OpenEdition (UNIX System Services) MVS Integrated Sockets

This chapter contains information about Cisco IOS for S/390 compatibility with OpenEdition (UNIX System Services) MVS Integrated Sockets. It includes these sections:

- Introduction

  Provides a short overview of OpenEdition (UNIX System Services) Socket support.

- References

  Lists reference material.

- Installation Considerations

  Describes installation steps specific for OpenEdition (UNIX System Services) socket support.

- Configuration Information

  Describes configuration changes specific to OpenEdition (UNIX System Services) socket support.

- Additional Socket Files

  Describes the additional files included in the SAMP data set.

- Additional Socket Call Parameters

  Describes options for ioctl(), getsockopt() and setsockopt().

- Resolving Names and Addresses

  Describes how to do host name and address resolution for OpenEdition (UNIX System Services) sockets.

- TSO Commands

  Describes CONVXL8, LOADXL8 commands.

- Debugging Information

  Describes how to get control block information.

- Stopping and Starting Sockets

  Describes how to start and stop the socket API.

- Limitations

  Describes the limitations of OpenEdition (UNIX System Services) sockets.

- Release Information

  Describes support for OpenEdition (UNIX System Services) by MVS releases.

- Common INET Support

  Describes the use, advantages, and disadvantages of Common INET support

Introduction

Cisco IOS for S/390 sockets support the OpenEdition (UNIX System Services) Integrated sockets as provided in IBM MVS/ESA. The first release of OpenEdition was part of MVS/ESA 4.3. OpenEdition Socket support was introduced with MVS/ESA 5.1 and above. Review your IBM documentation to determine OpenEdition (UNIX System Services) support for your site.

References

This document provides information on Cisco IOS for S/390 socket compatibility with OpenEdition (UNIX System Services) Integrated Sockets. It is not intended as a reference for IBM OpenEdition (UNIX System Services) MVS Integrated Sockets.

For more information on OpenEdition sockets, check the following references:

AD/Cycle C/370 Library Reference: OpenEdition MVS Sockets (IBM Manual number SC23-3024-xx (MVS/ESA 5.1 version))

IEEE POSIX Standard 1003.1g, Protocol Independent Interfaces (ISO/IEC JTC 1/SC22/WG15N, P1003.1g/Draft 6.0 (January 1995), Information Technology - POSIX - Part xx: Protocol Independent Interfaces)

X/Open CAE Specification, Networking Services, Issue 4 (X/Open Document Number C438)

C/C++ for MVS/ESA V3.1 C/MVS Library Reference: OpenEdition MVS Sockets (IBM Manual number SC23-3875-xx)

MVS/ESA Application Development Reference: Assembler Callable Services for OpenEdition MVS (IBM manual number SC23-3020-01 (MVS/ESA 5.2 version))

MVS/ESA Application Development Reference: Assembler Callable Services for OpenEdition MVS (IBM manual number SC23-3020-02 (MVS/ESA 5.2.2 version))

MVS/ESA OpenEdition MVS File System Interface Reference (IBM manual number SC23-3802-00 (MVS/ESA 5.1 and 5.2 versions))

MVS/ESA OpenEdition MVS File System Interface Reference (IBM manual number SC23-3802-01 (MVS/ESA 5.2.2 version))

Installation Considerations

The OpenEdition (UNIX System Services) Physical File System (PFS) and the Socket API are both installed automatically.

OpenEdition PFS Transport Driver load modules are installed in the PFSLOAD partitioned data set.This data set must be APF authorized and must be in either the STEPLIB concatenation for OpenEdition MVS or be part of the system link list concatenation.

Caution The Cisco IOS for S/390 address space must be defined to your security package (such as RACF, ACF2, etc.) with a valid OpenEdition MVS security segment.

The OpenEdition MVS procedure must be stopped and restarted after Cisco IOS for S/390 is installed and after the PFSLOAD data set is added to the OpenEdition MVS PROC or the system STEPLIB. OpenEdition must be configured prior to the restart.

Configuration Information

Configuration information for Cisco IOS for S/390 OpenEdition (UNIX System Services) Integrated sockets is presented in the *Cisco IOS for S/390 Customization Guide*.

