IPv6/VSE

Programming Guide

Current Build

Table of Contents

Preface

About this Publication

This is the **IPv6/VSE Programming Guide**. The manual will introduce you to IPv6/VSE programming. The manual is basically an extension to the IBM z/OS EZA API programming manuals and the IBM TCP/VSE for VSE Programming Guide for the Assembler Sockets API.

SC31-8788 is the z/OS Communications Server IP Sockets Application Programming Interface Guide and Reference

SC33-6766 is the TCP/IP for VSE v1.5.0 Programming Guide

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.

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

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

If the TCP/IP-TOOLS application partition stops responding to its console interface, use the **AR DUMP XX** command to obtain a dump of the partition.

EZASMI/EZASOKET

EZA Compatibility

The primary programming API for IPv6/VSE is the EZA interface as defined by the following manual:
z/OS Communications Server:
IP Sockets Application Programming Interface Guide and Reference
Document Number SC31-8788

In addition, the EZA interfaces on z/VSE have been extended to include support for the GSK SSL functions. The GSK SSL extensions were first defined in the following manual:
TCP/IP for VSE/ESA:
TCP/IP for VSE/ESA – IBM Program Setup and Supplementary Information
Document Number SC33-6601-05

The following describes the differences between IPv6/VSE and the manual mentioned above:

All function calls

ALET=

ALETs are not supported on any calls. As an IPv6/VSE extension, the ALET= parameter can be used to specify the upper 4 bytes of buffers located “Above The Bar” once the 64bit Migration Facility is enabled by using an enabling IOCTL call. Additional details can be found in this manual under “64bit Migration Facility”.

ECB=

The z/OS implementation requires that all ECBs contain a 104 byte work area (total 108 bytes) after every ECB fullword. While IPv6/VSE requires only 28 additional bytes of work area (total 32 bytes), it is recommended that 156 additional bytes (total 160 bytes) be provided for compatibility with other z/VSE vendors.

Names

All names passed to, or returned from, IPv6/VSE are limited to 80 characters.

ACCEPT

All parameters are supported.

BIND

All parameters are supported.

CANCEL

All parameters are supported.

CLOSE

All parameters are supported.

CONNECT

All parameters are supported.

FCNTL

All parameters are supported.
The following COMMAND sub-parameter functions are supported:

F_GETL
F_SETL

FREEADDRINFO

All parameters supported

GETADDRINFO

The following HINTS sub-parameters are supported:

AI_ALL
AI_NUMBERICHOST
AI_NUMBERICSERV
AI_PASSIVE
AI_V4MAPPED

The following HINTS sub-parameter is 'forced on' when using a IPv6 Assist Mode stack:

AI_V4MAPPED

IPv6/VSE Extensions:
Supports local '.LUNAME' requests

GETCLIENTID

All parameters are supported.

GETHOSTBYADDR

All parameters are supported.
NO ALIAS information is returned.
This is an IPv4 call and will return data only if IPv6/VSE is COUPLEd.

GETHOSTBYNAME

All parameters are supported.
NO ALIAS information is returned.
This is an IPv4 call and will return data only if IPv6/VSE is COUPLEd.
IPv6/VSE Extensions:
Supports local '.LUNAME' requests with the following note:

The address 255.255.255.255 is returned for Ipv6 connected terminals.

GETHOSTID

All parameters are supported.
This is an IPv4 call and will return data only if IPv6/VSE is COUPLEd. It will return the IP address of the COUPLEd stack.

GETHOSTNAME

All parameters are supported.

GETIBMOPT

All parameters are supported.

GETNAMEINFO

The following parameters are not supported:

SERVICE
SERVLEN

The following FLAGS sub-parameters are supported:

NAMEREQD

GETPEERNAME

All parameters are supported.

GETSOCKNAME

All parameters are supported.

GETSOCKOPT

All parameters are supported.
The following OPTNAME sub-parameters are supported:

SO_REUSEADD
SO_KEEPALIVE
SO_LINGER
SO_RCVBUF
SO_ERROR
SO_TYPE
SO_V6_ONLY
SO_DEBUG
SO_ACCEPTCON

GIVESOCKET

All parameters are supported.
GIVE and TAKE applications must be talking to the same stack.

GSKFREMEM

All parameters are supported.

GSKGETCIPHINF

All parameters are supported.

GSKGETDNBYLAB

All parameters are supported.

GSKINIT

All parameters are supported.

GSKSSOCCLOSE

All parameters are supported.

GSKSSOCINIT

All parameters are supported.

GSKSSOCREAD

All parameters are supported.

GSKSSOCRESET

All parameters are supported.

