File

The following is a sample job stream for allocating the BSTTCAP file

```
// DLBL IJSYSUC,'VSESP.USER.CATALOG',,VSAM
// EXEC IDCAMS,SIZE=IDCAMS
   DELETE IPV6.VSE.PACKET.CAPTURE.FILE CLUSTER PURGE
   DEFINE CLUSTER -
   (NAME(IPV6.VSE.PACKET.CAPTURE.FILE) -
    SPEED -
    REUSE -
    SHR(2 3) -
    NONINDEXED -
    RECSZ(1500 18425) -
    VOL(SYSWK1))-
   DATA(NAME(IPV6.VSE.PACKET.CAPTURE.FILE.DATA) CISZ(18432) CYL(10 0))
   LISTC ENT(IPV6.VSE.PACKET.CAPTURE.FILE) ALL
/*
```

The highlighted fields in the above sample JCL may be changed. The VSAM ESDS cluster name, volume and number of cylinders to allocate are user modifiable. The CISZ should not be changed. Each cylinder allocated will hold slightly less than 1MB of trace data. Therefore, a cylinder allocation of 100 cylinders will hold a little less than 100MB of trace data.

Add BSTTCAP to Standard Labels

This sample JCL will add a label to System Standard labels. You should also add the label for the BSTTCAP file to your normal system IPL procedure.

```
// JOB STDLABEL
// OPTION STDLABEL=DEL
BSTTCAP
/*
/&
// JOB STDLABEL
// OPTION STDLABEL=ADD
// DLBL BSTTCAP,'IPV6.VSE.PACKET.CAPTURE.FILE',,VSAM,CAT=VSESPUC
/*
/&
```

Capturing Data

The CAPTURE command is used to enable (ON) or disable (OFF) the capture of data.

To enable the external packet capture facility ...

```
MSG BSTT6NET,D=CAPTURE ON
```

The BSTTCAP file will automatically open when the next packet is processed by the stack.

To disable the external packet capture facility ...

```
MSG BSTT6NET,D=CAPTURE OFF
```

The BSTTCAP file will automatically close after the next packet is processed by the stack.

Transferring BSTTCAP Data for Viewing

The following sample BSTTFTPC JCL can be used to transfer the BSTTCAP file to a remote host for viewing by the ethereal/wireshark packet viewer.

```
// EXEC BSTTFTPC,SIZE=BSTTFTPC
ID 66
OPEN ::FFFF:192.168.1.60
USER userid
PASS password
*
CWD directory
INPUT VSAM BSTTCAP
TYPE I
PASV
STOR file.name.pcap
*
QUIT
/*
```

The hilighted fields in the above sample JCL can be modified by you for your installation.

You can now use wireshark/ethereal to open the packet capture file.

TCP/IP EZA Application Considerations

Compiling and Assembling TCP/IP EZA Applications

Compiling or Assembling requirements are dependent on the level of VSE.

For Pre-VSE 2.5, the BSI sublibrary must be available during the Compiling or Linking process.

For later systems, but prior to z/VSE 4.2 plus APAR DY47077 (IBM IPv6 Support), if the program is using any IPv6 functions, the BSI sublibrary must be available during any Assemblies using the EZASMI macro.

Running TCP/IP EZA Applications

For Pre-VSE 2.5, the BSI sublibrary must be available during program execution.

For later systems, but prior to z/VSE 4.1, the BSI sublibrary must be LIBDEFed prior to any IBM libraries during program execution and the EZASOH00 phase must be removed from the SET SDL list.

For z/VSE 4.1 and z/VSE 4.2 prior to APAR DY47077, it is strongly recommended that the BSI sublibrary be LIBDEFed prior to any IBM libraries during program execution and that the EZASOH00 phase be removed from the SET SDL list. Alternatively, the IBM Multiple-Vendor EZA Interface may be used. This interface requires that the IBM version of EZASOH00 be LIBDEFed prior to the BSI version of EZASOH00. Please see the information below about the IBM Multiple-Vendor EZA Interface for setup instructions.

For z/VSE 4.2 plus APAR DY47077 (IJBOSA IPv6 Support), it is recommended that the BSI sublibrary be LIBDEFed prior to any IBM libraries during program execution and that the EZASOH00 phase be removed from the SET SDL list. Alternatively, the IBM Multiple-Vendor EZA Interface may be used. The IBM Multiple-Vendor EZA Interface is required for those partitions executing programs that utilize both the IPv6/VSE and TCP/IP for VSE products. This interface requires that the IBM version of EZASOH00 be LIBDEFed prior to the BSI version of EZASOH00. Please see the information below about the IBM Multiple-Vendor EZA Interface for setup instructions.

Known users of the EZA API include: CICS TS and VSE/POWER.

IBM Multiple-Vendor EZA Interface

With z/VSE 4.1, IBM introduced a new EZA Interface to better support multiple TCP/IP products on a single z/VSE partition. The use of this interface does require either additional JCL statements or the passing of additional information during the INITAPI EZA call. The following information summarizes what may be found in 'TCP/IP for VSE/ESA - IBM Program setup and supplementary information' manual (SC33-6601). More details may be found in that manual. In addition, for CICS environments, this interface also requires the activation of the EZA 'TASK-RELATED-USER-EXIT' (TRUE). Please see the chapter 'CICS Considerations for the EZA Interfaces' in the above manual for details.

BSI added support for this interface with Build 2.48. Do not use this interface with prior Builds.

To use this interface, in addition to providing the Stack-ID, the name of the applicable Vendor Interface Routine must also be passed to the EZA Interface. This is accomplished by using one of the following methods:

1. Pass the name of the Vendor Interface Routine during the INITAPI call by specifying the name in the ADSNAME parameter.
2. Using a setparm: // SETPARM EZA$PHA='routine-name'

When these are required varies depending on the z/VSE environment.

In the case where all stacks are licensed from BSI, the use of a system SETPARM during IPL is recommended: "// SETPARM SYSTEM EZA$PHA='BSTTIPS1' ". This removes any requirement that the vendor routine be specified within each partition or via an INITAPI call.

In the case where stacks from different vendors exists within a z/VSE image, how the stacks are being used must be considered. If a specific partition is only accessing a stack(s) from one vendor, they will need to specify the correct SETPARM as required by the stack vendor. This removes any requirement that the routine name be specified during the INITAPI call.

The most complicated situation is where a program(s) within a single partition access multiple stacks from multiple vendors. Because there is only one SETPARM EZA$PHA setting within a single partition, and because it is an override, not a default, the SETPARM EZA$PHA must **NOT** be used. All programs **MUST** provide the ADSNAME values by using an INITAPI call as the first EZA call. In addition, any program not using the default stack as indicated by the SYSPARM value, MUST provide the correct TCPNAME value. To address some of the maintenance and support issues with actually coding a vendor phase name within every program, BSI provides an optional BSI TCP/IP EZA API Multiplexer described below.