Additional Socket Files

Included in the SAMP data set, there are two members needed for OpenEdition (UNIX System Services) socket support.The header include file icssckt.h includes defines for OpenEdition socket function specific to the Cisco IOS for S/390 socket implementation.There is also an equivalent assembler macro, ICSSCKTM.

You will need to move these files to your own libraries to use them.

Caution Use of definitions in these files may make socket applications binary incompatible with the IBM TCP/IP or IBM Any/Net socket implementations.

Additional Socket Call Parameters

In addition to the socket call parameters described in Chapter 3, Socket Library Functions, the OpenEdition (UNIX System Services) Integrated sockets support additional parameters for several socket calls.

ioctl() Parameters

The following additional parameters for the ioctl() call are supported for OpenEdition (UNIX System Services) Integrated Sockets users.

FIOGETOWN	Get the process or process group ID specified to receive signals. This value is type int.
FIOSETOWN	Set the process or process group ID specified to receive signals. This value is type int.
SIOCADDRT2	Adds a Cisco IOS for S/390 format route entry.
SIOCDELRT2	Deletes a Cisco IOS for S/390 format route entry.
SIOGIFHWADDR	Get the hardware address. Uses the if_req struct.
SIOCGIFNUM	Get the number of interfaces. Uses the if_req struct
SIOCGIFMTU	Get the interface MTU (Maximum Transmission Unit). Uses the if_req struct.
SIOCGPGRP	Synonym for FIOGETOWN. This value is type int. This parameter is defined in OpenEdition (UNIX System Services) MVS 5.2.2.

SIOCSIFMETRIC	Sets the network interface routing metric. This value is type int. Uses the if_req struct.
SIOCSPGRP	Synonym for FIOSETOWN. This value is type int. This parameter is defined in OpenEdition (UNIX System Services) MVS 5.2.2.

Note The SECIGET command will not be supported since it is valid only in the AF_UNIX domain.

The FIOASYNC command documented in 1003.1g will not be supported. This command enables signal-driven I/O and cannot be reasonably implemented until it is supported by OpenEdition (UNIX System Services) Sockets.

getsockopt() Parameters

The following additional parameters for the getsockopt () call are supported for OpenEdition (UNIX System Services) Integrated Sockets users.

Socket-Level Options

Socket-Level options for getsockopt():

SO_ACCEPTCONN	Reports whether socket listening is enabled (in other words listen() has been issued).
SO_ERROR	Reports information about error status and clears it.
SO_KEEPALIVE	Reports whether connections are to be kept open with periodic transmissions of messages. Use TCP_KEEPALIVE to report the current interval between packets.
SO_RCVLOWAT	Reports minimum number of bytes to process for socket input operations.
SO_RCVTIMEO	Reports timeout value for socket input operations. The timeout value is valid only for MVS 5.2.2 and above.
SO_REUSEADDR	Reports whether the rules used in validating addresses supplied to bind() should allow reuse of local addresses.
SO_SNDLOWAT	Reports minimum number of bytes to process for socket output operations.
SO_SNDTIMEO	Reports timeout for output operations.
SO_TYPE	Reports socket type.

TCP-Level Options

TCP-Level options for getsockopt():

TCP_KEEPALIVE	Reports the current time interval (in seconds) between keepalive packets. This parameter is given as type int. The option SO_KEEPALIVE must be enabled first.

UDP-Level Options

UDP-Level options for getsockopt():

UDP_CHECKSUM	Reports whether UDP checksum computation is to be performed. This parameter is given in type int.

IP-Level Options

IP-Level options for getsockopt():

IP_HDRINCL	Reports whether, for a SOCK_RAW socket, the complete IP header will be included with the data on send operations.
IP_OPTIONS	Reports whether IP options are to transmitted in the IP header of each outgoing packet and what those options are.
IP_TOS	Reports the type-of-service field in IP header of outgoing packets.
IP_TTL	Reports the time-to-live field in IP header of outgoing packets.

setsockopt() Parameters

The following additional parameters for the setsockopt () call are supported for OpenEdition (UNIX System Services) Integrated Sockets users.

Socket-Level Options

Socket -Level options for OpenEdition (UNIX System Services) setsockopt():

SO_KEEPALIVE	Sets whether connections are to be kept open with periodic transmissions of messages. Packet interval is the TIB KEEPALIVETIMER value. Use TCP_KEEPALIVE to change the interval.
SO_RCVLOWAT	Sets minimum number of bytes to process for socket input operations. This option can be set but will be ignored for SOCK_DGRAM and SOCK_RAW socket types.