GSKSSOCWRITE

All parameters are supported.

GSKUNINIT

All parameters are supported.

INITAPI

All parameters except UEEXIT all supported with the following notes:
The default value for MAXSOC is 64. The maximum value for MAXSOC is 8192.
If TCPNAME is not specified, it will default to using the stack indicated by the job SYSPARM value.
TCPNAME may also contain:

1. Stack ID left justified with trailing blanks.
2. Stack ID right justified with leading blanks.
3. Stack ID in the form “SOCKETxx' where 'xx' is the stack ID.
4. Any name defined to the BSI EZA Multiplexer if the Multiplexer is being used.

APITYPE=1 is not supported.
IPv6/VSE Extensions:
ASYNC is supported, but not in the same way as z/OS. ASYNC can be used to specify the address of a user written WAIT routine. Any WAITs within the EZA interface will be passed to the specified exit routine using a BASSM. When the exit routine is entered, the following registers will be set:

R1	Pointer to the parameter list containing: 4 byte token from the INITAPI EXIT= parameter The address of a wait ECB (if high order bit on) or the address of a list of wait ECBs (if the high order bit is off).
R13	Pointer to a 72 byte save area.
R14	Return address.
R15	Address of Exit routine. If not zero on return, ERRNO will be set to 10333.

IOCTL

All parameters are supported.
The following COMMAND sub-parameter functions are supported
    FIONBIO
    FIONREAD
IPv6/VSE Extensions:
The Stack Command Interface can be called directly using a special IOCTL command.
    IOCTL_BSICOM (x'F00B5102')
        REQARG = The address of a 4 byte buffer length field
            When called, the length is the maximum size of returned data allowed
        RETARG = Buffer
            When called, contains the stack command as a null-terminated string
            Upon return, contains the stack response
        RETCODE
            Upon return (if not negative) contains the length of data returned.
The 64bit Migration Facility is enabled or disabled use the following command:
    IOCTL_BSIBUF64 (x'F00B5103')
        REQARG = X'00000000' disables the facility
        REQARG = X'00000001' enables the facility
        RETARG is not used
    This request will fail with error 10154 if z/VSE does not support 64bit addressing.
Large Buffer Support is enabled or disabled use the following command:
    IOCTL_BSILRGBUF (x'F00B5104')
        REQARG = X'00000000' disables the support
        REQARG = X'00000001' enables the support
        RETARG is not used

LISTEN

All parameters except BACKLOG supported.
A UDP connection requires that a BIND be issued prior to the LISTEN.

NTOP

All parameters are supported.

PTON

All parameters are supported.

READ

All parameters are supported.

READV

All parameters are supported.

RECV

All parameters are supported.
The following FLAGS sub-parameters are supported:
MSG_PEEK

RECVFROM

All parameters are supported.
The following FLAGS sub-parameters are supported:
MSG_PEEK

RECVMSG

The RECVMSG function is not supported.

SELECT

All parameters are supported.

SELECTEX

All parameters are supported.
A maximum of 255 ECBs can be specified in a SELECB list.

SEND

All parameters except FLAGS area supported.

SENDMSG

The SENDMSG function is not supported.

SENDTO

All parameters except FLAGS supported.
Any FLAGS values are ignored.

SETSOCKOPT

All parameters are supported.
The following OPTNAME sub-parameters are supported:
SO_KEEPALIVE (Allowed but Ignored)
SO_LINGER (Allowed but Ignored – Always on)
SO_REUSEADR
SO_DEBUG
SO_V6_ONLY

SHUTDOWN

All parameters are supported.
HOW=2 is required.

SOCKET

All parameters are supported.

TAKESOCKET

All parameters are supported.

TASK

All parameters are supported.

TERMAPI

All parameters are supported.

WRITE

All parameters are supported.

WRITEV

All parameters are supported.
IVCNT can have a maximum value of 120

IPv6/VSE Extensions:

Local ".LUNAME" Requests

When using the IPv6/VSE TELNET Server, the IP address of the terminal can be extracted by requesting a name lookup (GETHOSTBYNAME or GETADDRINFO) using a special hostname in the format "luname.LUNAME". For example, to determine the IP address for a terminal with the LUNAME of "T010", issue a name lookup using "T010.LUNAME". Please note that if the terminal name lookup fails, a normal DNS lookup is performed.

64bit Migration Facility

As defined by IBM, the EZASMI interface only supports 31bit addresses. This limit prevented the exploitation of 64bit data areas introduced in z/VSE 5.1. The 64bit Migration Facility provides a method to use 64bit areas within the limits imposed by the EZASMI macro as provided by IBM. 64bit addresses are passed to IPv6/VSE by splitting the 64bit address into two 32 bit numbers. The high-half of the address is passed using the ALET= parameter and the low-half 32 bits are passed in the BUF= parameter. This facility must first be enabled using the IOCTL call BSI_BUF64 as documented in this manual under IOCTL.