BSI TCP/IP EZA API Multiplexer

The BSI TCP/IP EZA API Multiplexer is an optional use feature. By using the Multiplexer, the programmer is no longer required to know or code the name of the vendor routine normally required by the IBM Multiple-Vendor EZA Interfaces as delivered with z/VSE 4.1. Additionally, it allows a use of site specific stack TCPNAMEs that do not need to be of the form "SOCKETnn" when specified during an INITAPI call. The BSI EZA Multiplexer is "vendor independent" and will properly handle requests for non-BSI stacks.

The Multiplexer is controlled by a system wide table that is generated as an SVA eligible phase that is named BSTTCPMC. The Multiplexer Control Table is designed to also be used at some point in the future by LE/C and BSD Multiplexers and thus includes some information that is not currently used. A sample job is delivered in the BSI sublibrary under the name BSTTCPMC.Z. The following is an excerpt from that sample:

```
BSTTCPMC CSECT
BSTTCPMC RMODE ANY
BSTTCPMC AMODE ANY
* SAMPLE BSTTCPMC MULTIPLEXER CONTROL TABLE
         BSTTCPME SYSID='00',TCPNAME='$DEFAULT',                    X
               LEPHASE='XXXXXXXX',EZAPHASE='EZASOH00',              X
               BSDPHASE='YYYYYYYY'
         EZATCPME SYSID='33',TCPNAME='LFP',                         X
               LEPHASE='XXXXXXXX',EZAPHASE='IJBLFPEZ',              X
               BSDPHASE='YYYYYYYY'
         BSTTCPME SYSID='66',TCPNAME='IPV6',                        X
               LEPHASE='XXXXXXXX',EZAPHASE='BSTTIPS1',              X
               BSDPHASE='YYYYYYYY'
         END
```

Each invocation of the BSTTCPME macro provides the information related to a specific stack. Multiple entries can indicate the same stack SYSID allowing multiple TCPNAMEs to reference the same TCP/IP stack. A special entry for TCPNAME='$DEFAULT' is available and will be used with the programmer provides neither a TCPNAME or SYSID. All operand values must be specified within single quotes as shown in the example.

SYSID=

The SYSID specifies the two character system ID specified during the TCP/IP stack startup.

TCPNAME=

The TCPNAME operand specifies a name that can be used by a program as passed in the INITAPI TCPNAME field value. A special TCPNAME of '$DEFAULT' may be used to indicate a default stack when one is not specified by the programmer or if the program passes a TCPNAME that does not exist in the table.

EZAPHASE=

The name of the EZA Vendor Interface Routine as specified by the specific stack vendor. For BSI stacks, the correct name is 'BSTTIPS1'. At the time of this writing, the correct name for the CSI TCP/IP for VSE product is 'EZASOH99'. Please refer to the documentation provide by each vendor for this value.

LEPHASE=

The name of the LE/C Vendor Interface Routine as specified by the specific stack vendor. This field is currently not used, but must be specified. Any value may be used.

BSDPHASE=

The name of the BSD/C Vendor Interface Routine as specified by the specific stack vendor. This field is currently not used, but must be specified. Any value may be used.

The generated table phase must be placed in the SVA or available via LIBDEFs for any partition using the Multiplexer. If the table is not located as execution time, the program will receive an ERRNO value of 20114. Once the table is loaded by the first INITAPI call within a partition, it will not be refreshed until end of step even if a replacement phase is cataloged.

The actual multiplexer phase name is BSTTCPMX. It must be specified in one of two ways:

1. Using a system wide setting via “// SETPARM SYSTEM EZA$PHA='BSTTEZMX' ”
2. Specifying it's name in each partition using “// SETPARM EZA$PHA='BSTTEZMX' ”

If a SETPARM is not located, the IBM Multiple-Vendor EZA Interface will use “EZASOH99” which will produce undesired results. The phase BSTTCPMX must be available via the LIBDEF for the partition and must not be loaded into the SVA. If the phase is not available at execution time, the program will receive an ERRNO value of 20113.

LE/C TCP/IP Application Considerations

Note: This interface, as defined by IBM, does not currently support IPv6 connections prior to z/VSE 4.2 plus APAR DY47077 (IBM IPv6 Support).

Running LE/C TCP/IP Applications

This interface is not available prior to VSE/ESA 2.3.

For Pre-VSE 2.3, the BSI sublibrary must be available during program execution.

For later systems, but prior to z/VSE 4.3, the BSI sublibrary must be LIBDEFed prior to any IBM libraries during program execution and the customer must rename the BSTTTCPV.PHASE (supports IPv4 only) or BSTTTCP6.PHASE (supports IPv4 and IPv6) to $EDCTCPV.PHASE in the BSI lib.slib.

1. If using IPv6/VSE Build 2.47 or earlier, the BSI LE/C TCP/IP Application Interface requires that the EZASOH00 phase be removed from the SET SDL list. Please review the EZA TCP/IP Application Considerations elsewhere in this manual as the EZA interface may still require that the EZASOH00 phase be removed from the SET SDL list. the BSI phase and the EZASOH00 phase must be removed from the SET SDL list.

For z/VSE 4.3 and later, it is recommended that the BSI sublibrary be LIBDEFed prior to any IBM libraries during program execution and the customer must rename the BSTTTCPV.PHASE (supports IPv4 only) or BSTTTCP6.PHASE (supports IPv4 and IPv6) to $EDCTCPV.PHASE in the BSI lib.slib. Alternatively, the IBM LE/C TCP/IP Multiplexer may be used. Please see the information below about the IBM LE/C TCP/IP Multiplexer for setup instructions.

Independent of the version of z/VSE, when using IPv6/VSE Build 2.47 or earlier, the BSI LE/C TCP/IP Application Interface requires that the EZASOH00 phase be removed from the SET SDL list. Please review the EZA TCP/IP Application Considerations elsewhere in this manual as the EZA interface may still require that the EZASOH00 phase be removed from the SET SDL list. the BSI phase and the EZASOH00 phase must be removed from the SET SDL list.

Known users of the LE/C API include: CICS TS, MQSeries, VSE Connector Server, DB2 and the SNMP Agent for z/VSE.

IBM LE/C TCP/IP Multiplexer

Information on setup and use of the IBM LE/C TCP/IP Multiplexer can be found in the following manual:

IBM Language Environment for z/VSE – C Run-Time Library Reference; SG33-6689-05
Section 3.2 Configuring the LE/C TCP/IP Socket API Multiplexer
http://publibz.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/fl2cle05/3.2

The name of the TCP/IP interface phase supplied by IPv6/VSE is BSTTTCPV.PHASE (supports IPv4 only) or BSTTTCP6.PHASE (supports IPv4 and IPv6).

