SAS/C Socket Library Interface (LSCNCOM)

This appendix describes the LSCNCOM interface between the SAS/C socket library and the API.

SAS/C Socket Interface

The SAS/C socket interface, LSCNCOM, is a vendor independent socket library that is provided as part of the SAS/C compiler. This interface allows programs to be written that can use different vendors' TCP products at execution time without having to be recompiled.

The LSCNCOM interface relies on the SAS/C compiler and runtime library at the version 5.50 level or higher.

Restrictions

The following features of the interface, as documented in the SAS *Technical Report C-111* (hereafter referred to as simply *C-111*), are not supported by this release of the Cisco IOS for S/390 module:

- No support is provided for raw sockets or options that deal with the basic IP data stream. This includes the MSG_DONTROUTE option, ioctl() SIOCGIFxxx options, and setsockopt() SO_RAW option. Also, other operations that refer to fields in the if.h structure may not be supported.

Note See the SAS *Usage Note 1108* for a workaround zap that lets the RPC library function without raw support.

- SO_KEEPALIVE is not supported.
- Options F_GETFD and F_SETFD of fcntl() are not supported in the LSCNCOM routine since the #define values are being used for Cisco IOS for S/390 options. This conflict will be resolved in a future release.
- The ioctl() option FIONREAD is not documented properly. The SAS manual states that this option returns a value of 1 if there is data to be read, and a value of 0 if there is no data. However, this function actually returns the number of bytes waiting (if greater than zero), as the socket library (and the BSD man page) states.
- Writes that are flow-controlled may not be redriven.

- Only the gethostbyname() and gethostbyaddr() functions are handled by the Domain Name Resolver (DNR) when it is chosen. The database-related calls listed below (and described in *C-111*) are handled by the SAS resolver even though an equivalent file (and function) may be provided by Cisco IOS for S/390. Therefore, to use all functions of the interface, the IBM-style /etc files (described in *C-111*) must also be defined.

getnetbyname	getservbyport	setprotoent	getnetbyaddr
getpeerent	setservent	gethostbyname	getnetent
endpeerent	gethostbyaddr	getprotoent	endnetent
getprotobyname	getservent	endprotoent	getprotobynumber
setpeerent	endservent	getservbyname	setnetent

  See the table “Setup for SAS Socket /etc Files” for equivalent DNR parameter files.

- Due to Cisco IOS for S/390 restrictions, the functions givesocket() and takesocket() are limited to passing sockets between tasks in the same address space.

Requirements

SAS/CONNECT requires a PTF from SAS in order to function correctly with the LSCNCOM routine. With this co-requisite fix, SAS validates the use of SAS/CONNECT with Cisco IOS for S/390 and LSCNCOM.

Customers can run SAS/CONNECT or SAS/SHARE with Cisco IOS for S/390. In addition to the necessary Cisco IOS for S/390 maintenance for the SAS/C 5.50 socket library, customers also need the following:

- V6.08 of SAS (the latest SAS major release)

AND

- Maintenance Level TS410 with the zap documented in SAS Note V6-SYS.SYS-08338

OR

- Maintenance level TS415 and no zap needed

Certification

The RPC portion of the SAS/CSL product has not been certified. Certification will be completed in a future release.

Note SAS has a workaround zap, documented in their *Usage Note 1108*, that can be used to bypass some of the unsupported ioctl() options. The standard SUN/RPC getmyaddress function does not work with Cisco IOS for S/390 because the ioctl() function supplied does not support SIOCGIFCONF or SIOCGIFFLAGS ioctl commands. Instead, a hard-coded loopback IP address (127.0.0.1) must be returned. To correct the problem, apply the zap referenced by Z1001108.

Usage

The LSCNCOM routine, and its alias L$CNCOM, should replace or be placed before the SAS-supplied version in the link-list search order. The routine is dynamically loaded on the first call to a SAS Socket Library function.

Using the Cisco IOS for S/390 Variables

If you want to use the Cisco IOS for S/390 socket variables, you must define the following symbol in your source file. It must be placed before the #include statement. Use the following as a guide:

```
.
.
.
#define __INTERLINK_TCPIP
.
.
.
#include <sys/socket.h>
```

Environment Variables

The environment variables below are recognized by the LSCNCOM interface:

ICS_SUBSYS

The subsystem name of the Cisco IOS for S/390 API task that was defined in the ACPCONxx parameter member. For compatibility with earlier releases, SUBSYS and TCPIP_MACH (first four bytes only) are also recognized.

Default: ACSS

ICS_RESOLVER

Defines the order in which the DNR and SAS resolver are used:

ONLY	Cisco IOS for S/390 resolver only, return OK or error
FIRST	Cisco IOS for S/390 is called first; SAS is called if there is an error
LAST	SAS is called first; Cisco IOS for S/390 is called if there is an error
NEVER	SAS resolver always, return OK or error

These variables should be set prior to the first call to an LSCNCOM function. Either the PUTENV TSO command or inline (execute-time) parameters can be used to define or override the above variables.

Since it is difficult to delete permanent environment variables, the LSCNCOM interface treats a variable that is defined but has a null value as if it were not set. If none of the options are set, the default is used.

Default: ONLY

Setup for SAS Socket /etc Files

This table shows the configuration files used by the LSCNCOM interface, along with format documentation and equivalent parameter member for the DNR configuration. Either the files can be created with the expected names or environment variables can be setup to override the file names as documented in *C-111*.

Table A-1 Setup for SAS Socket /etc Files

SAS/C Default MVS Names	UNIX File	MAN Page	Equivalent DNR PARM Member
id.ETC.PROTOCOLS	/etc/protocols	protocols(4)	DNRPRTxx
id.ETC.SERVICES	/etc/services	services(4)	Syntax differs; you must create this from the DNRSVCxx member or copy from the workstation. The SAS resolver recognizes case-sensitive names, but DNRSLCxx is uppercase only.
id.ETC.HOSTS	/etc/hosts	hosts(4)	Syntax differs; you must create this from the DNRHSTxx member (static) or use the ICS or SAS resolver.
id.ETC.NETWORKS	/etc/networks	networks(4)	DNRNETxx DNRNETxx should be unnumbered or the SAS resolver interprets the sequence field as an alias.
id.ETC.RESOLV.CONF	/etc/resolv.conf	resolv.conf(4)	No equivalent; you must extract information from DNRSVCxx (domain name) and DNRNSCxx (name servers) to create this file. If no name servers are defined, static name resolution is used. See C-111.
id.ETC.RPC	/etc/rpc	rpc(4)	DNRRPCxx The overrides can either be via a DD name or via a fully qualified data set name.

Environment Variable Example

This is an example of the environment variables used during the testing of these updates:

Example

```
ICS_SUBSYS = acss
X11R5_PREFIX = SASC.CSL100
ICS_RESOLVER = only
DISPLAY = UNIX:0.0
ETC_PROTOCOLS = dsn:tcpics.parm(dnrprt00)
ETC_NETWORKS = dsn:tcpics.parm(dnrnet00)
ETC_RESOLV_CONF = dsn:tcpics.test.conf
ETC_HOSTS = dsn:tcpics.test.hosts
ETC_SERVICES = dsn:tcpics.test.service
ETC_RPC = dsn:tcpics.parm(dnrrpc00)
```