CA Common Services for z/OS

Reference Guide

Release 14.1.00

CA Technologies Product References

This document references some of the following CA Technologies products:

- CA 1® Tape Management
- CA 7® Workload Automation
- CA 11™ Workload Automation Restart and Tracking
- CA ACF2™
- CA Allocate™ DASD Space and Placement
- CA Audit
- CA Automation Point
- CA Balancing
- CA Bundl®
- CA Database Analyzer™ for DB2 for z/OS
- CA Datacom®/AD
- CA Data Compressor™ for DB2 for z/OS
- CA DB2
- CA Deliver™
- CA Disk™ Backup and Restore
- CA Dispatch™
- CA Earl™
- CA Endevor® Software Change Manager
- CA Fast Check® for DB2 for z/OS
- CA Fast Index® for DB2 for z/OS
- CA Fast Load for DB2 for z/OS
- CA Fast Recover® for DB2 for z/OS
- CA Fast Unload® for DB2 for z/OS
- CA IDMS™
- CA IDMB™/DB
- CA Insight™ Database Performance Monitor for DB2 for z/OS
- CA Index Expert™ for DB2 for z/OS
- CA JARS®
- CA JARS® Resource Accounting

- CA Jobtrac™ Job Management
- CA Log Analyzer™ for DB2 for z/OS
- CA Mainframe Software Manager™ (CA MSM)
- CA Merge/Modify™ for DB2 for z/OS
- CA MIA Tape Sharing
- CA MIC Message Sharing
- CA MICS® Resource Management
- CA MII Data Sharing
- CA MIM™ Resource Sharing
- CA NetMaster® File Transfer Management
- CA NetMaster® Network Automation
- CA NetMaster® Network Management for SNA
- CA NetMaster® Network Management for TCP/IP
- CA NetMaster® Network Operations for TCP/IP
- CA NetSpy™ Network Performance
- CA Network and Systems Management
- CA NSM System Status Manager
- CA OPS/MVS® Event Management and Automation
- CA Partition Expert™ for DB2 for z/OS
- CA Plan Analyzer® for DB2 for z/OS
- CA Quick Copy for DB2 for z/OS
- CA Rapid Reorg® for DB2 for z/OS
- CA RC/Extract™ for DB2 for z/OS
- CA RC/Migrator™ for DB2 for z/OS
- CA RC/Query® for DB2 for z/OS
- CA RC/Secure™ for DB2 for z/OS
- CA RC/Update™ for DB2 for z/OS
- CA Recovery Analyzer™ for DB2 for z/OS
- CA Roscoe®
- CA Scheduler® Job Management
- CA SYSVIEW® Performance Management
- CA Service Desk (Service Desk)
- CA Spool™ Enterprise Print Management

- CA SQL Ease® for DB2 for z/OS
- CA SYSVIEW® Performance Management
- CA TCPAccess™ Communications Server for z/OS
- CA TLMS Tape Management
- CA Top Secret®
- CA TPX™ Session Management for z/OS
- CA Value Pack for DB2
- CA Vantage™ Storage Resource Manager
- CA View®
- CA XCOM™
- CA Workload Control Center

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Event Management Commands

This section contains the following topics:

caevtsec-Update Message Action Restriction Rules

Valid on UNIX/Linux

This is a menu-driven executable that lets you update the rules for sending the message actions COMMAND, UNIXCMD, and UNIXSH to your local host. It will alter the actnode.prf file with the information you supply. caevtsec is an alternative to using vi or another UNIX/Linux editor to update the file.

actnode.prf-Maintain Message Action Restriction Rules

Use this file to maintain policies that specify how message action restriction is to be enforced based on the submitting node and RUNID. It is located in the *$CAIGLBL0000/opr/config/hostname* directory. The file must be owned by root and only a UID of 0 can have write access to it.

This file is created when Event Management is installed. A prompt lets you decide whether you want to override the default setting that disables the message action restriction feature.

An individual entry in the actnode.prf file has the following format:

```
-n=nodename,runid,flag
```

nodename

This is the node from which the COMMAND, UNIXCMD, or UNIXSH message action is initiated. It can contain a trailing generic mask character.

runid

RUNID to whom the rule applies. It can contain a trailing generic mask character.

flag

Use **D** for disable (feature is active; disallow the message action submitted by RUNID from nodename), **E** for enable (allow the RUNID from nodename to submit the message action), or **W** for warn (check the rule but allow the message action submission to occur).

Examples

This is the default rule in effect if, during installation, you elected not to activate message action restriction:

```
-n=*,*,E
```

The rule states that for all nodes and all RUNIDs, COMMAND, UNIXCMD and UNIXSH message action submission is allowed.

This is the default rule in effect if, during installation, you elected to activate message action restriction:

```
-n=*,*,D
```

The rule states that for all nodes and all RUNIDs, COMMAND, UNIXCMD and UNIXSH message action submission is disallowed.

This combination of rules only enforces a message action restriction on RUNID root and allows all other RUNIDs to submit the message actions:

```
-n=*,*,E
-n=*,root,D
```

This combination of rules allows all RUNIDs to bypass message action restriction unless the request comes from the node mars:

```
-n=*,*,E
-n=mars,*,D
-n=*,root,W
```

In that case, message action restriction is enforced for all RUNIDs. The last entry sets a warning type restriction rule for RUNID root if it comes from a node other than mars.

Event Management scans the entire configuration file for a best match and uses that rule. It uses the node field as a high level qualifier when searching for a best match. For example:

```
-n=mars,*,D
-n=*,root,W
```

If these are the only two entries in the file, any request coming from the node mars uses the disallow rule. The user root only uses the warning rule if the request comes from a node other than mars.

camibwalk/camibwlk-Query SNMP Agents for MIB Information

Valid on UNIX/Linux, Windows, z/OS

The camibwalk/camibwlk command is used to query SNMP agents running on your network nodes for status and other information that can be extracted from the objects defined by the MIBs. The types of nodes that can be queried go beyond machines running UNIX/Linux and include intelligent devices such as routers, printers, X-terminals, hubs, and others.

Following are examples of the types of data that camibwalk/camibwlk can extract:

From HP LaserJet IIIsi printers:

- Printer OFFLINE
- Printer ONLINE
- Human Intervention required
- Unspecified Paper Problem (probably jam, or out of forms)
- Report Device Error Codes

From HP LaserJet 4si printers:

- All the data extractable from HP LaserJet IIIsi printers
- Door Open
- Paper Out
- Paper Jam
- Toner Low
- Peripheral error
- Current contents of LCD status display.

Other general information that camibwlk can extract from these agents:

- System Description
- Hardware address
- Time since last cold start
- Adjacent nodes in the network topology

In addition to these, CA continues to add additional extensions to the camibwalk/camibwlk MIB interpretation capabilities, to extract greater levels of detail from the expanding class of SNMP agents.

For UNIX/Linux, z/OS

`camibwalk` *`node community variable`*

For Windows

```
camibwlk  node community variable
```

Example

This example queries the SNMP agent on the hpnode33 node, in the public community, for information about object .1.3.6.1.2.1.1.1 which is the system description:

```
camibwalk hpnode33 public .1.3.6.1.2.1.1.1
```

Output

```
Name: system.sysDescr.0
OCTET STRING- (ascii):HP-UX hpnode33 A.09.00 E 9000/8371443363221G
```

In the sample output above, "Name" is the translated MIB OBJECT-TYPE string of the numeric object provided in the query. camibwlk does not translate any enterprise-specific OBJECT-TYPE; they are displayed numerically.

OCTET-STRING- (ascii) is the MIB SYNTAX type of the numeric object provided in the query. This example indicates that system description is defined as an OCTET-STRING in the MIB-II MIB. The information after the SYNTAX type is the value of the queried object.

node

The IP address or the host name of the node to be queried.

Note: For IPv6, a hostname must be used.

community

Identifies the community to which the queried node belongs.

variable

Identifies the portion of the object identifier to be queried. This variable has the format .a.b.c.d..., where a, b, c, and d are sub-identifiers in decimal notation.

careply-Reply to a cawtor Message

Valid on UNIX/Linux, Windows, z/OS

Use the Event Management careply command at any terminal to reply to a message sent to the Enterprise Management operator console with the cawtor command.

Replies sent using the careply command are written to the standard output of the process issuing the cawtor command

This command has the following format:

careply [-n *node*] [-s *source*] [-g *category*] *m* [*text*]

node

Identifies the machine name to which the reply is to be directed. If the node is not available, the reply is sent to the local machine. Note that for the node option, it is necessary to type the brackets [ ] as part of the command.

-n *node*

Directs the reply to the identified node, which is one other than the node the user is on. If the node is not available, the reply is sent to the local node.

-s *source*

Assigns the identified source to the message for the purpose of message matching.

-g *category*

Assigns the identified category to the message for the purpose of message matching.

text

1- to 255-alphanumeric bytes of text, which complete the reply to the message.

Note: Since the careply command is issued from the command line, the text string you supply is subject to evaluation by the shell you are executing. Therefore, embedded blanks, special characters, and quotes in the text string require special consideration. See the documentation provided with your operating system for the rules and guidelines regarding text strings.

n

Identifies the message number generated by the cawtor command of the message that you must respond to.

Environment Variables

CAI_WTOR_NODE

Environment variable that you can set to an alternate node. If this variable is set, and you issue careply without a node specification, the value of CAI_WTOR_NODE is used.

Valid on UNIX/Linux only

/usr/tmp/cawtor.n

If for any reason the Event Management daemon or CAICCI is down, these files are created so that the cawtor and careply commands can continue to function normally. The value of *n* is the message number the cawtor command generated. This message number is identical to the PID of the process issuing the cawtor when either the Event Management daemon or CAICCI is down.

Before a command, reply, or acknowledgement is executed from the Console GUI using the oprcmd, careply, or delkeep commands, or by a message action, an authorization check is performed to verify that the user ID can perform this action.

If CA NSM security is inactive, the user ID issuing the command must be listed in the Users authorized to issue commands environment variable. This list is a setting that is edited in the Configuration Settings notebook, on the Event Management Preferences page.

Note: Wild cards can be used. See the online help for the user profile--Access Management dialog for more details.

Examples

If this command is assigned to the message number 25, the following command sends the reply to the process issuing the cawtor:

```
cawtor OK to continue?
careply 25 Yes.
```

catdadmin-Control the catrapd Daemon

Valid on UNIX/Linux, z/OS

The catdadmin command ($CAIGLBL0000/snmp/scripts/catdadmin) controls the operation of the catrapd daemon. You can specify only one action per invocation.

This command has the following format:

```
catdadmin {-i|-d|-o|-s}
```

-i

Initiate (start) the catrapd daemon.

-d

Enable diagnostic traces.

-o

Turn diagnostic traces off.