BSD/C TCP/IP Application Considerations

Please contact BSI for information on using the BSI BSD/C TCP/IP interface.

Network Interface Device Support

The BSTT6NET TCP/IP stack currently supports several network interface devices.

- OSA Express (OSAX) devices running on QDIO mode
- Hipersockets (HIPR) devices
- 6in4 static tunnel devices
- CTCA devices

Partition Size

The BSTT6NET TCP/IP stack requires a ***minimum 24MB partition***. This amount of storage will allow for stack startup and storage for a few users. Running the BSTT6NET TCP/IP stack in a 24MB partition will allow for basic testing but 24MB is to small for production usage.

The basic formula is 24MB + 124K per TCP connection. For example, on a system with 100 TCP (TN3270E) connections plus 10 FTP (client and server) connections would require 24MB +124 x 110 (13640K) = ~38MB.

The maximum number of TCP connections currently supported is 4095. To support the maximum number of connections the stack would need to run in a 520MB partition.

Partition Priority

Ideally a production VSE/ESA system should have two network interfaces. A high priority BSTT6NET TCP/IP stack for interactive traffic would use the 1st network interface. The 2nd network interface would be used by lower priority BSTT6NET TCP/IP stack for bulk data transfer (FTP) traffic.

The BSTT6NET TCP/IP stack may be balanced or even run lower priority than the TCP/IP applications it is servicing. On VSE/ESA systems with a single network interface this allows for greater flexibility in priority management.

Remember that TCP/IP is a timer driven protocol. The BSTT6NET TCP/IP stack can run at lower priorities but still must have access to the CPU when timer processes need to run.

Network Traffic Filtering

Any network interface used by VSE/ESA should be attached to a filtering switch.

- Filter all Ethernet packets except types x'86DD' (IPv6).
- Filter all IP Ethernet packets except IP protocol 58 (ICMPv6), 6 (TCP), 17 (UDP).

Port Number Selection

RFC 1340 Assigned Numbers states ...

"The Well Known Ports numbers are controlled and assigned by the IANA and on most systems can only be used by system (or root) processes or by programs executed by privileged users. The assigned ports use a small portion of the possible port numbers. For many years the assigned ports were in the range 0-255. Recently, the range for assigned ports managed by the IANA has been expanded to the range 0-1023."

Please keep this in mind when selecting port numbers to use with IPv6/VSE applications.

Link Local IPv6 Address

For IPv6 each network interface has 2 IP addresses. The Link Local IPv6 address and the assigned IPv6 address. The Link Local IPv6 address can only be used on the local Link.

For example, the assigned IPv6 address might be 806::1:2. The Link Local IPv6 address is created dynamically. It is based on the MAC address of the interface. So, if the MAC address is 02000000000a then the Link Local IPv6 address is fe80::0001:0200:0000:000a. The 0001 inserted in front of the MAC address is the interface number (01).

Unique Local IPv6 Addresses

http://en.wikipedia.org/wiki/Unique_local_address

If you do not have an assigned/registered IPv6 network prefix for you company, you can still work with IPv6 networking using IPv6 Unique Local Addresses. The wiki link explains the concept of Unique Local Addresses.

Quoted from the Wiki article ...

A unique local address (ULA) is an IPv6 address in the block fc00::/7 or fd00::/7, defined in RFC 4193. It is the approximate IPv6 counterpart of the IPv4 private address. Unique local addresses are available for use in private networks, e.g. inside a single site or organization or spanning a limited number of sites or organizations. They are not routable in the global IPv6 Internet.

Waiting for the TCP/IP stack to come up

You can use the BSTTWAIT program to cause a job stream to wait until the BSTT6NET TCP/IP stack is ready. The BSTTWAIT program will wait for the BSTT6NET TCP/IP stack to come up or a message from the operator. Use the MSG xx (xx is the partition id) command to terminate the BSTTWAIT program early.

```
// LIBDEF PHASE,SEARCH=(ttlib.slib)
// OPTION SYSPARM='nn'
// EXEC BSTTWAIT,SIZE=BSTTWAIT
/*
```

ttlib.slib is the IPv6/VSE lib.slib

nn is the BSTT6NET stack ID

Using the Services Table

The TCP/IP stack supports a services table. The services table is in the BSTTSERV.T member of the BSI lib.slib. The services table defines services by name and port. Applications can access the services table to determine the port number they should be using.

ASM SOCKET API application can use the GETPORTBYNAME Control request to query the services table. EZA API applications can use the GETHOSTINFO to query the services table. At this time, only the BSI TCP/IP stacks support a services table.

BSI applications support using the services table. When an OPEN command is processed, the IP address is resolved. If the port number is specified on the OPEN command, this is the port number used. If the port number is not specified, the services table will be queried to determine port number to be used. If the services table does not contain an entry for that application or the port number is zero, the default port number (hard-coded in the application) is used.

The advantage of using the BSTTSERV.T services table to specify a port number is simple. Changing the port number used for a service can be done without any changes to JCL or applications.

Sample BSTTSERV.T Services Member

```
CATALOG BSTTSERV.T                          REPLACE=YES
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*   COMMON SERVICES USED                                                  *
*                                                                         *
*   THIS TABLE PROVIDES A SMALL LIST OF SERVICES AVAILABLE                *
*                                                                         *
*   THE TABLE SHOWS THE SERVICE NAME AND THE DEFAULT PORT                 *
*   NUMBER. THE PROTOCOL IS TREATED AS A COMMENT BUT MAY BE               *
*   USED IN THE FUTURE.                                                   *
*                                                                         *
*   ANY LINE WITH AN ASTERISH (*) IN COLUMN 1 IS A COMMENT.               *
*                                                                         *
*   APPLICATION CAN USE THE ASM SOCKET API OR THE EZA API                 *
*   TO QUERY THE DEFAULT PORT NUMBER FOR THE SERVICE THEY ARE             *
*   PROVIDING. FOR EXAMPLE, AN FTP CLIENT OR SERVER CAN                   *
*   QUERY THE FTP SERVICE FOR THE DEFAULT PORT NUMBER TO USE.             *
*                                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
FTP              21 TCP
TELNET           23 TCP
SMTP             25 TCP
DOMAIN           53 UDP
HTTP             80 TCP
NTP             123 TCP
HTTPS           443 TCP
EXEC            512 TCP
SYSLOG          514 UDP
PRINTER         515 TCP
/+
```

CICS TS Socket Domain Performance

Applications using the CICS TS Socket Domain can have poor performance. This is because CICS TS does not directly wait on the ECB used by the Socket Domain. Instead CICS TS checks these ECBs periodically based on the CICS TS ICV value. BSI provides a performance enhancement to assist application using the Socket Domain interface.