The maximum amount of data that can be sent in a single send request is 2GB-1 (or X'7FFFFFFF') bytes. The actual maximum amount of data that can be received in a single receive request is 4MB. This amount is dependent on the SHIFT value in use by the BSTTINET/BSTT6NET TCP/IP stack.

EZASMI Considerations FOR COUPLEd Stacks

Since the EZASMI interface is IP version neutral, there is no “IPv4 only API”. The type of address passed is based on the value of the FAMILY field within the NAME structure. (This document will use “NAME.FAMILY” when discussing this field.)

The following assumes that the IPv6 stack has been properly coupled to an active IPv4 stack using the IPv6/VSE “COUPLE” command.

NAME.FAMILY=AF_INET => IPv4 Stack

All requests for this socket are passed to the IPv4 stack.

NAME.FAMILY=AF_INET6 => IPv4 Stack

The address is inspected for an “IPv4 mapped as IPv6” address. If true, the request is passed to the IPv4 stack. If the address is not an “IPv4 mapped as IPv6” address, the request is terminated with ERRNO set to 00046 (EPFNOSUPPORT).

NAME.FAMILY=AF_INET => IPv6 Stack

All requests for this socket are passed directly to the IPv4 stack.

NAME.FAMILY=AF_INET6 => IPv6 Stack

The address is inspected for an “IPv4 mapped as IPv6” address. If true, the request is passed to the IPv4 stack. If the address is not an “IPv4 mapped as IPv6” address, the request is passed to the IPv6 stack.

INITAPI

If an IPv6 stack is requested, the name of the coupled IPv4 stack is validated during the INITAPI call. If an IPv4 stack is not coupled or if the coupled IPv4 stack is not available at this time, AF_INET and “IPv4 mapped as IPv6” addresses will be rejected.

GETADDRINFO

If using an IPv4 primary stack, the returned RES information will contain only a single IPv4 addresses. If using an IPv6 stack, IPv6/VSE obeys the HINTS flags AI_ALL and AI_V4MAPPED.

GETHOSTBYNAME/GETHOSTBYADDR

These are IPv4 only calls. If an IPv4 stack is not available, they will fail with ERRNO set to 10218.

GETHOSTID

This is an IPv4 only call. If an IPv4 stack is not available, the address returned in the RETCODE will be low-values (x‘00000000’). Currently, this is the only function available to the application to identify the IPv4 home address of a coupled stack.

GETHOSTNAME

Returns the name of the primary stack only.

GETIBMOPT

The second byte of each IMAGE entry will contain either a x'01' or x'02' indicating that the stack is either IPv4 or IPv6.

GIVESOCKET

The GIVESOCKET is handled by the same stack that is processing other requests for the socket, not necessarily the primary stack.

LISTEN

Passive connections are handled depending on the SETSOCKOPT option IPV6_V6ONLY. If this option is not set, while using an IPv6 primary stack, the interface will listen on the IPv6 stack and, if available, the IPv4 stack also.

TAKESOCKET

Normally a child program must use the same stack ID as the GIVESOCKET program. This rule is relaxed under the following strict conditions: If the original parent application specified an IPv6 primary stack, but was using NAME.FAMILY = AF_INET, a child application can successfully take the socket using only an IPv4 primary stack but only if the child’s primary stack ID is the same as the parent’s coupled IPv4 stack ID.

EZA Traces

Setting Trace Options

Trace options may be set using any of the following methods:

JCL:	// SETPARM IPTRACE='yyyyyyyy'
Stack Command:	TRACEEZ xx yyyyyyyy
EZA CALL	SETSOCKOPT.OPTNAME=SO_DEBUG

Trace Options

There are 8 trace option flags available. Each trace option flag is indicated by the positional 'N' or 'Y' in the IPTRACE or TRACEEZ setting. (SETSOCKOPT.OPTNAME=SO_DEBUG will always use a setting of 'YNNYNNNN' for 'ON' or 'NNNNNNNN' for 'OFF'.) The 8 trace flags are:

```
'Y.......' produce base EZA trace information
'.Y......' produce trace information for BSI internal control blocks
'..Y.....' produce console messages on entry and exit to the EZA interface
'...Y....' produce trace information on SYSLST, not direct to LST queue
'....Y...' produce trace information for internal WAIT lists
'.....Y..' produce one line entry and exit trace messages
'......Y.' Force 'Y.......' for any call which results in an error
'.......Y' Trace full SEND/RECEIVE buffer
```

All flags can be used in combinations.