-s

Shut down the catrapd daemon.

catrap-Issue SNMP Traps in a Network

Valid on UNIX/Linux, Windows, z/OS

Note: Windows TCP/IP support must be installed for this command to be operational.

In this discussion, %CAIGLBL0000% for Windows and $CAIGLBL0000 for UNIX/Linux refers to the Enterprise Management installation directory.

A key feature of the SNMP facilities of Enterprise Management is the catrap command (%CAIGLBL0000%\bin\catrap.exe (Windows), or $CAIGLBL0000/snmp/bin/catrap (UNIX/Linux, z/OS)). catrap can issue SNMP traps to any destination in your network.

Unlike other commands capable of issuing SNMP traps, catrap does not require optional Network Management products to be licensed on the node on which it is executed. Additionally, no special authority is needed to run the catrap command.

The catrap command supports all the operands accepted as Open System standards for an SNMP trap command. It can be used interactively, through shell scripts, or as part of automated event handling policies defined to the Event Management function. The operands provided as destination and information data to the catrap command are automatically converted into the appropriate Open Systems standard datagram and sent to the designated trap destination.

The catrap command can coexist with existing SNMP daemons through the use of port sharing. The catrap command opens the port only for the duration of time required to send the SNMP message. The catrap command makes use of the system call getservbyname() to determine the actual port number to use (by convention, port 161 is reserved for this use). If this service has been modified, catrap may not be able to locate the correct port. Most operating systems with TCP/IP support utilize an /etc/services file, which can be consulted to determine the port number reserved for SNMP on that machine. On a Windows system, look for this file in %SystemRoot%\SYSTEM32\DRIVERS\ETC\SERVICES .

SNMP traps are typically issued by an agent implemented in the firmware of a particular device. The catrap command, however, makes it simple for user applications, shell scripts that are part of production jobs, or Event Management policies to issue SNMP traps of their own, by executing this command with the appropriate arguments.

Unlike other SNMP trap commands, the catrap command provided with Enterprise Management is not restricted to any particular set of ISO or Enterprise MIBs, and is totally open for use with any MIB or pseudo MIB. While just as real and meaningful as an ISO or Enterprise MIB, a pseudo MIB takes many defaults from the platform on which it is executing.

This command has the following format:

```
catrap [-d] [-t timeout] [-r retries] [-p port] [-c community]
          target-node
          enterprise-id
          agent address
          generic trap
          specific trap
          time stamp
          [variable binding info1][variable binding info2] [variable binding
           infon]
```

This command uses the following parameters:

-d

Display the trap request datagrams in dump format.

-t timeout

Identifies the timeout value as a positive integer in 1/10 second increments.

-r retries

Identifies the number of retries to be made when a busy or other recoverable error condition is detected while attempting to send the SNMP trap.

-p port

Identifies the remote port number to which the SNMP trap request is to be sent. Use this option if your Network Manager or the catrapd service provider is listening on a port other than the internet standard snmp-trap port, 162.

-c community

Send this trap to the specified SNMP community. This option may be used to override the default of public.

target-node

Identifies the node to which this SNMP trap is to be sent. It may be specified either as a host name or as an IP address.

Note: For IPv6, a hostname must be used.

enterprise-id

Identifies the enterprise MIB that should be associated with this SNMP trap. If specified as the setting for Windows or UNIX/Linux, system environment variable %CAICATD0000% (NT) or $CAICATD0000 (UNIX/Linux) is used. If %CAICATD0000% or $CAICATD0000 is not set, the CA Enterprise code 1.3.6.1.4.1.791 is used.

agent-address

This operand is provided for proxies that need to send trap requests for a host that cannot send SNMP trap requests. This agent address is used to identify the SNMP trap on the Event Management console log. This may be specified as a hostname or as an IP address. If specified as "" the IP address of the current hostname is used.

Note: For IPv6, a hostname must be used.

generic-trap

Specifies a single digit, in the range 0 to 6, which defines the class of generic trap being sent. Under most circumstances, use code 6 to indicate that an Enterprise-specific SNMP trap code is being used.

Codes 0 through 5 have specific industry standard predefined meanings as defined by the Internet Activities Board (IAB) RFC1215. These are:

0 Coldstart

1 Warmstart

2 Link down

3 Link up

4 Authentication failure

5 EGP neighbor loss

specific-trap

Identifies an Enterprise specific trap number up to a 32-bit integer. This number may identify a trap request for your organization. It can also be used to define an Enterprise specific MIB for your organization that identifies the trap codes to use. This allows your network manager to take advantage of the other facilities of SNMP Management.

time-stamp

Indicates the time, in hundredths of a second, that the application sending the SNMP trap has been active. Specify any whole number greater than or equal to zero. If specified as "", the time stamp value defaults to the value of the system uptime() call.

variable-binding-info, variable-binding-info2,…variable-binding infon (trapvar)

There are three parts to the binding information:

object-ID

data-type

data-value

You may specify multiple sets of binding information in a single trap request. However, in order to specify a subsequent set of binding information, the previous binding information must be completely specified (all three parts must be present).

The SNMP standard limits trap requests to 484 bytes. This limitation should not cause a problem during normal use. However, when you use multiple or long variable bindings, this limitation could cause a problem. To estimate the size of a trap, count the number of characters used in the catrap command, add 19 for datagram overhead, and 6 for each set of variable bindings. For a more accurate count, run catrapd with the d option. If you find that you are exceeding the length restriction, try splitting the variable bindings into multiple catrap commands, or reduce the length of text strings in the bindings.

Examples

1.3.6.1.2.1.1.1.0 octetstringascii 'hello world'

1.3.6.1.2.1.1.1.0 integer 12345

1.3.6.1.2.1.1.1.0 null

object-ID

The Object ID associated with this variable. If not specified, the setting for the Windows System Environment Variable%CAICATD0001% is used. If %CAICATD0001% is not set, the Internet MIBII-MGMT SYSTEM sysDescr code of 1.3.6.1.2.1.1.1.0 is used.

data-type

The type of variable data. The support types are defined in the IAB RFC1155 and derived data types, as follows:

integer

A number that can be represented as a signed 32-bit integer.

octetstring

An octet string of data.

octetstringascii

A string of ASCII characters. If this value is more than one token, enclose the value in single or double quotes.

gauge

A non-negative integer that may increase or decrease. The maximum value is 2 to the power of 32 -1 (4294967295 decimal)

counter

A non-negative integer that monotonically increases to a maximum value of 2 to the power of 32 -1 (4294967295 decimal)

timeticks

A non-negative integer that counts the time in hundredths of a second since some epoch.

opaque

See the Internet Activities Board (IAB) RFC1155 description.

objectidentifier

An Object ID.

null

No value.

ipaddress

An IP address, represented as an octetstring of length 4 (for example, 999.999.999.999).

data-value

The value to be associated with the variable binding in an ASCII representation.

Example 1

The following fictional scenario demonstrates how the catrap command might be used.

A company has defined its own pseudo MIB, which describes an event tree. Each node on the tree represents information that could be sent when specified as a variable on the catrap command.

Sending a trap with the variable of 999.1.1.2 is equivalent to sending the message that the enterprise database server that handles the general ledger database has been started.

The variable of 999.1.1.3 indicates that the General Ledger database has encountered a journal full condition. And the variable 999.2.1.1 indicates that General Ledger financial application has resumed processing after a temporary outage (warm start).

To take the example further, assume that Enterprise Management is executing on several nodes in this network, but you have decided that all SNMP trap traffic should be directed to a single monitoring machine running on a node called EVNTMGR. The EVNTMGR node receives trap traffic that is recorded and acted upon by the Event Management function of Enterprise Management.

Another machine in the network is used for the production of financial applications. This node is called FINPROD. For some unknown reason, an error occurs and the General Ledger production application running on node FINPROD terminates with an error. Testing the return code issued by this executable, the shell script realizes that the exit code indicates there was a problem, and issues an SNMP trap to alert the EVNTMGR node that something has gone wrong, by simply executing the following command:

```
catrap EVNTMGR ""   "" 6 0 22 999.2.1.3 integer 128
```

Note: A full syntactical description of the catrap command is included after this example.

The first operand directs the catrap command to send the identified trap information to the node EVNTMGR. The next two operands, "" and "", instruct catrap to take the default Enterprise code and the default agent address for this node. The number 6 indicates that this command is sending a specific trap, the 0 identifies the specific trap number for this example and 22 is an arbitrary number we have selected as a timestamp indicator. The next three operands identify the variable binding information for the trap, as follows: 999.2.1.3 is the Object ID of the object about which information is being sent. (If you refer back to the event tree described earlier, you can see that the string 999.2.1.3 refers to an error in the enterprise financial application, General Ledger). The last two operands provide additional information about the event; in this case, "send an integer value of 128 to node EVNTMGR." For this example, we are assuming that 128 is an error code that has some meaning to the General Ledger application, or it is the exit code that the shell script detected as indicating an error. When received at the trap target node, in this example, EVNTMGR, the Event Management function can then decode the event and perform automatic actions in response.

If you see the event tree, you can see what other types of events can be sent, such as 999.1.1.1, indicating that the Enterprise data server Database for the General Ledger system has been shut down.

The catrap command provides additional functionality when coupled with the capabilities of Enterprise Management. For example, you can use the catrap Event Management facilities to intercept the error messages from any application and automatically execute user customized catrap commands in response. The Workload Management function could detect key events and send traps in response to files available for processing, or applications completing their processing. When traps are received, Event Management message handling policies can be used to automatically open problem tickets, send warning messages to other consoles or terminals, start recovery jobs, post dependencies as having been met (so that other production jobs can proceed), issue additional SNMP traps to other nodes, or any number of other actions. The possibilities for using SNMP trap information are numerous.

Example 2

When the Enterprise Management catrap service provider is operational, SNMP traps directed to the node on which it is executing are automatically forwarded to Event Management. Once received, these trap messages are automatically recorded in the event log, and become eligible for sophisticated automatic processing by the Event Management function of Enterprise Management. This example shows you how to issue an SNMP trap (using the catrap command) and the format of the resulting message as it is presented to the Event Management function by the catrap service provider (catrapd) that receives the trap.