The program BSTTPLT0 runs in the CICS TS PLT StartUp and PLT Shutdown. This program issues a CICS START for the transaction that runs the BSTTPLT1 program. The BSTTPLT1 program will run until CICS termination. If the BSTTINET TCP/IP stack is recycled without CICS TS being recycled the BSTTPLT0 can be used to terminate and restart the BSTTPLT1 program.

These transactions and program must be defined to CICS TS. Any group and transaction names can be used. The G(BSTT) and transactions BST0/BST1 are examples.

```
         DFHPLT TYPE=ENTRY,PROGRAM=BSTTPLT0

E G(BSTT)
ENTER COMMANDS
 NAME     TYPE         GROUP
 BSTTPLT0 PROGRAM      BSTT
 BSTTPLT1 PROGRAM      BSTT
 BST0     TRANSACTION  BSTT
 BST1     TRANSACTION  BSTT

 CEDA  View PROGram( BSTTPLT0 )
  PROGram        : BSTTPLT0
  Group          : BSTT
  DEscription    : PLT PROGRAM
  Language       : Assembler
  RELoad         : No
  RESident       : No
  USAge          : Normal
  USEsvacopy     : No
  Status         : Enabled
  RSl            : 00
  Cedf           : Yes
  DAtalocation   : Any
  EXECKey        : Cics
```

```
CEDA  View PROGram( BSTTPLT1 )
 PROGram        : BSTTPLT1
 Group          : BSTT
 DEscription    : PLT PROGRAM
 Language       : Assembler
 RELoad         : No
 RESident       : No
 USAge          : Normal
 USEsvacopy     : No
 Status         : Enabled
 RSl            : 00
 Cedf           : Yes
 DAtalocation   : Any
 EXECKey        : Cics

CEDA  View TRANSaction( BST0 )
 TRANSaction    : BST0
 Group          : BSTT
 DEscription    : START BSTT PLT PROGRAM
 PROGram        : BSTTPLT0
 TWasize        : 00000             0-32767
 PROFile        : DFHCICST
 PArtitionset   :
 STAtus         : Enabled           Enabled | Disabled
 PRIMedsize     : 00000             0-65520
 TASKDATALoc    : Any               Below | Any
 TASKDATAKey    : Cics              User | Cics

CEDA  View TRANSaction( BST1 )
 TRANSaction    : BST1
 Group          : BSTT
 DEscription    : LONG RUNNING ECB WATCHER
 PROGram        : BSTTPLT1
 TWasize        : 00000             0-32767
 PROFile        : DFHCICST
 PArtitionset   :
 STAtus         : Enabled           Enabled | Disabled
 PRIMedsize     : 00000             0-65520
 TASKDATALoc    : Any               Below | Any
 TASKDATAKey    : Cics              User | Cics
```

BSTT6NET commands

ATTACH

```
ATTACH TCP/IP
```

The ATTACH command is used to start the TCP/IP stack subtask. This command must be present in the BSTT6NET startup command and must also be the last command.

BACKLOG

```
BACKLOG nn
```

The BACKLOG value defines the TCP/IP stack's default connection queue depth. Values from 5 to 20 are valid. The default is 10.

CAPTURE

```
CAPTURE ON|OFF
```

The CAPTURE command enables (ON) or disables (OFF) the stack's external packet capture facility. The BSTTCAP VSAM ESDS capture file is used by this command. When the CATPTURE ON command is issued, the capture facility automatically opens the BSTTCAP file when the next packet is processed by the stack. When the CAPTURE OFF command is issued, the BSTTCAP file is automatically closed after the next packet is processed by the stack.

CLOSETOV

```
CLOSETOV nnn
```

The CLOSETOV command specifies the CLOSE Time Out Value. ***nnn*** is specified in minutes and the default is 15 minutes.

COUPLE

```
COUPLE id
```

The COUPLE command is used to tell the BSTT6NET TCP/IP stack the stack ID of the IPv4 stack. These TCP/IP stacks are considered to be coupled together. The default is no coupling.

DEVICE

```
DEVICE device_name OSAX cuu portname cuu3 LAYER2 macaddr
DEVICE VIRTUALx    OSAX cuu dev_name cuu3 LAYER2 macaddr
DEVICE device_name HIPR cuu portname cuu3
DEVICE device_name 6IN4 0 IPv4.address
DEVICE device_name CTCA cuu
```

The DEVICE command is used to define a network adapter to the BSTT6NET TCP/IP stack. This command must precede any LINK commands that reference the specified *device_name*. OASX indicates an OSA Express adapter. HIPR indicates a Hipersocket device. OSAX or HIPR cuu must specify the lowest cuu address of the required 3 OSAX/HIPR device addresses. If the control cuu is not in sequence with the OSAX/HIPR read/write ports cuu3 can be used to specify this cuu address. When OSA Express 3 adapters are used, the port number being used is specified in the adapter_number field of the LINK statement.

OSAX devices are only supported under z/VSE 4.2 + DY47077.
OSAX/HIPR devices require SETPFIX LIMIT=(256K,1100K) per DEVICE command.
Cuu is the unit address of the device.

portname is the OSAX or HIPR device portname.

OSAX devices requiring layer 2 support instead of the default layer 3 network interface must specify the literal LAYER2. When exploiting Layer 2 support with a z/VSE system that is not connected to a z/VM VSwitch, the user must manually configure a unique MAC address (which must be different from the default MAC address of the port) for the z/VSE network interface. The MAC address is specified as 12 hexadecimal characters. The first byte of the actual MAC address used will be forced to x'02' indicating a user defined MAC address. Layer 2 support is available only on z/VSE 5.1 (or higher) systems.

The DEVICE called VIRTUALx (x=0-9,A-Z) is a virtual OSAX (OSA Express) device. The portname field is used to reference the real OSA Express device. Virtual device specifications must precede the DEVICE statement for the real OSA Express device. This allows you to have multiple IP/VMAC addresses for each physical OSA Express adapter. Each device (real or virtual) can be on a different VLAN too.

IPv4.address is the IPv4 IP address of the remote end of the 6IN4 tunnel.

Sample VSE ADD statements

```
ADD 600:602,OSAX        Sample OSA Express ADD
ADD 600:602,OSAX,01     Sample Hipersocket ADD
ADD 300:301,CTCA,EML     Sample CTCA ADD
```

CTC Connections

CTC connections under VM normally use cross-coupled virtual CTC definitions. The CTC read address of system A is coupled with the CTC write address of SYSTEM B and the CTC write address of SYSTEM A is coupled with the CTC read address of SYSTEM B. The read address in the even numbered address. The write address is the odd address. When using real CTC connections between systems, one system must specify a LINK command-using adapter 0 while the other system must specify a LINK command-using adapter 1.