The default is to write all trace information (except console traces '..Y....') directly the VSE/POWER LST queue with a job name of 'EZALOGxx' (xx is the partition identifier). Many customers prefer to always use the '...Y....' flag to intersperse the trace output with normal or debug program output. Also, with heavy processing, the XPCC communication can slow the IP processing to the point that time based failures occur. Should this occur, use the '...Y...' flag as it is much faster to write to partition SYSLST. If writing to SYSLST, the trace will be interspersed with normal job output messages and may actually cause an overprint of the program output as the tracing function uses "write before advancing one line' commands.

JCL SETPARM Options

The following JCL SETPARM options are available when using the EZASMI/EZASOCKET interfaces.

SVABUF

// SETPARM SVABUF=YES indicates that the buffers are located in the SVA.

SENDALL

// SETPARM SENDALL=YES instructs the system to not return unless all data has been queued by the stack to be sent.

LOCALGT

// SETPARM LOCALGT=YES instructs the system to handle all GIVESOCKET/TAKESOCKETS within the partition and avoid the overhead of communicating with the stack partition. This option restricts all give/takes to within the same partition.

POST

// SETPARM POST=PARTITION instructs the stack to post not just the calling task, but the calling task's partition at the end of each call. This should be used only at the direction of the BSI staff. This SETPARM is no longer necessary when using z/VSE 4.2 (or higher).

TESTPSN

// SETPARM TESTPSN=NO instructs the stack to not validate the PSN field when posting requests. This should ONLY be used at the direction of the BSI staff.

LRGBUF

// SETPARM LRGBUF=YES instructs the stack to use 'Large Buffer Support' for this connection. This feature should only be used for high-volume data transfers. This feature can also be set/overridden at the socket level using the an IOCTL call.

EZA Sample Source Code Members

BSI provides many sample COBOL programs that utilized the EZASOKET interface. Positional characters in the program name indicate it's use. Sample names are in the form “BSTTEZxy.C”.

The values for “x” are:

- “0” Batch IPv4 Listener/Client/Child sample programs
- “1” Online IPv4 Listener/Client/Child/utility sample programs
- “2” Online IPv4 FTP Client sample programs
- “5” Batch IPv6 Enabled Listener/Client/Child/utility sample programs
- “6” Online IPv6 Enabled Listener/Client/Child/utility sample programs
- “A” Sample copybooks used by the sample programs (listed below)

The values for “y” are:

- “0” Sample PLT program to start the listener
- “1” Sample Listener program
- “2” Sample Client program (uses hard-coded IP address)
- “3” Sample Child program
- “5” Sample Client program (uses DNS look-up for IP address)

Sample copybooks:

“BSTTEZA”	Main Working-Storage copybook
“BSTTEZAP”	Main Procedure copybook (contains usage instructions)
“BSTTEZA6”	Working-Storage copybook for EZACIC06
“BSTTEZA8”	Working-Storage copybook for EZACIC08
“BSTTEZA9”	Working-Storage copybook for EZACIC09
“BSTTEZAC”	Working-Storage copybook used by the BSTTPREP process when converting EXEC TCP programs to EZASOKET.

BSI also includes the following additional source members that are “useful” for EZA programming:

“BSTTEZSP.A”	Allows COBOL for VSE programs to set “USAGE IS POINTER” Working-Storage fields to the address of any other Working-Storage field. This overcomes a compatibility issue between COBOL for z/OS and COBOL for VSE that hinders the use of GETADDRINFO. This routine is also provided pre-Assembled as “BSTTEZSP.OBJ”.
“EZACICTR.A”	Generates a local translation program (like EZACIC04/EZACIC05). See member for additional information. (©IBM) (Ported from z/OS)
“EZASMI.A”	IPv6 enabled EZASMI (©IBM) macro. (For pre-VSE 4.2.2 systems.)
“EZASMIx.A”	Macros used by EZASMI (©IBM) macro. (For pre-VSE 2.5 systems.)

All sample programs are provided “AS-IS” as an aid to our customers. It is the responsibility of the customer to determine the usefulness and to validate the correct processing of any fo these sample programs. BSI will provide support for sample programs only “as time allows”.

IPv6 Enabled ASM SOCKET API

The ASM SOCKET macro programming API is still available for use with IPv6. In fact, with some very simple and easy changes, ASM SOCKET macro applications can easily support both IPv4 and IPv6 interfaces.