The following catrap command causes an SNMP trap to be sent to node xyzlhu33:

```
catrap xyzlhu33 "" "" 6 1 1 1.3.6.1.4.1.791.1.4 octetstring "hello world"
```

The catrap daemon (catrapd) running on node xyzlhu33 receives the trap and presents it to the Event Management function in the following format:

```
.CATD_I_060 SNMPTRAP:  -c  public  Computer.Associates 999.999.9.999 xyzlhu33 6 1
00:00:00 1 OID: 1.3.6.1.4.1.791.1.4
.iso.org.dod.internet.private.enterprises.791.1.4 VALUE: hello world
```

Within the Event Management function of Enterprise Management, this message is divided into character "tokens" which can be tested individually or in combination with one another as part of user-defined Event Management policies. The message in the preceding example has 17 tokens. The tokens are:

Token	Value	Description
&1	CATD_I_060	CA standard prefix
&2	SNMPTRAP:	Constant literal indicating this message originated as an SNMP trap
&3	-c	Indicates the beginning of the community parameter

Token	Value	Description
&4	Public	The community type
&5	Computer.Associates	The enterprise name, if found in the %CAIGLBL0000%\DB\enterprise.dat file (Windows) or $CAIGLBL0000/snmp/dat/enterprise.dat (UNIX/Linux, z/OS). If the enterprise ID is not found in the enterprise.dat file, the numeric value is printed.
&6	999.999.99.999	The IP address of the node where the trap originated.
&7	xyzlhu33	The node name of the originating machine.
&8	6	As specified in the catrap command, a generic trap of 6 indicates an Enterprise specific code has been assigned.
&9	1	As specified in the catrap command, the number of the specific trap.
&10	00:00:00	The time stamp from the catrap command, displayed in hh:mm:ss.
&11	1	The sequence number of the variable binding information from the catrap command.
&12	OID	Constant literal that describes the object that was the subject of this trap.
&13	1.3.6.1.4.1.791.1.4	The object identifier of the object that was the subject of this trap.
&14	iso.org.dod.internet.private.enterprises.791.1.4	Enterprise ID organization
&15	VALUE	Constant literal
&16	Hello	A value as specified in the catrap command
&17	World	A value as specified in the catrap command.

The preceding explanation of how SNMP traps are sent and how they are presented to the Event Management function of Enterprise Management provides the information needed to begin using SNMP facilities in an installation.

Note: If your organization does not have its own enterprise ID (assigned by the Internet Assigned Numbers Authority), you can continue to use the system descriptor 1.3.6.1.4.791.1 that was used in the preceding example. This descriptor represents subtree 1 of the CA enterprise ID. (The formal system descriptor for the CA enterprise ID is 1.3.6.1.4.1.791.1, where "1.3.6.1.4.1.791" is the ID that the Internet Assigned Numbers Authority has reserved for CA, and the subsequent ".1" represents the subtree that CA has reserved for client use.)

Enterprise IDs are assigned by an independent industry organization, the Internet Assigned Numbers Authority. If you wish to obtain an enterprise ID for your organization, you must request one from the Internet Assigned Numbers Authority, whose email address is iana@isi.edu mailto:iana@isi.edu.

catrapd-Control the catrapd Daemon

Valid on UNIX/Linux, Windows, z/OS

Note: Windows TCP/IP and SNMP support must be installed for this command to be operational.

In this discussion, %CAIGLBL0000% (Windows) and $CAIGLBL0000 (UNIX/Linux, z/OS) refer to the Enterprise Management installation directory.

The catrapd service provider (%CAIGLBL0000%\bin\catrapd.exe or $CAIGLBL0000/snmp/bin/catrapd) provides access to the SNMP events and related services on your network, and provides support for the following commands provided with Enterprise Management:

camibwalk

Queries the SNMP agents running on the network nodes for status and other information that can be extracted from Management Information Bases (MIBs).

catrap

Issues SNMP traps to any destination in your network. catrap does not require optional Network Management products to be licensed on the node on which it is executed.

Start the catrapd Daemon

Enterprise Management starts and stops catrapd if catrapd was activated during installation.

Valid on Windows

To activate catrapd

1. Click Configuration/Settings in the Enterprise Management folder.
2. Click the Component Activation Tab.
3. Set the SNMP Trap Server Activated (CAIACTRPTSV) option to Yes.

If catrapd is activated on a node where a network manager is active, catrapd automatically detects and interfaces to the manager provided that the network manager was specified during the SNMP component installation and the network manager was started before catrapd was activated. Any received trap is unpacked (decoded) and sent to the Event Management console log where it is recorded. At the console log, the trap is also available for subsequent automatic mapping, interpretation, and action processing as defined by your Event Management policies.

Valid on UNIX/Linux, z/OS

If catrapd is activated on a node that does not have a network manager or when the network manager is not active, catrapd connects directly to the internet service port designated for SNMP-trap (for z/OS, port 161 is customarily used by catrapd to listen for traps). Whatever the number of the port assigned, the protocol for the internet service port must be defined as UDP.

The usual way to start catrapd on zOS is by using the command:

```
unicntrl start snmp
```

Uncomment this line in the $CAIGLBL0000/opr/scripts/emstart script in order to start catrapd when the other Event Management components are started.

The catrapd daemon makes use of the system call getservbyname() to determine the actual port number to use. If this service has been modified, catrapd may not be able to locate the correct port. Most operating systems with TCP/IP support use an /etc/services file, which can be consulted to determine the port number reserved for the SNMP trap on that machine. On z/OS the TCP/IP stack's PROFILE file usually contains information on reserved ports. If port 161 is unavailable, you may choose a different port and update file $CAIGLBL0000/snmp/scripts/envset to point to the chosen port as follows:

port number to listen on

CAICATD0001=9161

export CAICATD0001

Example

As an example, the catrapd daemon receives a trap and presents it to the Event Management function in the following format:

```
.CATD_I_060 SNMPTRAP:  -c  public  Computer.Associates 999.999.9.999 xyzlhu33 6 1
00:00:00 1 OID: 1.3.6.1.4.1.791.1.4
.iso.org.dod.internet.private.enterprises.791.1.4 VALUE: hello world
```

Within the Event Management function of Enterprise Management, this message is divided into character "tokens," which can be tested individually or in combination with one another as part of user-defined Event Management policies. The message in the preceding example has 17 tokens. The tokens are:

Token	Value	Description
&1	CATD_I_060	CA standard prefix.
&2	SNMPTRAP:	Constant literal indicating this message originated as SNMP trap.
&3	-c	Indicates the beginning of the community parameter.
&4	public	The community type.
&5	Computer.Associates	The enterprise name, if found in the %CAIGLBL0000%\DB\enterprise.dat file (Windows) or $CAIGLBL0000/snmp/dat/enterprise.dat (UNIX/Linux, z/OS). If the enterprise ID is not found in the enterprise.dat file, the numeric value is printed.

Token	Value	Description
&6	999.999.99.999	The IP address of the node where the trap originated.
&7	xyzlhu33	The node name of the originating machine.
&8	6	As specified in the catrap command, a generic trap of 6 indicates an Enterprise-specific code has been assigned.
&9	1	As specified in the catrap command, the number of the specific trap.
&10	00:00:00	The time stamp from the catrap command, displayed in hh:mm:ss.
&11	1	The sequence number of the variable binding information from the catrap command.
&12	OID	Constant literal that describes the object that was the subject of this trap.
&13	1.3.6.1.4.1.791.1.4	The object identifier of the object that was the subject of this trap.
&14	iso.org.dod.internet. private.enterprises.791.1.4	Enterprise ID organization.
&15	VALUE	Constant literal.
&16	hello	A value as specified in the catrap command.
&17	world	A value as specified in the catrap command.

cawto-Send a Message to the Console

Valid on UNIX/Linux, Windows, z/OS

Use this command to send a message to the Windows console or the system console without waiting for a reply. To send a message and wait for a reply, use the Event Management cawtor command.

cawto (UNIX/Linux, z/OS)

This command has the following format:

```
cawto [-a attribute]
      [-c color]
      [-g category]
      [-k]
      [-n node]
      [-s source]
      text
```

External Influences

CAI_WTOR_NODE

Environment variable you can set to an alternate node. If this variable is set and cawto is issued without a node specification, the value of CAI_WTOR_NODE is used.

Files

/usr/tmp/cawto.n

If either the Event Management daemon or CAICCI is down, these files are created so that the cawto and careply commands can continue to function normally. The value of n is the message number the cawto command generated. The message number is identical to the process issuing the cawto command when either the Event Management daemon or CAICCI is down.

Example

The following example alerts the console operator on node 23 that the node is not accepting FTP connections, and asks for action to be taken. The message will be displayed in red and will remain on the node 23 system console as a kept message until released by the node 23 operator.

```
cawto -k -n 23 -c red node23 is not accepting ftp -- please investigate
```

Parameters used in this example:

-k keep

This keep option specifies that the message should be displayed as a kept message on the system console. The message can then be removed from the kept message display.

-n node

Node to which the message is directed which is one other than the node the user is on. If the node is not available, the message is sent to the local node.

-c *color*

Indicates that the message should be displayed on the system console in the identified color, which is other than the default. Supported colors are:

Default (black), Red, Orange, Yellow, Green, Blue, Pink, and Purple

Note**:** The color name can be specified in upper, lower, or mixed case.

-a *attribute*

Indicates that the message should be displayed on the system console in the identified attribute, which is other than the default. Supported attributes are:

DEFAULT

BLINK

REVERSE

Note: The attribute name can be specified in upper, lower, or mixed case

-s *source*

Assigns the identified source to the message for purposes of message matching.

-g *category*

Assigns the identified category to the message for purposes of message matching.

node

Node to which the message is to be directed. If the node is not specified, the message will be sent to the local node. If CAI_WTOR_NODE is specified, the message will be sent to the node that corresponds to CAI_WTOR_NODE.

text

Specify from 1 to 255 alphanumeric bytes of message text.

Note: Since the cawto command is issued from the command line, the text string you specify is subject to evaluation by the shell you are executing. Therefore, imbedded blanks, special characters, and quotes in the text string require special consideration. See the documentation provided with your operating system for the rules and guidelines regarding text strings.

cawto (Windows only)

This command has the following format:

```
cawto   [options] message-text

        -cat/-g category
        -i invisible
        -msg msgnum
        -s/-sou source
        -sev/-v severity
        text
        [-a attribute]
        [-c color]
        [-k keep]
        [-l]
        [-n node]
        [-s source]
```

Environment Variables

CA_OPERA_NODE

Configuration variable that can be set to an alternate node. If this variable is set and cawto is issued without a node specification, the value of CA_OPERA_NODE is used.

-l

Event should be logged without matching against defined policy records.

-msg *msgnum*

This is a numeric value associated with the message (event) that names the message. It is part of the prefix for CA NSM messages. For example: CASH_999_W where 999 is the message number.

Value: numeric 1 - 99999999

Default: no message number

-s/-sou *source*

Identifies the application that is the source of the message (event). For example: CASH_999_W, where CASH is the source of the event and CASH=Workload.

Value: character string, 1-255

Default: none

-sev/-v *severity*

Severity of the message (event). When viewing events with severity codes in the GUI console, different icons appear to the left of the events indicating the severity status.