6in4 Driver

The IPv6/VSE 6in4 static tunnel driver is designed to allow you to pass IPv6 packets over an IPv4 link to a IPv6 Tunnel Broker. The IPv6 Tunnel Broker provides (generally at no cost) an IPv6 P.O.P (Point Of Presence) gateway to the IPv6 backbone. The 6in4 driver is designed for testing IPv6 functionality and is generally not a high performance interface. However, the driver does allow you to use your existing IPv4 infrastructure to test IPv6 functionality.

DNS

```
DNS IPv6_address PRI|SEC
```

The DNS command specifies a presentation format IPv6 address of a DNS server. If the PRI option is specified the primary DNS IP address is set or updated. If the SEC option is specified the secondary DNS IP address is set or updated. If the DNS command is issued without an option (PRI/SEC) the first DNS command sets the primary DNS IP address and the second DNS command sets the secondary DNS IP address.

DOMAIN

```
DOMAIN .name
```

The DOMAIN command specifies a domain name suffix to be used by the DNS look up routines. The name must begin with a leading period. The length is limited to 55 characters. Up to 4 DOMAIN commands may be specified. If the name is defined with a HOST command this value is used. Look ups are done on the qualified names before the unqualified name look up is done.

DOS

```
DOS ADD IPv6-address mask
DOS DEL IPv6-address mask
DOS PRINT
```

The DOS command is used to have the BSTT6NET TCP/IP stack deny service to an IP address or range of IP addresses. **Warning:** Be sure you specify the correct subnet mask or you can deny service to a broad range of IP addresses. The DOS PRINT command will print the DOS table.

FILTER

```
FILTER ON|OFF
```

The FILTER command is used to enable or disable packet filtering. The default is OFF. When FILTER ON is specified echo packet and packet sent to the wrong network interface are discarded before processing.Mac Add

GARBAGE

```
GARBAGE ON|OFF
```

The GARBAGE command adjusts the TCP/IP packet type used in TCP Keep Alive processing. GARBAGE OFF (default) sets NULL packet mode. GARBAGE ON set garbage packet mode (single byte of out-of-sequence data).

HOST

```
HOST name IPv6_address
```

The HOST command specifies a host file entry. The host file associates a name with a presentation format IPv6 address. When performing DNS name-to-IP or IP-to-name lookups the host file is searched first. Multiple HOST commands can be specified in the BSTT6NET startup commands. At least one HOST command is required for each network adapter used by the BSTT6NET TCP/IP stack. This HOST statement identifies the local IP address and its associated name. When the IP address specified in the HOST command specifies ::0 the HOST entry will be deleted. If the name specified in the HOST command already exists the IPv6 address of the name is replaced. A HOST command that does not contain a colon (:) (specifying an IPv6 address) will be ignored. See the installation library member HOSTS.T for instructions on setting up a common HOST command member for use with multiple TCP/IP stacks. Host names defined with the HOST command are limited to 56 characters.

HOTSWAP

```
If your primary DEVICE command is ...
DEVICE OSAX610 OSAX 610 portname 612

Use this HOTSWAP command ...
HOTSWAP OSAX610 0710 0712 0
```

The HOTSWAP command specifies a spare OSA Express adapter and port that can be used by the BSTTINET stack in the event the primary OSA Express device fails. The HOTSWAP command specifies the device name and cuu addresses of the read/write pair and the control cuu address. The port number to use is also specified. The cuu address must be specified as 4 character hexadecimal values.

ID

```
ID nn
```

The ID command specifies the TCP/IP stack identifier. The TCP/IP stack identified is a 2 digit number in the range 00 to 99. This command must be specified for proper stack startup processing.

INTERVAL

```
INTERVAL nnn [EXCEPT lport lport lport …]
```

The INTERVAL specifies the number of seconds the TCP/IP stack (BSTT6NET) waits after receiving a packet for a connection before it sends a keep-alive packet for that connection. This command must be specified in the startup commands. Not specifying this command in the BSTT6NET startup results in INTERVAL 0 which disables Keep-Alive processing. If specified in the BSTT6NET startup the minimum value is 60 and should be a multiple of 60 seconds. The EXCEPT clause identifies up to 8 local ports (lport) to be excluded from keep-alive processing. The EXCEPT clause is optional.

IP ARPTAB

```
IP ARTTAB
```

The IP ARPTAB command displays the contents of the ARP table.

Sample Output

```
msg r2,d=ip arptab
AR 0015 1I40I  READY
R2 0046 BSTT010I IP ARPTAB
R2 0502 BSTT613I NI St Mac Address  IP Address
R2 0502 BSTT613I -- -- ------------ -------------------------
R2 0502 BSTT613I  1  R 020000000003 fe80::200:0:100:3
R2 0502 BSTT613I  1  R 020000000003 806::1:1
R2 0502 BSTT613I  1  R 020000000009 806::1:3
R2 0502 BSTT613I  1  R 020000000009 fe80::200:0:300:9
R2 0502 BSTT011I COMMAND PROCESSING COMPLETE
```

IP BPOOL

```
IP BPOOL
```

The IP BPOOL command displays the buffer pool statistics.

Sample Output

```
BSTT010I IP BPOOL
BSTT613I pool= 0 count  100 bsize=  32, sem=8187, inuse=   7 (  7%)
BSTT613I pool= 1 count 1024 bsize=1524, sem=8183, inuse=   5 (  0%)
BSTT613I pool= 2 count   32 bsize=65567, sem=8182, inuse=   0 (  0%)
BSTT011I COMMAND PROCESSING COMPLETE
```

IP CLOSE

```
IP CLOSE nnn
```

The IP CLOSE command closes all connections on the specified local port.

IP IP2NAME

```
IP IP2NAME IPv6_address
```

The IP IP2NAME command translates a IPv6 presentation format IPv6 address to a name. Reverse DNS lookups are supported only for host names defined using a HOST command to the stack.

Sample Output

```
BSTT010I IP IP2NAME 806::1:2
BSTT612I DNS RESPONSE ZVSE42
BSTT011I COMMAND PROCESSING COMPLETE
```

IP NAME2IP

```
IP NAME2IP name
```

The IP NAME2IP command translates a name into a presentation format IPv6 address.