SO_REUSEADDR	Sets whether the rules used in validating addresses supplied to bind() should allow reuse of local addresses. When enabled, local addresses that are already in use may be bound. The system checks at connect time to ensure that the set <laddr, lport, raddr, rport> are not already in use by another association; if the set is already in use, EADDRINUSE is returned. The system checks at listen time to see if the port is already being listened on.
SO_SNDLOWAT	Sets minimum number of bytes to process for socket output operations. This option can be set but will be ignored for SOCK_DGRAM and SOCK_RAW socket types.

Note SO_RCVBYTCNT, SO_RCVREQCNT, SO_SNDBYTCNT, SO_SNDREQCNT, SO_SENDALL, and SO_READFRAG are not supported for OpenEdition (UNIX System Services) sockets.

TCP-Level Options

TCP-Level options for setsockopt():

TCP_KEEPALIVE	Sets the time interval between keep alive packets in seconds. SO_KEEPALIVE must be enabled first.

UDP-Level Options

UDP-Level options for setsockopt():

UDP_CHECKSUM	Sets whether UDP checksum computation is to be performed. This value is given in type int. 0 = OFF, non-zero = ON.

IP-Level Options

IP-Level options for setsockopt():

IP_HDRINCL	Sets whether, for a SOCK_RAW socket, the complete IP header will be included with the data on send operations.
IP_OPTIONS	Sets whether IP options are to transmitted in the IP header of each outgoing packet and what those options are.
IP_TOS	Sets the type-of-service field in IP header of outgoing packets.
IP_TTL	Sets the time-to-live field in IP header of outgoing packets.

recvmsg()

Ancillary data will not be supported in this release. Ancillary data is referenced via the msghdr fields msg_control and msg_controllen. These fields are not yet supported in the OpenEdition (UNIX System Services) msghdr structure.

Resolving Names and Addresses

OpenEdition (UNIX System Services) MVS version 1.2 uses the LE/370 version 1.3 or 1.4 runtime libraries to perform certain socket related functions such as gethostbyname(), getprotobyname(), etc. To perform this functionality, the LE/370 runtime library (RTL) reads specific MVS data sets to map services to names and to obtain domain name resolution configuration information.

Note These data sets and the utilities to build them are NOT distributed with either OpenEdition (UNIX System Services) MVS or the LE/370 Runtime Library.

IBM distributes these data sets with the IBM TCP/IP product for MVS only.

This section describes the steps necessary to build and configure these required data sets using sample members and utilities provided with the Cisco IOS for S/390 product.

The LE/370 version 1.5 RTL uses members in the /etc directory for the same functionality as the MVS data sets.

Using /etc Files with OpenEdition (UNIX System Services) for Domain Name Resolution

OpenEdition (UNIX System Services) users running MVS/ESA 5.2.1 or earlier are limited to the MVS host files described in this chapter for resolving host name and other DNR requests.

With MVS/ESA 5.2.2, it is possible to use HFS /etc files for DNR resolution. To use the /etc files, these program changes are necessary.

1. Delete the #include for <manifest.h>.
2. Replace any #include of <bsdtypes.h> with <sys/types.h>.
3. Replace any #include of <bsdtime.h> with <sys/time.h>.
4. Remove SYS1.SFOMHDRS from the -INC list for C89 or any compile PROCLIB cataloged procedures.
5. Replace any #define _OPEN_SOCKETS (or other references to _OPEN_SOCKETS) with _OE_SOCKETS.
6. If you are using threads, replace any extern h_errno with #define h_errno *(__herrno()).
7. Where RPC functions are being used, add a #include for <rpc/netdb.h>.

Making these changes generates the code needed to reference the /etc files for DNR requests. If no changes are made, the MVS files will be used as described in the rest of this chapter.