When using the ASM SOCKET API with an IPv4 TCP/IP stack the IPv4 address is 4 bytes and fits into a single fullword of storage or a register. Simply stated, when using the ASM SOCKET API for IPv6 communications the single fullword or register contains the address of a 30 byte IPv6 Socket Address Structure (SAS).

IPv6-X Flag

The IPv6-X Flag indicates to the ASM Socket API that IPv6 Extensions to the ASM SOCKET API are in use. The flag is located in the first byte of the SRBLOK. The first field in an SRBLOK is a z/VSE ECB field. The SRBLOK is specified by the ECB= parameter in the SOCKET macro. Setting the first byte of the SRBLOK to “6” (x'F6') during the OPEN call indicates that the IPv6 Extensions to the ASM SOCKET API are being used AND that IPv6 extensions will continue to be used for this socket. It is not necessary to set the IPv6-X flag on any other type of ASM SOCKET request. This flag is set only on ASM SOCKET OPEN calls.

IPv6 Enabled Applications

The Assembler SOCKET API has been extended to support IPv6 applications. When using the IPv6 extensions to the ASM SOCKET API, the program **must** indicate this during **every** OPEN request by setting the IPv6-X Flag. With the IPv6-X Flag set, any IP address being passed to the API is passed within a Socket Address Structure (SAS). The description of the SAS varies between IPv4 and IPv6 IP addresses but is defined as follows:

```
       * IPv4 socket address structure.
        01 NAME.
           03 FAMILY         PIC 9(4) BINARY   VALUE +2.
           03 RESERVED       PIC X(2) VALUE LOW-VALUES.
           03 IP-ADDRESS     PIC 9(9) BINARY.
           03 RESERVED       PIC X(8) VALUE LOW-VALUES.

       * IPv6 socket address structure.
         01 NAME.
            03 FAMILY        PIC 9(4) BINARY   VALUE +19.
            03 RESERVED      PIC X(2) VALUE LOW-VALUES.
            03 RESERVED      PIC X(4) VALUE LOW-VALUES.
            03 IP-ADDRESS    PIC X(16).
            03 RESERVED      PIC X(4) VALUE LOW-VALUES.
```

When using the IPv6 Extensions to the ASM SOCKET API, the contents of the field specified in the ASM SOCKET FOIP= parameter will not contain an IP address, instead the ASM SOCKET FOIP= parameter will contain a pointer to an SAS (Socket Address Structure).

ASM SOCKET Macro Changes

The following table shows the changes made to the ASM SOCKET macro API.

Macro	*Description*
OPEN	For IPv6, the FOIP= parameter will be the address of a Socket Address Structure (SAS).
	For IPv6, SRFOIP will contain the address of a 16 byte IPv6 address.
SEND	For IPv6, SRROIP will contain the address of a 16 byte IPv6 address and the
	For IPv6, the FOIP= parameter (if specified) will be the address of a Socket Address Structure (SAS).
RECEIVE	For IPv6, SRFOIP will contain the address of a 16 byte IPv6 address.
CLOSE	No changes

ASM SOCKET CONTROL Call Changes

The following table shows the changes made to the ASM SOCKET CONTROL calls.

Macro		*Description*
SEND	GETHOSTBYADDR	For IPv6, expects up to 45 bytes containing the presentation format IPv6 address terminated by x'15'.
RECEIVE	GETHOSTBYNAME	For IPv6, returns a 16 bytes binary IPv6 address followed by up to 45 bytes containing the presentation format IPv6 address.
RECEIVE	GETHOSTID	For IPv6, returns a 16 bytes binary IPv6 address followed by up to 45 bytes containing the presentation format IPv6 address.
RECEIVE	GETVENDORINFO	For IPv6, returns 'BSIIPV6'
RECEIVE	GETVENDORINFO	For IPv4, returns 'BSIIPv4'

ASM SOCKET New CONTROL Calls

The following table shows the new ASM SOCKET CONTROL calls. NTOP refers to Network-to-Presentation, PTON refers to Presentation-to-Network calls. GETPORTBYNAME converts Service names to numeric Port numbers. These calls can be used to simplify your application code as local routines to convert IP addresses or Port Names from network to presentation and back again are no longer necessary.

Macro		***Description***
SEND	GETPORTBYNAME	Expects a presentation format service name.
RECEIVE	GETPORTBYNAME	Returns a “9(9) COMP” (fullword) port number.
SEND	NTOP	Expects a Socket Address Structure (SAS)
RECEIVE	NTOP	Returns a presentation format IP address terminated by x'15'
SEND	PTON	Expects a presentation format IP address terminated by x'15'
RECEIVE	PTON	Returns a Socket Address Structure (SAS)

Determining the type of Stack

IPv4 API => IPv4 Stack

In this situation, all calls to the API are treated as IPv4 calls. All addresses passed to the API are considered 4 byte IPv4 addresses. All addresses returned are 4 byte IPv4 addresses.