Sample Output

```
BSTT010I IP NAME2IP ZVSE42
BSTT612I DNS RESPONSE 806::1:2
BSTT011I COMMAND PROCESSING COMPLETE
```

IP NETINFO

```
IP NETINFO
```

The IP NETINFO command displays information about the network devices in use. The primary and secondary DNS server IP addresses are also shown.

Sample Output

```
msg r2,d=ip netinfo
AR 0015 1I40I  READY
R2 0046 BSTT010I IP NETINFO
R2 0502 BSTT608I DEVICE 01 NAME OSAX720  MAC 020000000008 MTU   8192
R2 0502 BSTT693I LL  fe80::200:0:100:8 /128
R2 0502 BSTT686I IP  806::1:2 /64
R2 0502 BSTT687I DNS 806::1:10 806::1:11
R2 0502 BSTT626I INTERVAL   120 SECONDS
R2 0502 BSTT639I BUFFERS: THREAD   115 TRANSMIT   139 RECEIVE   139
R2 0502 BSTT011I COMMAND PROCESSING COMPLETE
```

IP NETSTATS

```
IP NETSTATS
IP LOGSTATS
```

The IP NETSTATS display network statistics on Console/SYSLST.

The IP LOGSTATS command displays network statistics on SYSLST.

Sample Output

```
BSTT010I IP NETSTATS
BSTT011I COMMAND PROCESSING COMPLETE
```

IP PING

```
IP PING IPv6_address
IP PING name
```

The IP PING command pings the specified IPv6 address. The IPv6_address can be specified as a presentation format IPv6 address or as a name. If a name is specified the NAME2IP routine is used to convert the name to an IPv6 address.

Sample Output

```
msg r2,d=ip ping 806::1:3
AR 0015 1I40I  READY
R2 0046 BSTT010I IP PING 806::1:3
R2 0502 BSTT688I PINGING 806::1:3
R2 0502 BSTT689I REPLY TIME     6 FROM 806::1:3
R2 0502 BSTT689I REPLY TIME     5 FROM 806::1:3
R2 0502 BSTT689I REPLY TIME     5 FROM 806::1:3
R2 0502 BSTT011I COMMAND PROCESSING COMPLETE
```

IP RAWINFO

```
IP RAWINFO
```

The IP RAWINFO command is used to display information about RAW Socket connections.

Sample Output

```
msg s1,d=ip rawinfo
AR 0015 1I40I  READY
S1 0048 BSTT010I IP RAWINFO
S1 0102 BSTT613I Raw        Remote IP Proto Queue ID
S1 0102 BSTT613I --- ---------------- ----- ----- --
S1 0102 BSTT613I   0         806::1:3     1    -1 P2
S1 0102 BSTT011I COMMAND PROCESSING COMPLETE
```

IP ROUTE

```
IP ROUTE ADD device_name network_address network_mask gateway metric
IP ROUTE DEL device_name network_address network_mask gateway
IP ROUTE PRINT
```

The IP ROUTE command will add, delete or print the current routing table. The metric value will default to zero and may be omitted unless the route defines a non-local network.

Sample Output

```
msg r2,d=ip route print
AR 0015 1I40I  READY
R2 0046 BSTT010I IP ROUTE PRINT
R2 0502 BSTT685I NI MT NETWORK/MASK/GATEWAY
R2 0502 BSTT684I 01 01 :: :: 806::1:1
R2 0502 BSTT684I 00 00 ::1 /128 ::1
R2 0502 BSTT684I 00 00 806:: /128 806::1:2
R2 0502 BSTT684I 00 00 806::1:2 /128 806::1:2
R2 0502 BSTT684I 01 00 806:: /64 ::
R2 0502 BSTT684I 00 00 fe80:: /128 fe80::200:0:100:8
R2 0502 BSTT684I 00 00 fe80::200:0:100:8 /128 fe80::200:0:100:8
R2 0502 BSTT684I 01 00 fe80:: ffc0:: ::
R2 0502 BSTT011I COMMAND PROCESSING COMPLETE
```

IP TRACE

```
IP TRACE ON
IP TRACE OFF
```

The IP TRACE enables or disables tracing of the stack. Trace data is printed on SYSLST.

Sample Output

```
BSTT010I IP TRACE ON
BSTT011I COMMAND PROCESSING COMPLETE
```

IP TRACERT

```
IP TRACERT IPv6_address
IP TRACERT name
```

The IP TRACERT command traces the network path taken to reach the specified destination.

Sample Output

```
msg r2,d=ip tracert 806::1:3
AR 0015 1I40I  READY
R2 0046 BSTT010I IP TRACERT 806::1:3
R2 0502 BSTT691I TRACING ROUTE TO 806::1:3
R2 0502 BSTT692I     1     5     5     5 806::1:3
R2 0502 BSTT011I COMMAND PROCESSING COMPLETE
```

IP UDPINFO

```
IP UDPINFO
```

The IP UDPINFO command displays information about UDP sockets.

Sample Output

```
msg r2,d=ip udpinfo
AR 0015 1I40I  READY
R2 0046 BSTT010I IP UDPINFO
R2 0502 BSTT613I  DG LPort RPort Flg Queue ID Remote IP
R2 0502 BSTT613I --- ----- ----- --- ----- -- ---------------
R2 0502 BSTT613I   0   123     0 012    -1 Z2
R2 0502 BSTT011I COMMAND PROCESSING COMPLETE
```

LINK

```
LINK name adapter_no IPv6_addr netmask mtu VLAN number prty GLOBAL
```

The LINK command identifies a link from the network to the network device specified. This command must follow the DEVICE command for the device_name specified.

Adapter_no is normally 0 but may be up to 3 for devices with multiple adapters. When OSA Express 3 adapters are used, the port number being used is specified in the adapter_number field.

IPv6_addr is the presentation format IPv6 address of this network interface.

Network_mask is the network mask of the network.

Mtu specifies the MTU size to be used. CTCA and OSAX devices support an MTU size of up to 9000. While HIPR devices support an MTU of up to 57216. Specifying a value larger than the maximum supported for the device will result in the maximum being used.

If the network interface is to be connected to a VLAN trunc port the literal VLAN must be specified along with the VLAN number and priority. If a VLAN is not being used or the adapter is connected to a VLAN access port these parameters may be omitted

The GLOBAL parameter applies to OSA Express and Hipersocket interfaces using Layer 3 only. Layer 2 is not supported. Usage of this parameter requires z/VSE 5.1 and IJBOSA at APAR level DY47264/UD53715/UD53716.