Value: 1-3 characters. CA recommends the following to ensure that icons appear:

I - Informational
S - Success
W - Warning
E - Error
F - Failure

Default: none

-cat/-g *category*

The category associated with the event.

Value: character string, 1-255

Default: none

cawtor-Send a Message to the Console and Wait for Reply

Valid on UNIX/Linux, Windows, z/OS

The Event Management cawtor command sends a message to the system console and waits for a reply.

Messages sent to the system console using the cawtor command remain in a "HELD Message Area" until acknowledged by an operator using the console GUI or in response to an autoreply message action or a careply command. Typically, the number of messages that are "reply pending" or "acknowledge pending" are few in number and are viewable at one time. When the number of messages held exceed the number that can be concurrently viewed, scroll bars can be used to view the remainder of the held messages.

These messages are removed from the system console only when the console operator uses the Event Management careply command responding to the message or by defining a message record (cautil MSGRECORD) with an action of AUTOREPLY.

Replies sent with careply are written to the standard output of the process issuing the cawtor command. When issued, the cawtor command does not complete until a reply is received from the console operator.

cawtor (UNIX/Linux, z/OS)

This command has the following format:

```
cawtor [-a attribute]
       [-c color]
       [-g category]
       [-n node]
       [-q]
       [-qn node]
       [-s source]
       text
```

External Influences

CAI_WTOR_NODE

Environment variable that you can set to an alternate node. If this variable is set and cawtor is issued without a node specification, the value of CAI_WTOR_NODE is used.

Files

/usr/tmp/cawtor.n

If the Event Management daemon or CAICCI is down, these files are created so that the cawtor and careply commands can continue to function normally. The value of n is the message number the cawtor command generated. Also, the message number is identical to the process issuing the cawtor command when the Event Management daemon or CAICCI is down.

Example

When executed, this shell script prompts the console operator for a specific date. The date contained in the operator reply is written to standard output and assigned to the variable ATBDATE by the shell. The program atb then uses the value of the variable ATBDATE.

```
ATBDATE=$(cawtor -q Enter the starting date for the aged trial balance) atb $ATBDATE
```

Note: ATBDATE is a user-defined program.

To specify both quiet and alternate node, use this syntax:

```
cawtor -qn node text
```

-k

This option specifies that the message should be displayed as a kept message on the system console. The message can then be removed from the kept message display.

-c *color*

Indicates that the message should be displayed on the system console in the identified color, which is other than the default. Supported colors are:

Default (black), Red, Orange, Yellow, Green, Blue, Pink, and Purple

Note: The color name can be specified in upper-, lower-, or mixed case.

-a *attribute*

Indicates that the message should be displayed on the system console in the identified attribute, which is other than the default. Supported attributes are:

DEFAULT

BLINK

REVERSE

Note: The attribute name can be specified in upper, lower, or mixed case.

-s *source*

Assigns the identified source to the message for purposes of message matching.

-g *category*

Assigns the identified category to the message for purposes of message matching.

-q

Quiet option. Indicates that the message reply is not echoed on the terminal of the user who generated the message.

-qn *node*

Identifies the target node to which to direct the message or command being specified. -qn specifies that the message or reply instructions will not be displayed on the terminal of the user who generated the message.

[*node*] of cawtor command

Node to which the message is to be directed. If the node is not specified, the message will be sent to the local node. If CA_WTOR_NODE is specified, the message will be sent to the node that corresponds to CA_WTOR_NODE.

Note: If node is specified, you *must include the brackets* [ ] as part of the command.

-n *node*

Node where the message is directed. If the node is not available, the message is sent to the local node.

text

1- to 255-alphanumeric bytes of message text.

Note: Since the cawtor command is issued from the command line, the text string you specify is subject to evaluation by the shell you are executing. Therefore, imbedded blanks, special characters, and quotes in the text string require special consideration. See the documentation provided with your operating system for the rules and guidelines regarding text strings.

cawtor (Windows only)

This command has the following format:

```
cawtor [-q quiet]
       [-a attribute]
       [-c color]
       [-g category]
       [-n node]
       text
       [-msg msgnum=nnn]
       [-s/-sou source]
       [-sev/-v severity]
       [-cat/-g category]
       [-i invisible]
```

Environment Variables

CA_OPERA_NODE

Configuration variable that can be set to an alternate node. If this variable is set and cawtor is issued without a node specification, the value of CA_OPERA_NODE is used.

Before a command, reply, or acknowledgement is executed from the Console GUI using the oprcmd, careply, or delkeep commands, or by a message action, an authorization check is performed to verify that the user ID can perform this action.

If CA NSM security is inactive, the user ID issuing the command must be listed in the Users authorized to issue commands environment variable. This list is a setting that is edited in the Configuration Settings notebook, on the Event Management Preferences page.

If CA NSM security is active, the user ID must have explicit permission to execute that command. To do this, a rule must be created for the asset type CA-CONSOLE-COMMAND.

Note: Wild cards can be used. See the online help for the user profile--Asset Access Management dialog for more details.

Example

cawtor is part of a .bat file that sends a request to the operator to mount a tape. The message "TAPE: please mount today's Order Processing tape" is sent to the console and put in the held area. Once the tape is mounted and the operator has replied to the message, the ordproc program (user program) is started.

```
Cawtor TAPE: please mount today's Order Processing tape
ordproc
```

delkeep-Acknowledge Held Console Messages

Valid on UNIX/Linux, Windows, z/OS

Acknowledges held console messages, thereby removing them from the held message display. This command is an alternative to similar functionality performed using the console GUI.

Note: Only one message is acknowledged at a time.

This command has the following format:

```
delkeep  [options] message-text
```

Options

-a

Deletes all kept messages.

-n node

The CAICCI sysid of the machine where the command should be sent for processing.

message-text

The text, or partial text with wild cards, of the held message to be removed from the display. On UNIX/Linux, the message text must be enclosed in double quotes.

Example

```
delkeep INFO0001W*
```

This example removes from the help display all messages beginning with the text "INFO0001W."

Before a command, reply, or acknowledgement is executed from the Console GUI using the oprcmd, careply, or delkeep commands, or by a message action, An authorization check is performed to verify that the user ID can perform this action.

If CA NSM security is inactive, the user ID issuing the command must be listed in the Users authorized to issue commands environment variable. This list is a setting that is edited in the Configuration Settings notebook, on the Event Management Preferences page.

If CA NSM security is active, the user ID must have explicit permission to execute that command. To do this, a rule must be created for the asset type CA-CONSOLE-COMMAND.

Note: Wild cards can be used. See the online help for the user profile--Asset Access Management dialog for more details.

oprcmd-Pass Command to Event Management Service

Valid on UNIX/Linux, Windows, z/OS

Use the oprcmd command to send a command or configuration variable value to the Event Management service. The Event Management service executes your request.

When Event Management starts, it reads all message action and message record definitions from the Enterprise Management database and creates active message action and message record lists.

Subsequent changes to the database are not automatically reflected in the active message record and message action lists used by the Event Management service. You can use the oprcmd command to direct the Event Management service to refresh the active message record and message action lists with the definitions stored in the Enterprise Management database (oprcmd opreload).

In addition, you can use the oprcmd command to shut down the Event Management service.

Example

```
oprcmd opreload
```

Directs the Event Management service to refresh the active message record and message action lists with the definitions stored in the Enterprise Management database.

oprcmd (UNIX/Linux, z/OS)

This command has the following format:

```
oprcmd [-n node]
       [-c color]
       [-a attribute]
       [-s source]
       [-g category]
       {opreload|caistop|cmd|export variable=value|emexit exit=value|emexit
status}
```

-n *node*

Directs the message to the identified node, which is one other than the node the user is on. If the node is not available, the message is sent to the local node.

-c *color*

Indicates that the message should be displayed on the system console in the identified color, which is other than the default. Supported colors are:

Default (black), Red, Orange, Yellow, Green, Blue, Pink, and Purple

-a *attribute*

Indicates that the message should be displayed on the system console in the identified attribute, which is other than the default. Supported attributes are:

DEFAULT

BLINK

REVERSE

Note: The attribute name can be specified in upper, lower, or mixed case.

-s *source*

Assigns the identified source to the message for purposes of message matching.

Note: The color name can be specified in upper, lower, or mixed case.

-g *category*

Assigns the identified category to the message for purposes of message matching.

opreload

Directs the Event Management daemon to refresh the active message record and message action lists with the definitions stored in the Enterprise Management database. All requests for automated message processing are queued until the refresh of the active message record and message action lists is complete.

caistop

Shuts down the Event Management daemon (equivalent to kill -15).

cmd

Specify the command request, up to 255 alphanumeric bytes.

Note: Since the oprcmd command is issued from the command line, the *cmd* string you specify is subject to evaluation by the shell you are executing. Therefore, embedded blanks, special characters, and quotes in the text string require special consideration. See the documentation provided with your operating system for the rules and guidelines regarding text strings.

export *variablename=value*

Dynamically changes the value of any environment variable. This operand is equivalent to the Korn shell command, export.

emexit *exit=value*

Sets the value of the enabling environment variable for the identified Event Management user exit to either ON or OFF. The available exit names are:

Exit	Environment Variable	User Exit Function Name
MSGPRE	EMEVTENV_MSGPRE	EmEvt_ExitMessagePre
MSGPOST	EMEVTENV_MSGPOST	EmEvt_ExitMessagePost
ACTPRE	EMEVTENV_ACTPRE	EmEvt_ExitActionPre
ACTPOST	EMEVTENV_ACTPOST	EmEvt_ExitActionPost
LOGPRE	EMEVTENV_LOGPRE	EmEvt_ExitLogPre
LOGPOST	EMEVTENV_LOGPOST	EmEvt_ExitLogPost
SYSINIT	EMEVTENV_SYSINIT	EmEvt_ExitSystemInit
SYSTERM	EMEVTENV_SYSTERM	EmEvt_ExitSystemTerm

emexit status

Causes the settings of all of the enabling environment variables for the Event Management user exits to be displayed on the system console.

oprcmd (Windows only)

This command has the following format:

```
oprcmd [-n node]
       [-c color]
       [-a attribute]
       [-s source]
       [-g category]
         command
```

Notes

Before a command, reply, or acknowledgement is executed from the Console GUI using the oprcmd, careply, or delkeep commands, or by a message action, an authorization check is performed to verify that the user ID can perform this action.

If CA NSM security is inactive, the user ID issuing the command must be listed in the Users authorized to issue commands environment variable. This list is a setting that is edited in the Configuration Settings notebook, on the Event Management Preferences page.

If CA NSM security is active, the user ID must have explicit permission to execute that command. To do this, a rule must be created for the asset type CA-CONSOLE-COMMAND.

Note: Wild cards can be used. See the online help for the user profile--Asset Access Management dialog for more details.