IPv6 API => IPv4 Stack

NAME.FAMILY = AF_INET

All requests for this socket are passed to the IPv4 stack.

NAME.FAMILY=AF_INET6

The address is inspected for an “IPv4 mapped as IPv6” address. If true, the request is passed to the IPv4 stack.

If the address is not an “IPv4 mapped as IPv6” address, the request terminates with SRCODE set to 4 (SROPFAIL).

IPv6 API => IPv6 Stack

NAME.FAMILY = AF_INET

All requests for this socket are passed to the IPv4 stack.

NAME.FAMILY=AF_INET6

The address is inspected for an “IPv4 mapped as IPv6” address. If true, the request is passed to the IPv4 stack. If the address is not an “IPv4 mapped as IPv6” address, the request is passed to the IPv6 stack.

ADDITIONAL SOCKET CONSIDERATIONS

OPEN

During an OPEN, the first byte of the the ECB= area is checked for the IPv6-X Flag. If found, the socket request will be processed using the IPv6 extensions to the ASM SOCKET API.

OPEN PASSIVE=YES (IPv6-X set)

During the OPEN process, the contents of the FAMILY field in the SAS is inspected. When set to AF_INET (+2), the OPEN will listen using the Coupled IPv4 stack (if available). When set to AF_INET6 (+19), the OPEN will listen on the Ipv6 stack (if available). Any other value indicates that the OPEN should listen on the Primary stack.

An application wishing to listen on multiple stacks (i.e., both IPv4 and IPv6) will need to issue two separate opens. One OPEN for IPv6 connections, the other for IPv4 connections. While only one socket needs to be opened using the IPv6-X Flag, it may be easier to always use the IPv6-X Flag for both OPENs and use IPv4 mapped as IPv6 addresses for IPv4 connections.

SEND CONTROL GETHOSTBYADDR (IPv6-X set)

Expects up to 45 bytes containing the presentation format IP address. The presentation format IP address is inspected to see if it contains an IPv4 address, an IPv6 address, or an “IPv4 mapped as IPv6” address. The request will be passed to the correct stack.

STATUS (IPv6-X set)

Neither CCLOIP nor CCFOIP will contain IP addresses, but will instead contain pointers to the memory addresses containing the actual 16 byte IPv6 addresses.

RECEIVE CONTROL GETHOSTBYNAME (IPv6-X set)

Returns a buffer containing a 16 byte IP address followed by up to 45 bytes containing the presentation format IPv6 address.

RECEIVE CONTROL GETHOSTID (IPv6-X set)

Returns a buffer containing a 16 byte IP address.

New CONTROL Calls

Several new CONTROL calls have been added. They are described below:

GETPORTBYNAME

This control call ignores the IPv6-X Flag. It will convert a Port Name in presentation format into a valid port number.
INPUT: Up to 45 bytes of text terminated by x'15'.
OUTPUT: A port number in the range 1-65535 as a fullword (“PIC S9(9) COMP”).
A returned value of zero (0) indicates that the service name was not found.

GETVENDORINFO

This control call ignores the IPv6-X Flag. It will return the vendor defined information about the Primary Stack. The ASM SOCKET OPEN,CONTROL macro this request must specify NEGOT=NO parameter.
INPUT: None
OUTPUT: Either “**BSIIPV4**” or “**BSIIPV6**”.

GETHOSTBYLUNAME

This control call ignores the IPv6-X Flag. This call will return the IPv6 (or IPv4 mapped IPv6 address) of the host using an LUNAME.
INPUT: 'GETHOSTBYLUNAME' + ' ' + LUANME + x'15'
OUTPUT: buffer will have 16 byte IPv6 address. If IPv4 it will be a 16 byte v4 mapped address.

NTOP

This control call ignores the IPv6-X Flag. It will format the address found in a passed SAS into the correct presentation format.
INPUT: A Socket Address Structure (SAS)
OUTPUT: Up to 45 bytes of text terminated by x'15'.

PTON

This control call ignores the IPv6-X Flag. It will convert a IP address in presentation format into a valid IPv6 address.
INPUT: Up to 45 bytes of text terminated by x'15'.
OUTPUT: A Socket Address Structure (SAS).

The general format of the Control Call text input is a command, a space, an operand and a new line (x'15') character. When an operand is not present, the space should be omitted. When either the operand is not present, or the operand is of a known fixed length, the new line character is not required by IPv6/VSE. For compatibility with other vendors, it is recommended that the new line character should always be provided.