LTWBUF

```
LTWBUF nnn
```

The LTWBUF command is used to specify the amount of 31-bit partition GETVIS to reserve during TCP/IP stack startup for large TCP window buffers. This command is valid under z/VSE 4.2 and higher only. The amount of storage that needs to be specified is related to the SHIFT command. The SHIFT 4/5/6/7 command indicates 1/2/4/8 MB buffers respectfully. To allocate 20 buffers, reserve 20 * buffer_size. Under z/VSE 5.1 (or higher) 64-bit virtual will be used. Usage of 64-bit virtual also requires 64-bit virtiual storage be made available with the SYSDEF SYSTEM command at IPL time.

PORT

```
PORT nnn SUSPEND
PORT nnn RESUME
PORT
```

The PORT command is used to SUSPEND or RESUME connections on a specified port. To disable connections on a port use the SUSPEND command. To re-enable connections use the RESUME command. To query suspended ports enter the PORT command with no options.

ROUTE

```
ROUTE device_name network_address network_mask gateway metric

Examples:
ROUTE OSAX600 806::0 /64 ::0 0
ROUTE OSAX600 ::0 ::0 806::1 1
```

The ROUTE command specifies a static route using the specified device_name to the network. Normally two route statements are required. The first specifies the route used by the local subnet and the second specifies the default route. The default route identifies the gateway to be used to deliver a packet to an outside network. The gateway IPv6 address must be specified even if it is null (::0).

The route metric is used to tell the stack the number of hops required to reach the specified outside network. A metric of zero (0) indicates that the network is the local subnet. A metric of one (1) or more indicates that the route is to an outside network and the IPv6 addresses on this network can only be accessed via the specified gateway IPv6 address. The default metric is zero (0).

In the example above, the first ROUTE command specifies a local subnet route. The IPv6 addresses on subnet 806::0 /64 are local. This is indicated by the metric of zero (0). The second ROUTE command specified a default route with a gateway IPv6 address of 806::1. The metric of one (1) indicates the IPv6 addresses using this route are not local.

REDIRECT

```
REDIRECT ON|OFF
```

The REDIRECT command enables (ON) or disables (OFF) ICMP Redirect processing. The default is ON.

SEGMENT

```
SEGMENT * $$ LST …
```

The SEGMENT command is used to tell the BSTT6NET command processor to issue a VSE/POWER SEGMENT request.

SHIFT

```
SHIFT 0|4|5|6|7
```

The SHIFT command is used to enable large TCP receive window support. The command SHIFT 0 is used to disable the support. This command must be part of the TCP/IP stack startup commands and once enabled the SHIFT value cannot be changed except to disable the support. The SHIFT 4|5|6|7 command specified 1|2|4|8 MB buffer usage respectfully. The default is 0 for z/VSE 4.1 (and before), 4 for z/VSE 4.2 (and higher) and 6 for z/VSE 5.1 (and higher).

TRACEIP

```
TRACEIP IPv6_address mask
```

The TRACEIP command is used to limit the output from an TRACE2 ON packet trace. The IPv6_address value is the IPv6 address or network address. The mask is the network mask. If an IPv6_address is specified use a mask of /128. The mask must be specified.

TRACEID

```
TRACEID xx
```

The TRACEID command is used to limit output from a TRACE ON API trace. Trace output is limited to a specific partition.

TRACEEZ

```
TRACEEZ xx yyyyyyyy
```

The TRACEEZ command is used to enable and disable EZASOKET API traces in a partition. The xx value is the partition ID. The yyyyyyy value indicates Y/N values for each possible trace.

EZASOKETs Trace flag settings

```
// SETPARM IPTRACE='xxxxxxx'
'Y.......' produce call parms trace information
'.Y......' produce iprpl trace information
'..Y.....' produce console messages on entry and exit
'...Y....' produce trace information on SYSLST, not direct to LST queue
'....Y...' produce waitlist trace information
'.....Y..' produce one line entry and exit messages trace information
'......Y.' Force 'Y.......' for any call which results in an error
'.......Y' Trace full SEND/RECEIVE buffer
```

USERMSS

```
USERMSS 1440
```

The USERMSS command is used to override the RFC 1122 Host Requirements minimum value for the Maximum Segment Size (MSS) used. The default is 1220. Values over 1440 should not be used without contacting BSI support.

BSTT6NET TCP/IP Stack JCL

In general the BSTT6NET JCL follows the following format.

```
// OPTION LOG,PARTDUMP,NOSYSDUMP,SADUMP=1
// LIBDEF PHASE,SEARCH=(toollib.slib)
// LIBDEF SOURCE,SEARCH=(cfglib.slib,toollib.slib)
// SETPFIX LIMIT=(256K,1100K)
// EXEC BSTT6NET,SIZE=BSTT6NET,OS390
ID nn
INTERVAL nnn
GARBAGE ON
DEVICE devname …
LINK   devname …
ROUTE  devname …
ROUTE  devname …
HOST … (1 required for each LINK statement)
ATTACH TCP/IP
/*
```

toollib.slib is the IPv6/VSE installation lib.slib

cfglib.slib is the lib.slib where the BSTTPARM.A member is stored. Normally PRD2.CONFIG.

The // EXEC statement includes the '***,OS390'*** parameter.

The // SETPFIX LIMIT should be set to (256K,1100K) for each DEVICE command.

For more information about the ROUTE command see the ROUTE command documentation.

The INTERVAL should be specified. This command enables TCP/IP Keep Alive processing and socket cleanup within BSTT6NET. The recommended INTERVAL value is 120.

Sample OSAX JCL

In VSE ADD 710:712,OSAX

```
// OPTION LOG,PARTDUMP,NOSYSDUMP,SADUMP=1
// ASSGN SYS000,SYSLST
// LIBDEF PHASE,SEARCH=(BSILIB.TTDEV)
// LIBDEF SOURCE,SEARCH=(PRD2.CONFIG,BSILIB.TTDEV)
// SETPFIX LIMIT=(256K,1100K)
// EXEC BSTT6NET,SIZE=BSTT6NET,OS390
ID 66
INTERVAL 120
*
DEVICE OSAX720 OSAX 720 IPV6TST 722
LINK   OSAX720 0 806::1:2 /64                    8192
*
ROUTE  OSAX720   806::0   /64                    ::0       0
ROUTE  OSAX720   ::0      ::0                    806::1:1  1
*
HOST ZVM54  806::1:1
HOST ZVSE42 806::1:2
HOST ZLINUX 806::1:3
HOST ROUTER 806::1:100
*
DNS 806::1:10 PRI
DNS 806::1:11 SEC
DOMAIN .bsiopti.com
*
*
ATTACH TCP/IP
/*
```