Data Sets for Host Resolution

The LE/370 RTL version 1.3 and 1.4 require five data sets to correctly perform the supporting socket functions such as gethostbyname(), getprotobyname(), etc. These five data sets are:

prefix.TCPIP.DATA	master configuration file
prefix.ETC.PROTO	protocol name mappings
prefix.ETC.RPC	RPC service name mappings
prefix.ETC.SERVICES	service name mappings
prefix.STANDARD.TCPXLBIN	ASCII to EBCDIC translation file

Two other data sets are used by the RTL if host name resolution is being done in local mode. These data sets are built with the IBM MAKESITE utility. Cisco IOS for S/390 does *not* provide a replacement for MAKESITE because the RTL can perform host name resolution using name servers without these other data sets, and users without a name server can upgrade to LE/370 version 1.5 and configure hosts in /etc/hosts as a workaround. The two other data sets are:

```
prefix.SITEINFO
prefix.ADDRINFO
```

The prefix.TCPIP.DATA data set is the primary configuration member for the RTL. It defines the prefix for the rest of the data sets used by the RTL and its prefix may be different from those data sets. In addition, it provides the configuration information for host name resolution process including the domain name (for example company.com), the resolver's internet address (for example 148.52.128.104) and resolver settings such as protocol (for example UDP), port (for example 53), timeout (for example 30 seconds) and retries (for example 3).

Search Procedures

The RTL performs a complex search sequence when it needs to locate the prefix.TCPIP.DATA data set, since there is no generic way to define to the RTL what the prefix actually is. The RTL attempts to locate the data set using the following search sequence:

1. If the environment variable, RESOLVER_CONFIG is defined, it uses this value to locate the data set. Usually this variable would be set in each user's .profile member in their home directory.

   ```
   export RESOLVER_CONFIG=//"''"IOS.OMVS.TCPIP.DATA"'"
   ```

   Note that the name must have two leading forward slashes and the actual data set name must be enclosed by a double-quote:single-quote:double-quote sequence (" ' ") on each end of the name.

2. It then looks for a SYSTCPDD DD statement. This is not recommended for any application which fork(s) due to OpenEdition MVS's failure to copy data set allocations across a fork.

   ```
   //SYSTCPDD DD DSN=IOS.OMVS.TCPIP.DATA,DISP=SHR
   ```

3. The jobname or userid is used as the prefix.

   For user IBMUSER, the RTL searches for IBMUSER.TCPIP.DATA.

4. The RTL then looks for SYS1.TCPPARM(TCPDATA).

5. The RTL then looks for TCPIP.TCPIP.DATA.

Configuration

Users migrating from IBM's TCP/IP for MVS to Cisco IOS for S/390 should note that the IBM TCP/IP installation utility EZAPPRFX will not affect the default prefix used by the Run Time Library (for example TCPIP).

The DATASETPREFIX must be configured in the prefix.TCPIP.DATA data set. It must be updated to specify the prefix used by the other four data sets. The DOMAINORIGIN keyword must be configured with the site's domain name (for example hq.company.com). NSINTERSERV keyword must be configured with the sites domain name server. The other keywords may be updated if desired to tune the resolver process.

The data sets, prefix.TCPIP.DATA, prefix.ETC.PROTO, prefix.ETC.RPC and prefix.ETC.SERVICES, must be allocated with attributes of RECFM=FB, LRECL=80, BLKSIZE=3120 and with an allocation of one track. The sample members TCPDATA, ETCPROTO, ETCRPC and ETCSERV, respectfully, should be copied from the IOS390.SAMP data set to the new data sets after allocation.

The prefix.STANDARD.TCPXLBIN data set must be allocated with attributes of RECFM=FB, LRECL=256, BLKSIZE=256 and an allocation of one track. Its purpose is to provide ASCII-to-EBCDIC translation tables for the domain name resolution process. It is created using either the CONVXL8 or LOADXL8 utilities.

```
LOADXL8 ENGLISH 'IOS390.OMVS.STANDARD.TCPXLBIN'
```

Users of the LE/370 RTL Version 1.5 and above must configure members in the /etc directory instead of in MVS data sets. The /etc/protocols, /etc/rpc and /etc/services may be created using the OPUT command and the Cisco IOS for S/390 sample members.

```
OPUT 'IOS390.SAMP(ETCPROTO)' '/etc/protocols'.
```

LE/370 RTL Version 1.5 and above may also require the configuration of other /etc members such as /etc/resolv.config, /etc/hosts and /etc/networks.