Control Calls Supported

The following is a complete list of all control calls supported by IPv6/VSE.

DEFINEPRODID

GETHOSTBYADDR

GETHOSTBYLU

GETHOSTBYNAME

GETHOSTID

GETHOSTNAME

GETPORTBYNAME

GETVENDORINFO

GIVESOCKET

NTOP

PTON

TAKESOCKET

VERIFYPRODKEY

Sample Programming

```
           MVC   IPFLAG,VENDBUF+6  SET IP FLAG
*
           MVC   IPADDR,HOSTBUF    SET IPv4 Address
           MVC   IPADDR6,HOSTBUF   SET IPv6 Address
           L     R9,IPADDR         r9 ← IPv4 address
           CLI   IPFLAG,IPV6A      IPV6?
           BNE   IPV4              No.
           LA    R1,IPADDR6        R1 ← A(IPv6)
           ST    R1,IPADDR         SET ADDR of IPv6 Address
           MVC   IPSAFM,=H'19'     SET SAS FAMILY
           MVC   IPSAIP,IPADDR6    SET IPV6 ADDRESS
           MVI   WKECB,C'6'        SET IPv6 enabled flag
           LA    R9,IPSAS          r9 ← SAS
IPV4       DS    0H
*
           SOCKET OPEN,TCP,
                 LOCAL=NO,
                 ACTIVE=YES,
                 PASSIVE=NO,
                 FOPORT=WKPORT,
                 FOIP=(9),
                 DESC=WKDESC,
                 ECB=WKECB
*
...
*
WKPORT     DC    H'1642'        Port number
IPADDR     DC    F'0'           IPv4 address
IPADDR6    DC    16X'00'        IPv6 address
IPSAS      DC    0XL40'00'     SAS
IPSAFM     DC    H'0'             Family
           DC    H'0'             Port
           DC    XL4'00'          ...
IPSAIP     DC    XL16'00'         IPv6 address
           DC    XL16'00'         ..
*
WKDESC     DC    F'0'           Socket Descriptor
WKECB      DC    14F'0'         Socket ECB
*
IPFLAG     DC    X'00'          IP Flag
IPV6A      EQU   C'6'             IPv6 Active
IPV4A      EQU   C'4'             IPv4 Active
*
           DS    0D
VENDBUF    DC    XL8'00'        GETVENDORINFO Buffer
HOSTBUF    DC    Xl64'00'       GETHOSTID Buffer
```

ASM SOCKET Traces

Setting Trace Options

Trace options may be set using any of the following methods:
JCL: // SETPARM IPTRACE='yyyyyyyy'
Stack Command: TRACEEZ xx yyyyyyyy

Trace Options

For compatibility with the EZA interface, there are 8 trace option flags available, but only 3 are referenced by the ASM SOCKET interface. Each trace option flag is indicated by the positional 'N' or 'Y' in the IPTRACE or TRACEEZ setting. The 3 trace flags available to the ASM SOCKET interface are:

```
'Y.......' produce base ASM SOCKET trace information
'.Y......' produce trace information for BSI internal control blocks
'...Y....' produce trace information on SYSLST, not direct to LST queue
'.......Y' Trace full SEND/RECEIVE buffer
```

All flags can be used in combinations.

The default is to write all trace information (except console traces '..Y....') directly the VSE/POWER LST queue with a job name of 'EZALOGxx' (xx is the partition identifier). Many customers prefer to always use the '...Y....' flag to intersperse the trace output with normal or debug program output. Also, with heavy processing, the XPCC communication can slow the IP processing to the point that time based failures occur. Should this occur, use the '...Y...' flag as it is much faster to write to partition SYSLST. If writing to SYSLST, the trace will be interspersed with normal job output messages and may actually cause an overprint of the program output as the tracing function uses "write before advancing one line' commands.

JCL SETPARM Options

The following JCL SETPARM options are available when using the ASM SOCKET interface.

SVABUF

// SETPARM SVABUF=YES indicates that the buffers are located in the SVA.

POST

// SETPARM POST=PARTITION instructs the stack to post not just the calling task, but the calling task's partition at the end of each call. This should be used only at the direction of the BSI staff. This SETPARM is no longer necessary when running z/VSE 4.2 (or higher).

TESTPSN

// SETPARM TESTPSN=NO instructs the stack to not validate the PSN field when posting requests. This should ONLY be used at the direction of the BSI staff.

LRGBUF

// SETPARM LRGBUF=YES instructs the stack to use 'Large Buffer Support' for this connection. This feature should only be used for high-volume data transfers.