Sample Hipersocket JCL

In VSE ADD 700:702,OSAX,01

```
// OPTION LOG,PARTDUMP,NOSYSDUMP,SADUMP=1
// ASSGN SYS000,SYSLST
// LIBDEF PHASE,SEARCH=(BSILIB.TTDEV)
// LIBDEF SOURCE,SEARCH=(PRD2.CONFIG,BSILIB.TTDEV)
// SETPFIX LIMIT=(256K,1100K)
// EXEC BSTT6NET,SIZE=BSTT6NET,OS390
ID 66
INTERVAL 120
*
DEVICE OSAX700 HIPR 700 IPV6TST2 702
LINK   OSAX700 0 806::2:2 /64                    8192
*
ROUTE  OSAX700   806::0   /64                    ::0       0
ROUTE  OSAX700   ::0      ::0                    806::2:1  1
*
HOST ZVM54  806::2:1
HOST ZVSE42 806::2:2
HOST ZLINUX 806::2:3
HOST ROUTER 806::2:100
*
DNS 806::2:10 PRI
DNS 806::2:11 SEC
DOMAIN .bsiopti.com
*
*
ATTACH TCP/IP
/*
```

Sample 6IN4 JCL

In VSE ADD 700:702,OSAX,01

```
// OPTION LOG,PARTDUMP,NOSYSDUMP,SADUMP=1
// ASSGN SYS000,SYSLST
// LIBDEF PHASE,SEARCH=(BSILIB.TTDEV)
// LIBDEF SOURCE,SEARCH=(PRD2.CONFIG,BSILIB.TTDEV)
// SETPFIX LIMIT=(512K,2200K)
// EXEC BSTT6NET,SIZE=BSTT6NET,OS390
ID 66
COUPLE 00
*
INTERVAL 120
*
DEVICE TUN6IN4 6IN4 0 66.117.47.228
LINK   TUN6IN4 0 2001:4830:1600:1DB::2 /64 1500
ROUTE  TUN6IN4   2001:4830:1600:1DB::0 /64 ::0 0
*
DEVICE OSAX720 OSAX 720 IPV6TST 722
LINK   OSAX720 0 FD00:806:1::2 /48 1480
ROUTE  OSAX720   FD00:806:1::0 /48 ::0 0
*
ROUTE  TUN6IN4   ::0 ::0 2001:4830:1600:1DB::1 1
*
HOST ZVM54   FD00:806:1:0::1
HOST ZVSE42  FD00:806:1:0::2
HOST ZLINUX  FD00:806:1:0::3
HOST ZVSE42B FD00:806:1:0::4
HOST ROUTER  2001:4830:1600:1DB::1
*
DNS FD00:806:1::3 PRI
DOMAIN .ipv6test.com
*
ATTACH TCP/IP
/*
```

In this sample 6IN4 driver startup, two networks are defined. The local IPv6 network is defined with the DEVICE OSAX720 command sequence (DEVICE, LINK, ROUTE). The 6IN4 tunnel is defined with the DEVICE TUN6IN4 command sequence. Any packet with the FD00:806::/48 prefix will be routed through the OSA Express device. Any packet with another prefix will be sent out the 6IN4 tunnel. The stack ID used for the IPv4 side of the 6IN4 tunnel is specified in the COUPLE command.

The DEVICE command for a 6IN4 tunnel specifies the IPv4 address of the Tunnel Broker's Point of Presence (POP) server. The 6IN4 tunnel allows you to send and receive IPv6 packets over an IPv4 network.

Sample CTCA JCL

In VSE ADD 320:321,CTCA,EML

```
// OPTION LOG,PARTDUMP,NOSYSDUMP,SADUMP=1
// ASSGN SYS000,SYSLST
// LIBDEF PHASE,SEARCH=(BSILIB.TTDEV)
// LIBDEF SOURCE,SEARCH=(PRD2.CONFIG,BSILIB.TTDEV)
// SETPFIX LIMIT=(256K)
// EXEC BSTT6NET,SIZE=BSTT6NET
ID 77
COUPLE 00
*
INTERVAL 120
*
DEVICE CTC320  CTCA 320
LINK   CTC320 0 FD00:806:2::1 /48 8192
ROUTE  CTC320   FD00:806:2::0 /48 ::0 0
*
HOST CTC320   FD00:806:2::1
*
ATTACH TCP/IP
/*
```

Sample BSTT6NET Startup Output

```
BSTT000I INITIATED  BSTT6NET Ver 2.46 03/25/09 14.42    EP=00520078
BSTT003I COPYRIGHT (C) 1998-2009 BARNARD SOFTWARE, INC.
BSTT669I IPv6/VSE BUILD 246PRE21
BSTT004I CB=TTLA A=0052E000 L=000013FC
BSTT019I VSE 8.20 MODE 31-BIT
BSTT004I CB=COMR A=0030B4F0 L=00000108
BSTT000I INITIATED  BSTTX6PC Ver 2.46 02/12/09 07.55    EP=00831000
BSTT045I TCP/IP ID SET TO 66
BSTT000I INITIATED  BSTTISRV Ver 2.46 03/26/09 17.49    EP=0084EB80
BSTT000I INITIATED  BSTT21EP Ver 2.46 04/01/09 11.27    EP=80341390
BSTT600I     0 INITIATED  ipboot   Ver 2.46 03/26/09 18.11
BSTT613I       Tracing is Active at Startup
BSTT600I     1 INITIATED  COMMAND  Ver 2.46 02/09/09 16.11
BSTT600I     2 INITIATED  netstart Ver 2.46 03/26/09 18.10
BSTT600I     3 INITIATED  slowtmr  Ver 2.46 02/11/09 06.44
BSTT601I     0 TERMINATED ipboot
BSTT600I     4 INITIATED  ipproc   Ver 2.46 02/11/09 08.37
BSTT600I     5 INITIATED  tcptmr   Ver 2.46 02/11/09 08.37
BSTT600I     6 INITIATED  tcpinp   Ver 2.46 02/11/09 09.59
BSTT600I     7 INITIATED  tcpout   Ver 2.46 02/10/09 16.26
BSTT600I    12 INITIATED  EOTTASK  Ver 2.46 04/10/09 08.01
BSTT600I    13 INITIATED  DEVOSAX  Ver 2.46 04/01/09 14.25
BSTT605I    13 DEVICE 01 OSAX720  TYPE OSAX     UNIT 0720
BSTT014I IJBOSA   LOADED A=8185CB80 L=00009440
BSTT010I IJBOSA D82TEMP 09072 200708L
BSTT600I    14 INITIATED  icsndrs  Ver 2.46 03/24/09 08.02
BSTT613I       OSAX720  NetIF  1 IPv6 806::1:2
BSTT613I       OSAX720  NetIF  1 llIP fe80::200:0:100:8
BSTT613I TCP/IP-TOOLS TCP/IP Stack Initialization Complete
BSTT601I     2 TERMINATED netstart
BSTT601I    14 TERMINATED icsndrs
```