A sample /etc/resolv.config member is:

```
domain hq.company.com
search hq.company.com ohio.company.com company.com
nameserver 148.52.32.165
nameserver 148.52.32.56
```

A sample /etc/hosts member is:

```
127.0.0.1      localhost loopback
148.52.32.165    homehost homehost.md.mycompany.com
```

A sample /etc/networks member is:

```
127       loopback
148.52    mycompany
```

Consult IBM C/C++ for MVS/ESA, C/MVS Library Reference: *OpenEdition MVS Sockets*, Version 3 Release 1, Document Number SC23-3875-00, Program Number 5655-121, Section: 1.3.6.1, "Understanding TCP/IP Data Set Names with OpenEdition MVS" for more information on configuring OpenEdition MVS and the LE/370 RTL.

TSO Commands

This section describes the TSO commands available to Cisco IOS for S/390 OpenEdition socket users.

CONVXL8 Command

The TSO command CONVXL8 converts a table from editable text to binary.

CONVXL8 creates a data set with three records. Each record is 256 bytes in length. The first record has "*TCP/IP translate tables" (in EBCDIC) starting in column one, with the remainder of the record padded with EBCDIC blanks (X'40'). The second record will have 256 EBCDIC values representing the ASCII-to-EBCDIC translation. The third record will have 256 ASCII values representing the EBCDIC-to-ASCII translation.

File names use TSO prefix as defined by TSO rules. A fully qualified data set name needs to be enclosed in quotes. A data set name without quotes may have a user specified prefix placed before the name. This prefix is defined by the user's TSO profile. Refer to TSO documentation for more information about prefixes.

Syntax

```
CONVXL8 INPUT OUTPUT
```

INPUT Specifies the source data set to be converted. The data set must be in standard IBM format for SBCS translation tables. If input is a PDS member, INPUT should be specified as dsname(member). This parameter is required.

Default: None

Note Translate tables in the SAMP data set are not in this format. Use the TSO command LOADXL8 to prepare them for use with OpenEdition (UNIX System Services). Read LOADXL8 Command for more information.

OUTPUT Specifies the output data set created by the conversion. If output is a PDS member, OUTPUT should be specified as dsname(member). This parameter is required.

Default: None

```
CONVXL8 LIB.SOURCE(TRANS) LIB.TRANTAB
```

This will read USER.LIB.SOURCE(TRANS) and create a translate table in USER.LIB.TRANTAB.

```
CONVXL8 'SYSTEM.TCP.DATA(TRAN)'  'SYSTEM.BIN.TRANS'
```

This will read SYSTEM.TCP.DATA(TRAN) and create a translate table in SYSTEM.BIN.TRANS.

LOADXL8 Command

LOADXL8 has the same functionality as the CONVXL8 command, but it reads the load module from compiled translate tables as input. Note that it does not read the source. It loads the module from STEPLIB or TSO TASKLIB. The module can be converted for OpenEdition use with this command.

```
CALL 'trgindx.LOAD(LOADXL8)'     'table-name '   'indx.STANDARD.TCPXLBIN' ' '
```

trgindx	Cisco IOS for S/390 high level qualifier
table-name	load module name of translate table to be loaded
indx	OpenEdition high level qualifier

ILATCH command

There is an IFS command that allows you to display and free latches used to serialize data. For more information about the ILATCH command, read the *Cisco IOS for S/390 System Management Guide*.

Debugging Information

This section describes procedures to help you debug problems with Cisco IOS for S/390 OpenEdition (UNIX System Services) support.

Initialization

If you are having problems with the initialization of OpenEdition (UNIX System Services) sockets, check the following:

- Examine the SMP/E job output to verify successful processing
- Verify the release and maintenance levels of OpenEdition - Version 5.1, 5.2, or 5.2.2.
- Verify the PFS configuration statements:
  - FILESYSTYPE
  - NETWORK
  - the OpenEdition MVS PROC
- Examine any IVP return codes or other return codes or messages from OpenEdition utilities.
- Verify the logs for error messages. Also look at the console log and any dump data sets for abends that may have been created.
- Verify non-OpenEdition access to Cisco IOS for S/390. Use tools such as ping, FTP, or Telnet to verify that the Cisco IOS for S/390 transport provider is functional.