64Bit Enabled ASM SOCKET API

IPv6/VSE's 64Bit Enabled ASM SOCKET API allows 64-bit virtual storage to be used for TCP SEND and RECEIVE requests. Usage of 64-bit virtual storage is supported *only* for SOCKET SEND,TCP and SOCKET RECEIVE,TCP requests.

If the application is running under z/VSE 5.1 (or higher) and the buffer address specified in the SOCKET SEND,TCP or SOCKET RECEIVE,TCP request has the high-bit on then the buffer address passed to the SOCKET macro is the address of a 64-bit address.

The maximum amount of data that can be sent in a single SOCKET SEND,TCP request is 2GB-1 (or X'7FFFFFFF') bytes. The actual maximum amount of data that can be received in a single SOCKET RECEIVE,TCP request is 4MB. This amount is dependent on the SHIFT value in use by the BSTTINET/BSTT6NET TCP/IP stack.

Sample Programming

```
*
         LA    R1,IARVWK                R1 <- MACRO WORK AREA
         IARV64 REQUEST=GETSTOR,
                FPROT=NO,
                COND=YES,
                SEGMENTS=BIGLEN,         INPUT
                ORIGIN=BIGBUF,           OUTPUT
                MF=(E,(1))
          LTR   R15,R15                  GOOD COMPLETION?
          BZ    *+4+2                    NO.
          DC    H'0'
*
          LA    R0,BIGBUF                R0 <- A(64-BIT BUFFER PTR)
          ST    R0,RECADR                SET ADDRESS
          OI    RECADR,X'80'             SET 64-BIT PTR FLAG
          L     R0,BIGLEN+4
          SLL   R0,20
          ST    R0,RECLEN
          MVC   BIGSAV,BIGBUF            SAVE BUFFER ADDRESS
*
...
*
RECADR    DS    A(0)
RECLEN    DS    F'0'
          DS    0D
          DC    CL8'BIG PTR '
BIGBUF    DC    AD(0)
BIGSAV    DC    AD(0)
          DC    CL8'BIG LEN '
BIGLEN    DC    AD(1)
          DC    C'IARVWK  '
IARVWK    DC    128X'00'
```

Converting CSI EXEC TCP Applications to EZASOKET

BSTTPREP.PROC is a migration program used to convert COBOL source from the CSI/IBM EXEC TCP API to the EZASOKET socket interface. BSTTPREP is supplied as a VSE REXX PROC. BSTTPREP is intended only as a one-time conversion program.

BSTTPREP can be run in VSE using the following JCL:

```
// EXEC REXX=BSTTPREP,SIZE=256K
<COBOL source goes here>
/*
```

VSE output is directed to SYSPCH.

BSTTPREP can also be run under VM (after transporting to VM via a LIBR PUNCH job) using the following:

BSTTPREP filename filetype filemode

VM output is a new CMS file on the “A” disk with the same filename.
‘BSI’ will be appended to the filetype.

BSTTPREP will convert all “EXEC TCP” statements found in the PROCEDURE SECTION to calls to the EZASOKET call interface. BSTTPREP will also partially convert all “EXEC TELNET” statements to a combination of EZASOKET calls and calls to the data translation routines “EZACIC04” and “EZACIC05”. With “EXEC TELNET” statements, the programmer should review the code as data translation is only one of the differences between the “EXEC TELNET” and the standard “EXEC TCP” calls.

BSTTPREP will NOT convert any CSI “EXEC UDP”, “EXEC FTP”, “EXEC CLIENT”, “EXEC CONTROL”, or “EXEC RAW” statements. As a note, the EZASOKET interface does support, but in a very different manner, most of the functions for which the “EXEC CONTROL” API was used. The EZASOKET interface also supports UDP functions, but due to the complexities of using UDP connections, it was considered outside the scope of BSTTPREP.

Upon processing the program source, BSTTPREP will place the literal “BSI” in columns 1-3 of any source line that was modified. Additionally, BSTTPREP will place “BSI-W’ or “BSI-E” in columns 1-5 of any statements that are warnings or error messages. In those cases, the message will be in columns 8-72. Warning messages will contain a “*” in column 7 so that the program will compile, but error messages will not so as to cause a compile error.

When the program being migrated contains multiple instances of “EXEC TCP OPEN” and “EXEC TCP CLOSE” commands, the code should be reviewed in these areas. A single program should only do

one “INITAPI” and “TERMAPI” call to the EZASOKET interface. BSTTPREP will insert multiples of these function calls, as it does not understand the program flow enough to know which open would be executed first and which close would be executed last.