Application Issues

If you are having problems with an OpenEdition (UNIX System Services) Applications, check the following:

- Before calling Customer Support, get a good description of the failure with symptoms, return code, errno, and errno junior codes. Examine any messages from the application. Be sure to include a brief description of the application and known socket services.
- Verify the release and maintenance level of OpenEdition - Version 5.1, 5.2, or 5.2.2.
- Examine the logs for error message. Look at the console log and any dump data sets for abends that may have been created.
- Isolate the problem to your application by testing the status of other OpenEdition applications and non-OpenEdition application functionality (ping, Telnet, FTP, etc.)
- Issue an “TCP SNAP ALL” command immediately after any application failure.

TCP SNAP ALL Command

The existing command:

```
/F IOS390,TCP SNAP ALL
```

has been modified to dump the OpenEdition PFS Transport Provider and Socket API control blocks in the Cisco IOS for S/390 address space.

Stopping and Starting Sockets

The socket API can be stopped and restarted by stopping and restarting the ACP task group within Cisco IOS for S/390.

The OpenEdition MVS PFS can only be stopped and restarted by stopping and restarting OpenEdition MVS.

Limitations

The number of Socket API endpoints supported by the OpenEdition PFS and Socket API is currently limited by the following constraints:

- The amount of virtual storage in the Cisco IOS for S/390 address space.
- The number of ports. Socket API programs generally don't share port numbers. The number of port numbers is limited to 65,535 for both TCP and UDP. Client applications which reuse the same port number when connected to different servers can help ease this limitation.

The range of ports for both TCP and UDP may be decreased by setting overriding values on the TCP and UDP statements of TCPCFGxx. Setting these values may decrease the number of ports and sockets.

- OpenEdition MVS requires that a maximum socket number be set.
- One latch per endpoint, with a maximum of 32,767.

Release Information

Cisco IOS for S/390 uses operating system facilities and architecture dependent instructions which are only available in MVS/ESA versions 3.1.3 and above.

The OpenEdition PFS only supports MVS/ESA version 5.1.0 and above.

The OpenEdition PFS Pre-Router is only supported on MVS/ESA 5.2.2 and above.

Common INET Support

IBM offers an OpenEdition (UNIX System Services) MVS facility within MVS/ESA Version 5.2.2 and higher that allows a socket program to be used with multiple TCP/IP stacks simultaneously, without knowledge of the application and without coding modifications. Although this may appear to be very beneficial, use of this feature is not without risk and customers are strongly advised to consider alternatives before implementing Common Inet Support.

Benefits of INET Support

The benefits of using Common Inet Support are:

- Applications are developed as if they were using the AF_INET family with a single TCP/IP stack.
- The system administrator adds and removes TCP/IP address spaces to the system configuration as required without application knowledge.
- Applications, if coded properly, may choose a particular TCP/IP region by issuing an ioctl() call after the socket is created.

Problems of INET Support

The problems with using Common Inet Support are:

- Multiple copies of TCP/IP running on the host all use the AF_INET value of 2 in order to appear to be a single Physical File System. This presents the following problems:
  - Host resource consumption is higher than that single region or multiple regions using different address family values. This is mainly the result of the Common Inet Layer opening a socket to every Physical File System every time a socket function is issued. Other socket function calls up through the time a connection is established via a particular Physical File System must be processed by all Physical File Systems. Once an outbound connection is established over a single Physical File System, the sockets created within the other Physical File Systems are closed by the Common Inet Layer. Listening sockets and datagram type sockets are always open in all Physical File Systems.
  - There are no conflicting parameters to deal with such as interface name, local host IP address, and interface addresses. Loopback connections are made via the selected Physical File System.

Problems of Multiple Physical File Systems

The problems with using multiple Physical File Systems are:

- If an application requires the use of more than one TCP/IP region at the same time, the application must open a socket to each. This, however, is beneficial if one TCP/IP region is recycled. The application can close the socket to that region, and then reopen it when the region is restarted. There is no bad side effect to the other sockets connected to other TCP/IP regions. On the positive side of this, however, the application immediately knows of a failure of one region.
- The application will most likely need to be using non-blocking socket functions. This is because events can occur simultaneously on multiple Physical File Systems (for example two inbound connects occur at the same time, one on each TCP/IP region).

Established connections are not fault tolerant across TCP/IP regions (only within a TCP/IP region).