Command Prefixes

A number of prefixes can be affixed to an oprcmd command. These prefixes control the behavior of the command, for example, interaction with the desktop, priority, or window appearance.

This command uses the following syntax:

```
["title"] [/Dpath] [/MIN] [/MAX] [/SEParate] [/LOw] [/NORMal] [/HIgh] [/REALtime]
[/INTeractive] [/DETached] oprcmd [options]
```

/Dpath

The startup directory. The path must be supplied immediately after the prefix. For example, to use the C drive as the startup directory, enter: /Dpathc:\

/MIN

Starts the window minimized.

/MAX

Starts the window maximized.

/SEParate

Starts 16-bit Windows programs in a separate memory space.

/Low

Starts the application in the idle priority class.

/NORMal

Starts the application in the normal priority class. This is the default priority.

/High

Starts the application in the high priority class.

/REALtime

Starts the application in the realtime priority class.

/INTeractive

Starts the application with access to the Windows desktop. Enterprise Management is configured to run as an NT Service, which, by default, does not have access to the desktop. Therefore, commands that interact with the user (that is, the GUI) will be invisible even though the process seems to be running (for example, CALC, Visual Basic applications, notepad).

/DETached

The new process does not have console/desktop access. This is the default.

Options

Refers to the same options explained in the oprcmd syntax.

Command

Any built-in Windows command, installed or user programs, and the following Event Management commands:

CAISTOP-Stops Event Manager daemon (use unicntrl stop opr)

EMEXIT-Displays and controls DLL exits and individual exits

EMLIB-Displays DLL information

EMUSER-Displays or logs off user logged on by Event Manager

OPRDUMP-Displays defined message records and message action on console

OPRELOAD-Reloads Event Management policies database

OPRSET-Sets internal environment variables

REPLID-Displays messages waiting for operator reply

TRACE_ON-Enables trace

TRACE_OFF-Disables trace

EMUSER[status|logoff *userid-mask*]

status

Displays user logged on by Event Manager as a result of command execution.

logoff userid-mask

Logs off the user ID logged on by Event Manager that matches the user ID mask (wild cards permitted)

EMEXIT[status|Exitname=ON|OFF]

status

Displays information about the DLL exits and individual exits.

Exitname=ON|OFF

Enables/disables execution of specified exit.

EMLIB[status|free *lib-mask*]

status

Displays information about DLLs loaded as a result of EXTERNAL actions.

Free

Unloads DLLs.

Prefix Examples

```
/INT /MAX /Dd:\ notepad doclist.txt
```

This command will start the notepad application with the doclist.txt parameter. The program will be interactive or visible (/INT), the window will be maximized (/MAX), and the startup directory is d:\ (/Dd:\).

```
/LOW zzprog
```

This command will run zzprog as a low priority process.

```
"App1" /HI cmd
```

This will start cmd as a high priority process with a title of App1.

```
"title"
```

The title to display in the window title bar. This applies to console-based/non-GUI applications. The title text must be enclosed in double-quotes ("*text*").

Command Execution

By default, commands are executed under the host user ID (the user ID associated with Enterprise Management--CAUNINT, by default). When a command is executed through Event Management, it can be run in one of three different environments. The run environment is set in the CA NSM Configuration Setting User context for running commands found in Event Management Preferences.

Commands that expect to execute under a specific user ID or under a user ID with specific authorizations might fail or not run properly. For example, IDs that need access to network drives can be restricted or not available; not all IDs can access printers; and so forth. This means you will not be able to print using CAUTIL LIST if a printer is not defined for your ID. By default, no printers are defined for CAUNINT. To enable print access for CAUNINT, you must log on as CAUNINT and define a default printer.

Note: If Enterprise Management is configured to run under the LocalSystem account, you cannot access network drives without changes to the registry. See the Microsoft Knowledge Base under "Null Session" for more information.

To be able to run commands under a user environment other than CAUNINT, see Defining the Environment for Command Execution.

oprdb-Maintain the Event Management Database

Valid on UNIX/Linux, Windows, z/OS

Use the oprdb command to save, list, or convert the data in the Event Management database or database image (DSB) file. "Converting" data means that the binary data contained in the database is converted into a character command representation, which can be used as input to the CAUTIL command utility to restore the database or reproduce its contents in another database.

This command has the following format:

```
OPRDB  [LIST {DB|dsb-file path}]
```

Valid on Windows

```
[LOAD dsb-file]
[SAVE dsb-file path]
[SCRIPT {DB|dsb-file path|>mycmds.txt|cautil -f mycmds.txt}]
```

Valid on UNIX/Linux, z/OS

```
[SCRIPT {DB|>mycmds.txt|cautil -f mycmds.txt}]
```

Examples

This example lists, in report form, the contents of DSB file c:\TNG\logs\caopr.dsb. In this example, this is the file that is used to save the Event Management database each time Enterprise Management is started.

```
oprdb list c:\TNG\logs\caopr.dsb
```

LOAD dsb-file

Valid on Windows

Loads the contents of the *dsb-file* into the Event Management database. The previous contents of the database is lost unless OPRDB SAVE *dsb-file* was previously issued. To put this new policy into effect, issue the opreload command.

SAVE dsb-file path

Valid on Windows

Saves the Event Management database messages and actions to the file specified (*dsb-file path*) for subsequent use as an alternative to the database should it become unavailable for any reason.

LIST {DB|*dsb-file path*}

Identifies the number of retries to be made when a busy or other recoverable error condition is detected while attempting to send the SNMP trap.

oprfix-Repair a Corrupted Event Management Log File

Valid on UNIX/Linux, Windows, z/OS

Use the oprfix command to repair a corrupted Event Management log file. Log file corruption can sometimes occur when the file system becomes full. As a result, only part of a log record can be written out.

Log file corruption can also occur if the system is not shut down in an orderly fashion (this is much less likely to occur).

The oprfix command returns a 0 to indicate success and a 1 to indicate failure in repairing the log file.

This command has the following format:

```
oprfix file-specification
```

Valid on Windows

```
oprfix %CAIGLBL0000%\logs\19970730
```

Valid on UNIX/Linux, z/OS

```
oprfix $CAIGLBL0000/opr/logs/oplog.19970809
```

oprping-Test Remote Event Management Availability

Valid on UNIX/Linux, Windows, z/OS

Use the oprping command to test whether Event Management is available on another node. Informs the user whether the specified node is up and available for communication, and whether the Event Manager is running on the node and is responding to messages.

This command has the following format:

Valid on Windows

```
oprping nodename
        times
        msg
```

Valid on UNIX, Linux, z/OS

```
oprping <nodename>
```

Examples

```
oprping USLASRV3 3 this is only a test.
```

Attempts to send the text "this is only a test" to node USLASRV3 event daemon three times. The command will produce messages substantiating the success or failure of the remote node daemon.

nodename

The Hostname of the node.

times

Specifies the number of times the test message is to be sent to the node (not applicable on z/OS).

msg

The message text to be sent to the node (not applicable on z/OS).

Console Commands for CAS9FEMS

The CAS9FEMS job or started task running under z/OS can accept and process the following modify commands:

CAS9FEMS,APPL=STATUS

Returns the status of the CAS9FEMS job including the main task process ID.

CAS9FEMS,APPL=DEBUGON

Sets debugging on.

CAS9FEMS,APPL=DEBUGOFF

Sets debugging off.

CAS9FEMS,APPL=SHUTDOWN

Performs orderly shutdown of CAS9FEMS.

Submit a Command to z/OS with CAconsole

Valid on z/OS

Use the Event Management CAconsole command at any terminal to submit a command or commands to z/OS and ascertain the results of those requests. Using CAconsole, the result of a z/OS command, which is normally displayed on the z/OS console or recorded in the z/OS system log, instead is returned to the CAconsole application and can optionally be captured in an application log.

Note: The choice of z/OS commands is not limited to those supported by IBM but also includes all interface commands that may be supported by any z/OS application program.

This command has the following format:

```
CAconsole [-Ttimeout] [-X] [-V] [cmdseq|<cmdfile] [>logfile]
```

-T*timeout*

The number of seconds to wait for a response to the z/OS command. If no timeout is specified, CAconsole continues to wait for a response.

-X

Transforms the z/OS command to uppercase before sending it to the z/OS Operator Messaging Facility.

-V

Retains the result of the command and directs it either to standard output or to a log file, if one is specified.

cmdseq

The z/OS command to send.

<cmdfile

A file containing the z/OS command or commands to send. If neither a command nor a command file is specified, the command is taken from standard input.

>logfile

Name of a file to which the result of the command is directed. If none is specified, the result is sent to standard output.

Examples

The command D TS,L is sent to the z/OS Operator Messaging Facility. The result of the command will be sent to standard output. -T specifies a timeout value of 1234 seconds, after which CAconsole will consider that **no** output was received.

```
CAconsole -T1234 -V D TS,L
```

The command D TS,L (after translation to uppercase) is sent to the z/OS Operator Messaging Facility. The result of the command will be sent to standard output. -T specifies a timeout value of 1234 seconds, after which CAconsole will consider that **no** output was received.

```
CAconsole -T1234 -V -X d ts,l
```

The file mycmdfile.txt will be examined and each line within it will be considered a separate command that is to be sent to the z/OS Operator Messaging Facility. The result of each command will be sent to standard output. No timeout is specified, indicating that each command sequence will result in a theoretical infinite wait for a response.

```
CAconsole  -V <mycmdfile.txt
```

The file mycmdfile.txt will be examined and each line within it will be considered a separate command that is to be sent to the z/OS Operator Messaging Facility. Each command line sequence will be translated to uppercase before it is sent. The result of each command will be sent to standard output. The CAconsole facility will wait up to 60 seconds before considering that no output was received from that command and then move on to the next command line within the file specified.

```
CAconsole  -V -X -T60 <mycmdfile.txt
```

The file mycmdfile.txt will be examined and each line within it will be considered a separate command that is to be sent to the z/OS Operator Messaging Facility. Each command line sequence will be translated to uppercase before it is sent. The CAconsole facility will wait up to 60 seconds before considering that no output was received from that command and then move on to the next command line within the file specified. The output will be directed to the file mylog.

```
CAconsole -V -X -T60 <mycmdfile.txt >mylog
```

Optional Batch Operation

CAconsole for z/OS can run as a standalone service using conventional z/OS JCL. The JCL should resemble this example:

```
//CACONSO JOB  . . .
 //$$$$$$@  EXEC PGM=BPXBATCH,
//        PARM='sh /u/users/framewrk/CAconsole -T1234 -V D TS,L '
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Applicable parameters are -T, -X, -V, and cmdseq; all are optional. If no command is specified, the input is derived from standard input.

Chapter 2: Agent Technology Commands

This chapter details the Agent Technology commands pertinent to the z/OS implementation. It includes information specific to the various platforms supported by CA Common Services.

On z/OS, most of these commands exist in two versions. One exists in the zFS system and another, with the same capabilities, in a LOADLIB. Execute the version in the zFS system by entering the corresponding command from a shell console. Execute the LOADLIB version in batch mode, by submitting the equivalent JCL member from the SAMPJCL library.

This section contains the following topics:

z/OS Environment

All batch jobs for the tools and utilities reside in the *prefix*.MFNSM.CAW0JCL library. The prototype JCL looks like this:

```
//jobname  JOB  40200000
// JCLLIB ORDER=prefix.MFNSM.CAW0JCL
// INCLUDE MEMBER=CCSVARS
//*
//stepname EXEC PGM=program,PARM='/ options'
//STEPLIB  DD  DISP=SHR,DSN=&CAI.MFNSM.CAW0LOAD.
//         DD  DISP=SHR,DSN=&LE37PREF.SCEERUN
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//ENVFILE  DD  DISP=SHR,DSN=&CAI.MFNSM.CAW0OPTV
 //SYSTCPD  DD  DISP=SHR,DSN=&TCPDATA
```

Notes:

- The PARM parameter should start with a slash character when it is needed to specify options for the program run. This character normally marks the separation between the run-time parameters and the program parameters.
- When you run in batch mode, the JES SYSOUT file contains the job output.

agentctrl-Agent Technology Control Utility

Valid on UNIX/Linux, Windows, z/OS

agentctrl sends Agent Technology control messages to an agent.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

prefix.CAW0JCL(AGENTCTL)

This command has the following format:

```
agentctrl [-h hostname]  [-s] [-c] mibName:agentName ctrlMessage  [-a | -m]
```

-h *hostname*

Specifies the host name or IP address of the system on which the agent is running.

Default: local node

-s

Extracts the information from the store, not the agent.

-c

Prints in ConfigSet (ldconfig utility) format.

mibName

Specifies the top level OID in the agent MIB (as registered with the setup registration function AWF_RegName). This value is the name of the MIB you want to browse. This name is often (but not always) the same as the agent name.

agentName

Specifies the name of the agent to send the control message to (as registered with the setup registration function AWF_RegName, usually the agent executable name).

ctrlMessage

Specifies the control message text string.

-a

Lists all objects (services and agents) registered with aws_orb.

-m

Lists all MIBs loaded in the store.

The following list shows all the control messages that can be sent to the agent.

CONTROL:AgentExit

Exits the agent.

CONTROL:AgentPause

Pauses the agent.

CONTROL:AgentResume

Resumes the agent.

CONTROL:AgentRestart

Exits the agent and automatically restarts it. A restart command must have been registered.

CONTROL:TaskPause:*taskName*

Pauses the specified task.

CONTROL:TaskResume:*taskName*

Resumes the specified task.

CONTROL:TaskDelete:*taskName*

Removes the task from the scheduled task list.

CONTROL:TaskInterval:*taskName*:*interval*

Changes the interval of the specified task.

CONTROL:StoreAgent

Stores all the agent's MIB values in Object Store.

CONTROL:RestoreAgent

Restores all the agent's MIB values from Object Store.

CONTROL:RemoveAgent

Removes all the agent's MIB values from Object Store.

CONTROL:StoreConfigSet:*configSetName:*
objectName

Stores the values in a group or table as a configuration set in Object Store.

CONTROL:RestoreConfigSet:*configSetName*

Restores the values in a configuration set into the agent's memory.

CONTROL:RemoveConfigSet:*configSetName*

Removes the values in a configuration set from Object Store.

CONTROL:StoreGroup:*groupName*

Stores all the attribute values for the specified group.

CONTROL:RestoreGroup:*groupName*

Restores all the attribute values for the specified group.

CONTROL:RemoveGroup:*groupName*

Removes all the attribute values for the specified group.

CONTROL:StoreTable:*tableName*

Stores all the attribute values for the specified table.

CONTROL:RestoreTable:*tableName*

Restores all the attribute values for the specified table.

CONTROL:RemoveTable:*tableName*

Removes all the attribute values for the specified table.

CONTROL:StoreVariable:*varName*

Stores the variable value for the specified variable.

CONTROL:RestoreVariable:*varName*

Restores the variable value for the specified variable.

CONTROL:RemoveVariable:*varName*

Removes the variable value for the specified variable.

PRINT:Table:*tblName*[:*fileName*]

Prints all attribute values for the specified table.

PRINT:Group:*grpName*[:*fileName*]

Prints all attribute values for the specified group.

PRINT:Registrations[:*fileName*]

Prints all registrations.

PRINT:Variable[:*fileName*]

Prints all variables.

PRINT:Agent[:*fileName*]

Prints all attribute values for the agent's entire MIB.

PRINT:ConfigSet:*configSetName*[:*fileName*]

Prints from Object Store all the attribute values for the specified configuration set.

LOGMSG:LogStart:*filename*

Starts logging into the specified file.

LOGMSG:LogStop

Stops logging.

LOGMSG:SetPattern:*pattern*

Filters logging on the specified pattern.

LOGMSG:DelPattern:*pattern*

Deletes the specified filter pattern.

LOGMSG:ListPatterns

Lists all patterns being filtered on.

LOGMSG:SetLevel:*level*

Sets the logging level.

LOGMSG:SetFile:*filename*

Sets the log file.

LOGMSG:GetFile

Returns the name of the log file.

LOGMSG:GetLevel

Returns the current log level.

GETATTR: *parName*:*varName*:*index*

Gets the specified attribute value.

SETATTR: *parName*:*varName*:*index*:*value*

Sets the specified attribute.

ADDATTR: *parName*:*varName*:*index*:*value*

Adds an instance of the specified attribute.

GETVAR: *varName*

Gets the specified variable value.

SETVAR: *varName*:*value*

Sets the specified variable.

awbulk-Retrieve MIB Attribute Values Using GetBulk

Valid on UNIX/Linux, Windows, z/OS

awbulk retrieves MIB attribute values using GetBulk requests (available with SNMP V2C and higher), as specified in RFC 1905. If you use the -o option, the attribute's OID is printed followed by the OID value.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

This command has the following format:

```
awbulk [-h hostname] [-p port] [-c community] [-v 1|2c|3] [-u user]
[-a authPassword] [-x privPassword] [-s secLevel] [-P DES] [-A MD5|SHA]
[-n contextName] [-r n_repetitions] [-t timeout] [-d level] [-f logFile] [-o] OID
```

-h *hostname*

Specifies the host name or IP address of the system on which the agent is running.

Default: local node

-p *port*

Specifies a name in the /etc/services file that translates to a port number, or the number of the UDP port where the agent listens for incoming requests. This value must match the one used by the SNMP Administrator.

Default: 161

-c *community*

Specifies the name of any valid community in the agent's configuration. If you want to retrieve the MIB attribute value from a specific instance of an agent, you must postfix the community string with the @ sign, followed by the specific agent instance name.

Default: public

-v *1|2c|3*

SNMP version.

-u *user*

Specifies the User ID that owns the security (valid with SNMP V3 only).

-a *authPassword*

Specifies the Auth password (SNMP V3 only).

-x *privPassword*

Specifies the Priv password (SNMP V3 only).

-s *secLevel*

Specifies the security level., one of the following:

1 NoAuthPriv (only Priv works)
2 AuthNoPriv (only Auth works)
3 AuthPriv (both work)

-P DES

Specifies the DES privacy encryption mechanism (SNMP V3 only). Requires specification of an authentication mechanism also (-A option).

-A MD5|SHA

Specifies which authentication mechanism to use (SNMP V3 only).

-n *contextName*

Specifies the MIBMUX instance name, for use with a MIBMUX (SNMP V3 only).

-r *n_repetitions*

Specifies the number of times the GetBulk request is repeated after the first request.

-t *timeout*

Specifies a maximum time value (in seconds) this program waits for a response to an SNMP request before exiting.

Default: 10 seconds

-d *level*

Debug level to use.

-f *logFile*

Name of the log file to use.

Default: stderr

-o *OID*

Specifies the object identifier of a specific instance of any attribute in the agent MIB.

awbulkwalk-Retrieve MIB Attribute Values Using GetBulk

Valid on UNIX/Linux, Windows, z/OS

awbulkwalk retrieves the value of every instance of every attribute defined in the MIB, from the specified OID through the last OID in the tree, inclusive, using GetBulk requests (available with SNMP V2C and higher), as specified in RFC 1905. awbulkwalk is the equivalent of repeated calls to awbulk. If you use the -o option, the attribute's OID is printed followed by the OID value.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

This command has the following format:

```
awbulkwalk [-h hostname] [-p port] [-c community] [-v 1|2c|3] [-u user]
[-a authPassword] [-x privPassword] [-s secLevel] [-P DES] [-A MD5|SHA]
[-n contextName] [-r n_repetitions] [-t timeout] [-d level] [-f logFile] [-o] OID
```

The option descriptions for awbulkwalk are identical to those for awbulk, with the following exceptions:

-r *n_repetitions*

This value maps to the max-repetitions parameter of the SNMP GetBulk command.

-o *OID*

Specifies the object identifier of any node in the agent's MIB where you want to begin traversing the tree.

awftest-Test Agent Technology Run-Time Environment

Valid on z/OS only

awftest is a collection of test functions designed to help control the environment, and to help debug problems that can arise when running Agent Technology for z/OS. awservices must not be running for this shell program or batch job to run successfully.

z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(AWFTEST)

This command has the following format:

```
awftest [functionName] [functionParameters]
```

functionName

To list all functions available with this utility, run awftest with the 'help' option. Functions are:

tcpip

Tests most of the TCP/IP socket functions that are used by Agent Technology for z/OS.

trapmon

Listens for traps and displays trap messages. For the full syntax of the trapmon function, run the awftest program with the 'help trapmon' option.

checkport

Checks the status of a TCP/IP and UDP port, or range of ports. For the full syntax of the checkport function, run the awftest program with the 'help checkport' option.

functionParameters

Specifies parameters available for checkport and trapmon.

awget-Retrieve a MIB Attribute Value

Valid on UNIX/Linux, Windows, z/OS

awget retrieves the value of the specified instance of a MIB attribute. If you use the -o option, the attribute's OID is printed followed by the OID value.

UNIX/Linux, z/OS

Many UNIX/Linux nodes have a MIB II-compliant SNMP agent built into the operating system, as does z/OS. These agents are hard coded to listen for manager requests on UDP port number 161. To prevent port conflicts between the operating system agent and your agent, the SNMP Administrator defaults to listening for requests on UDP port number 6665.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

prefix.MFNSM.CAW0JCL(AWGET)

This command has the following format:

```
awget [-h hostname] [-c community] [-p service] [-o] OID [-t timeout]
```

-h *hostname*

Specifies the host name or IP address of the system on which the agent is running.

Default: local node

-c *community*

Specifies the name of any valid community in the agent's configuration.

If you want to retrieve the MIB attribute value from a specific instance of an agent, then you must postfix the community string with the @ sign, followed by the specific agent instance name.

Default: public

-p *service*

Specifies a name in the /etc/services file that translates to a port number, or the number of the UDP port where the agent listens for incoming requests. This value must match that used by the SNMP Administrator.

Default: 6655 for z/OS

-o *OID*

Specifies the object identifier of a specific instance of any attribute in the agent MIB.

-t *timeout*

Specifies a maximum time value (in seconds) this program waits for a response to an SNMP request before exiting.

Default: 10 seconds

awm_catch-Message Display Utility

Valid on UNIX/Linux, Windows, z/OS

awm_catch displays messages being exchanged by the local agents and services. You must specify at least one message type and key with this utility.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

This command has the following format:

```
awm_catch messageType messageKey [-@ orbHostName] [-d debugLevel] [-f debugFile]
[-t=targetNode|targetName|targetClass] [-s sendType|sendKey|sendDate]
[-c maxNumberOfMessages] [-w waitSecond] [-F messageFilterString] [-q] [-n]
```

messageType

Identifies the pattern of message type to intercept and display, such as POLL_EVENT or FSM_CONFIG.

messageKey

Specifies the pattern of message key to listen for, such as SNMP, TRAP, or instance.

-@ orbHostName

Specifies the node name or IP address of a remote DSM.

-d *debugLevel*

Runs this utility in debug mode.

-f *debugFile*

Name of the debug file.

-t *targetNode|targetName|targetClass*

Specifies a particular recipient for a message.

-s *sendType|sendKey|sendData*

Enables the user to send one message before starting an awm_catch session. This prevents having an awm_config session open at the same time.

-c *maxNumberOfMessages*

Specifies the maximum number of messages to be displayed.

-w *waitSecond*

Specifies the number of seconds awm_catch listens for messages before exiting.

-F *messageFilterString*

Searches the type, key, and data of a caught message and displays the message only if the messageFilterString is found.

-q

Runs awm_catch in quiet mode.

-n

Displays the count number of the message at the front of each message.

awm_config-Send Messages to aws_orb

Valid on UNIX/Linux, Windows, z/OS

awm_config posts messages to the Distributed Services Bus (aws_orb). By using this command, you can make non-persistent, runtime changes to DSM policy and control DSM components. This command can be used with awm_catch for debugging purposes.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/Services/bin

This command has the following format:

```
awm_config [-@ orbHostName][-d debugLevel][-f debugFile][-t targetNode targetName targetClass][-s sendType sendKey sendData][ -w waitSecond][-c fileName]
```

-t targetNode targetName targetClass

Specifies the node, object name, and class to which this configuration information applies.

-s sendType sendKey sendData

Specifies message type, key, and data to send to the specified component.

-c fileName

Indicates that the DSM should read messages from the specified file.

awnext-Retrieve the Next Attribute after the MIB Attribute

Valid on UNIX/Linux, Windows, z/OS

awnext retrieves the next available instance of an attribute following the specified MIB attribute. If you use the -o option, the attribute's OID is printed followed by the OID value.

UNIX/Linux, z/OS

Many UNIX/Linux nodes have a MIB II-compliant SNMP agent built into the operating system. These agents are hard-coded to listen for manager requests on UDP port number 161. To prevent port conflicts between the operating system agent and your agent, the SNMP Administrator defaults to listening for requests on UDP port number 6665.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(AWNEXT)

This command has the following format:

awnext [-h *hostname*] [-c *community*] [-p *service*] [-o] *OID* [-t *timeout*]

-h *hostname*

Specifies the host name or IP address of the system on which the agent is running.

Default: local node

-c *community*

Specifies the name of any valid community in the agent's configuration.

If you want to retrieve the MIB attribute value from a specific instance of an agent, you need to postfix the community string with the @ sign, followed by the specific agent instance name.

Default: public

-p *service*

Specifies the name in the /etc/services file that translates to a port number, or the number of the UDP port where the agent listens for incoming requests. This value must match the one used by the SNMP Administrator, which is normally 6665.

Default: 161

-o *OID*

Specifies the object identifier of a specific instance of any attribute in the agent MIB.

-t *timeout*

Specifies the maximum time value (in seconds) this program waits for a response to an SNMP request before exiting.

Default: 10 seconds

awservices-Service Control Manager

Valid on UNIX/Linux, Windows, z/OS

The Service Control Manager (awservices) starts and stops all Agent Technology components and agents on the node in the correct order. Once this component is running, all other components can be installed, started, stopped, and/or uninstalled. Agent Technology dependent components are regulated by the Service Control Manager, thus preventing a component from being stopped that is essential to the operation of another component. Similarly the Service Control Manager also automatically starts certain components when a subordinate component is started.

z/OS

To start the services through the AWSTART batch job or PROC, ensure that the CLASS= parameter on the JOB statement is set to a high CPU limit job class. Although this job executes quickly, the services inherit the CPU time limit from this job.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0PROC(AWSTART)

&CAI.MFNSM.CAW0PROC(AWSTOP)

&CAI.MFNSM.CAW0JCL(AWSTATUS)

&CAI.MFNSM.CAW0JCL(AWLIST)

This command has the following format:

```
awservices {start|stop|list|version} [-m]
```

To start all Agent Technology components on the node in the correct order, enter the following command:

```
awservices start
```

-m

Starts awservices without starting any Agent Technology components.

awset-Set the Value of a MIB Attribute

Valid on UNIX/Linux, Windows, z/OS

awset sets the value of a MIB attribute. If you use the -o option, the attribute OID is printed followed by the OID value.

UNIX/Linux, z/OS

Many UNIX/Linux nodes have a MIB II-compliant SNMP agent built into the operating system. These agents are hard-coded to listen for manager requests on UDP port number 161. To prevent port conflicts between the operating system agent and your agent, the SNMP Administrator defaults to listening for requests on UDP port number 6665.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(AWSET)

Note: Your set request fails if the attribute is write-protected.

This command has the following format:

```
awset [-h hostname] -c community [-p service] [-o] OID type value [-t timeout]
```

-h *hostname*

Specifies the host name or IP address of the system on which the agent is running.

Default: local node

-c *community*

Specifies the name of any community supported for write access in the agent configuration.

If you want to retrieve the MIB attribute value from a specific instance of an agent, then you need to postfix the community string with the @ sign, followed by the specific agent's instance name.

Default: public

-p *service*

Specifies the name in the /etc/services file that translates to a port number, or the number of the UDP port where the agent listens for incoming requests. This value must match the one used by the SNMP Administrator, which is normally 6665.

Default: 161

-o *OID*

Specifies the object identifier of a specific instance of any attribute in the agent MIB.

type

Specifies one of the following options indicating the type of the specified attribute:

-i integer value

-o octet value

-s string value

-d object identifier (OID)

-a IP address

-c counter

-g gauge

-t time ticks

value

Specifies the new value of the attribute instance.

-t *timeout*

Specifies the maximum time value (in seconds) this program waits for a response to an SNMP request before exiting.

Default: 10 seconds

aws_orb-Distributed Services Bus

Valid on UNIX/Linux, Windows, z/OS

The Distributed Services Bus handles all processing overhead and routing of messages between agents, the common services, and the components of the Distributed State Machine (DSM).

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

This command has the following format:

```
aws_orb {install|remove|start|stop|status|debug|usage|version}
       [-d debugLevel]
       [-f logFile]
```

UNIX/Linux

```
       [-A AgentWorksDirectory]
```

aws_sadmin-SNMP Administrator

Valid on UNIX/Linux, Windows, z/OS

The SNMP Administrator checks the community string and internet protocol (IP) address of get, get-next, and set requests to make sure these requests come from authenticated management applications. This component forwards trap messages to the appropriate destinations. The SNMP Administrator stores configuration sets and MIB definitions for each agent on the node in memory. The SNMP Administrator also stores agent configuration information set during runtime.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

This command has the following format:

```
aws_sadmin {install|remove|start|stop|status|debug|usage|version}
       [-@ servicesHost]
       [-d debugLevel]
       [-f logFile]
       [-s]
       [-p port_number]
```

Valid on z/OS

```
[-x cache-size] [-y cache-size] [-z cache-size]
```

Example

To stop aws_sadmin and all running agents, leaving awservices, aws_orb, and any other Agent Technology components necessary to run the Distributed State Machine up, enter the following command:

```
aws_sadmin stop --andDependOn=yes
```

This command has the following format:

-s

Forces aws_sadmin to not use internal cache to save memory.

-x *cache-size*

Valid on z/OS

Specifies the cache size for the objects table in kilobytes (KB).

Default: 256

-y *cache-size*

Valid on z/OS

Specifies the cache size for the properties table in kilobytes (KB).

Default: 512

-z *cache-size*

Valid on z/OS

Specifies the cache size for the votree table in kilobytes (KB).

Default: 512

awtrap-Send an SNMP Trap PDU from one Node to Another

Valid on UNIX/Linux, Windows, z/OS

awtrap sends an SNMP trap protocol data unit (PDU) from the node you are on to another node. Use the awtrap utility to test a manager's ability to process an agent's traps without running the agent.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(AWTRAP)

This command has the following format:

`awtrap [-f from_addr] [-h dest_addr] [-p service] [-c community] enterprise type [subtype] [OID oidtype value]`

-f *from_addr*

Specifies the host name or IP address of the node sending the trap. If you want the trap to originate from another node, specify that node name or IP address.

Default: Local node

-h *dest_addr*

Specifies the host name or IP address of the node that is the destination of the trap.

Default: Local node

-p *service*

Specifies the name in the /etc/services file that translates to a port number, or the number of the UDP port where the application listens for traps. The SNMP standard port for traps is number 162.

Default: 162

-c *community*

Specifies the name of any valid community in the management application configuration. The community string can be any string that satisfies the community string syntax. This value is used without modification or validation. If the receiver of the trap is validating the community string when analyzing the trap, then you must specify a complete community string that the corresponding agent, which you are simulating, would normally send.

Default: Public

enterprise

Specifies the top-level OID in the MIB supported by the agent reporting the trap condition.

type

Specifies the generic trap type, as specified in the SNMP standard. Specify type 6 to send an enterprise-specific type.

subtype

Specifies the trap subtype for enterprise specific traps.

OID oidtype value

Specifies the following values:

OID

oidtype

The object identifier of a specific instance of any attribute in the agent's MIB.

Can be one of the following:

-i integer value

-o octet value

-s string value

-d object identifier (OID)

-a IP address

-c counter

-g gauge

-t time ticks

value

A value relevant to the trap condition. You can use up to 256 bytes. The combination of OID and value creates a unique OID/value pair for the variable bindings list in the trap PDU. You can specify only one OID/value pair using the awtrap utility.

awwalk-Retrieve Value of Every Attribute Defined in the MIB

Valid on UNIX/Linux, Windows, z/OS

awwalk retrieves the value of every instance of every attribute defined in the MIB, from the specified OID through the last OID in the tree, inclusive. Walking the MIB is the equivalent of repeated calls to awnext. If you use the -o option, the attribute OID is printed followed by the OID value.

This utility is most useful for obtaining all the values of the attributes in a MIB, group, table, or column. Specify the top (root) OID, group OID, table OID, or the OID of the first attribute in a column to see the values of all subordinate OIDs.

UNIX/Linux, z/OS

Many UNIX/Linux nodes have a MIB II-compliant SNMP agent built into the operating system, as does z/OS. These agents are hard coded to listen for manager requests on UDP port number 161. To prevent port conflicts between the operating system agent and your agent, the SNMP Administrator defaults to listening for requests on UDP port number 6665.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(AWWALK)

This command has the following format:

```
awwalk [-h hostname ] [-c community] [-p service] [-o] OID [-t timeout]
```

-h *hostName\ipaddr*

Specifies the host name or IP address of the node on which the agent is running.

Default: local node

-c *community*

Specifies the name of any valid community in the agent configuration.

If you want to retrieve the MIB attribute value from a specific instance of an agent, then you need to postfix the community string with the @ sign, followed by the specific agent's instance name.

Default: public, which is supported for read

-p *service*

Specifies the name in the /etc/services file that translates to a port number, or the number of the UDP port where the agent listens for incoming requests. This value must match the one used by the SNMP Administrator, which is normally 6665.

Default: 6665 for z/OS

-o *OID*

Specifies the object identifier of any node in the agent's MIB where you want to begin traversing the tree.

-t *timeout*

Specifies the maximum time value (in seconds) this program waits for a response to an SNMP request before exiting.

Default: 10 seconds

clean_sadmin-Remove Static Data and MIBs

Valid on UNIX/Linux, Windows, z/OS

clean_sadmin removes all static data and MIBs from the SNMP Administrator internal memory. Run this file on the common services host when you want to restore agent behavior to its initial state and reload MIBs into sadmin Store.

z/OS

For z/OS, once the clean_sadmin script file has been executed, you must reinstall all the MIBs you need at your site, using the install_mibs script file or, for z/OS only, the INSTMIBS batch job that has been customized during the Agent Technology for z/OS installation.

Windows

install-path\services\tools

UNIX/Linux

install-path/services/tools

z/OS JCL

&CAI.MFNSM.CAW0JCL(CLEANADM)

Note: This script clears all default state thresholds. To restore state thresholds, you must reload the MIBs and configuration files.

This command has the following syntax:

```
clean_sadmin
```

exagent-Control the Agent Technology Example Agent

Valid on Windows, z/OS

exagent starts and controls the Agent Factory example agent.

Note: See the Installation Guide for a description of how to build the example agent.

Windows

install-path\agents\samples\exagent

z/OS JCL

&CAI.MFNSM.CAW0JCL(EXAGNT)

This command has the following format:

```
exagent start [--instance=instName] [-t topDirectory] [-i interval] [-l log_level]
[-f logfile] [-s]
```

Sample z/OS JCL

This sample JCL requests the cmd /tmp instance of exagent to be started with /tmp as top directory under which all monitored directories must be found. It also requests a log level 4 with the log messages sent to the LOGFILE DD statement.

```
//stepname   EXEC PGM=EXAGENT,
 // PARM='/ debug --instance=/tmp -t /tmp -l 4 -f //DD:LOGFILE'
//LOGFILE  DD SYSOUT=*
```

Comments

If the --instance and -t parameters are not specified, then exagent will run without MIBMUX.

If you decide to run multiple instances of exagent, then specify the same names for the --instance and -t parameters in order to get consistent information displayed by the awservices status command and other commands such as orbctrl and agentctrl.

This command has the following options:

--instance=*instName*

Specifies the instance name as it will be known by awservices. This name will appear on the display of the awservices status command.

-t *topDirectory*

Specifies the top level directory under which all monitored directories must be found. This is the real instance name for the agent. It is used as instName parameter on the AWF_RegInstance function call.

-i *interval*

Specifies the interval for the main exagent task in seconds.

-f *logfile*

Specifies the name of the file where log messages are sent.

-l *log_level*

Specifies the message level for log messages.

-s

Uses the store instead of Register for Get Agent.

install_agents-Install or Remove Agents or Services

Valid on z/OS

This shell script can be used to add or remove entries in the awservices.cfg config file for any agents or services currently supported on z/OS.

z/OS

install-path/services/tools

This command has the following format:

`install_agents {install|installauto|remove}` *agent*

install

Adds the agent or service to the config file.

installauto

Adds the agent or service and specifies that it is to start up automatically when awservices is started.

Note: Not all agents can be started by awservices.

remove

Removes the agent or service from the config file.

agent

DB2

DB2 Agent

IDMS

CA IDMS Agent

CICS

CICS Agent

MQ

MQSeries Agent

OS390|zOS

z/OS System Agent

Memo

Agent for CA NSM System Status Manager CA-OPS/MVS Option

Exagent

Example Agent

aws_orb

Object Request Broker

aws_sadmin

SNMP Administrator

Note: If awservices is not active when this script is executed, it will be started automatically and stopped at the end of the run. If several agents or services have to be installed, it is recommended that awservices be active (you can issue the awservices start -m command).

install_mib-Load MIB Definitions

Valid on UNIX/Linux, Windows, z/OS

This shell script or batch job loads the MIBs on your system.

z/OS

Note: This shell script has been customized during the Agent Technology for z/OS installation.

If Agent Technology for z/OS is running, shut it down prior to running this script or batch job. To be sure the MIBs have been correctly loaded in the store, you should check the output file produced by the script run. It can be found in the install_mibs.out file, under the *install-path*/services/tools directory.

Windows

install-path\services\tools

UNIX/Linux

install-path/services/tools

z/OS JCL

&CAI.MFNSM.CAW0JCL(INSTMIBS)

This command has the following syntax:

```
install_mibs
```

ldconfig-Load Agent Configuration Files

Valid on UNIX/Linux, Windows, z/OS

ldconfig loads configuration data from an agent configuration file into sadmin Store. The common services must be running for this program to run successfully. This utility loads startup information contained in your agent configuration file into named sets of configuration data in the aws_sadmin store. Text messages indicate syntax errors found in configuration text file.

z/OS

If you run this as a batch job under z/OS, the configuration file for the example agent resides in &CAI.CAIPOTN(EXAGCFG).

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(LDCONFIG)

This command has the following format:

```
ldconfig [-r][-h host] fileName [-d level -f logfile] [mibName:configName]
[-d level -f logfile]
```

-r

Removes the configuration set associated with the MIB from sadmin Store. This action has no effect on the agent at the time of the removal because the agent runs with its own copy. When the agent is stopped and restarted again, the agent cannot load the removed configuration set.

-h *host*

Identifies the host name or IP address of the target node into which the configuration data is loaded.

Default: local node

fileName

Identifies the full configuration file name.

mibName:configName

Identifies the associated MIB and configuration file name.

-d *level*

Identifies the debug or logging level.

-f *logfile*

Identifies the name of the log file into which the debugging or logging information is sent.

ldmib - Load MIB Definitions

Valid on UNIX/Linux, Windows, z/OS

ldmib loads the MIB definition associated with an agent into sadmin Store. awservices must be running for this program to run successfully. The ldmib utility loads (or removes) a MIB whose definitions are encoded in SNMP concise MIB format as specified in RFC 1212, or in concise trap format as specified in RFC 1215.

Text messages indicate syntax errors found in the MIB definition file.

Windows

install-path/services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(LDMIB)

This command has the following format:

```
ldmib [-h hostName] -n mibName -m mibFile [-i|-r]
```

-h hostName

Specifies the host name or IP address of the node on which the common services are running.

Default: local node

-n mibName

Specifies the symbolic name of a node defined in the MIB definition file specified by *mibFile*. This symbolic name becomes the top OID of the MIB.

Note: This parameter is not required if the first line of the MIB contains the following line:

--MibName = mibname

This OID need not be the top-level node defined in the file. The OID can be any symbolic name that appears to the left of an ASN.1 expression like the following that is included in the MIB definition file:

mibName ::= OBJECT IDENTIFIER {xxxx n}

If the specified attribute is not the top-level node defined in the file, the resulting MIB loaded into sadmin Store includes only those branches of the tree below the specified node.

Note: This attribute is case sensitive and must be specified exactly as it appears in the text file.

-m mibFile

Specifies the name of the text file that holds the ASN.1-encoded definitions for the MIB attributes you want to load into sadmin Store. The MIB definitions in the file must be in RFC 1212 or RFC 1215 format. The file name must end with the .TXT extension.

-i

Generates an include file containing C# define statements for every object name in the specified MIB definition file. The comment lines in the include file also document the fully qualified OID of each MIB object.

The include file has the same name as the MIB definition file with the extension .h instead of .TXT.

-r

Removes the specified MIB definitions from sadmin Store.

Example

To load the MIB definition file for the Example Agent, exagent, enter the following command:

```
ldmib  -n exagent  -m "//'$AWORKS_MVS_PREFIX.MIBLIB(EXAGENT)' "
```

If the MIB definition is loaded successfully into sadmin Store, ldmib displays the following output:

```
ldmib: successful completion
```

mkconfig-Create Configuration Set from Agent Store

Valid on Windows, z/OS

mkconfig reads agent configuration information from the agent store and reformats this information into a configuration set that can be written to a file or to STDOUT.

Windows

install-path\services\bin

z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(MKCONFIG)

This command has the following format:

```
mkconfig {agentName[@host]| -s mibName[@host]} [-i instance] [name1...nameN]
[filename]
```

Example

To create a configuration set from the agent settings in agent store and put them into a file, enter the following command:

```
mkconfig -s caiNtOs foo.dat
```

agentName

Specifies the name of the agent whose settings you are going to use to create a configuration set

@host

Specifies the name of the host where the agent resides.

name1...nameN

Specifies the names of the groups or tables whose settings you are going to use to create a configuration set.

-s

Specifies that the settings should be taken from the agent store.

mibName

Specifies the name of the MIB where the settings you are going to use to create a configuration set are located.

filename

Specifies the name of the file where you will place the settings. If this is not specified, the settings are placed in STDOUT and written to the screen.

-i *instance*

Specifies database instance for mibmuxed database agents.

orbctrl-Distributed Services Bus Control Utility

Valid on UNIX/Linux, Windows, z/OS

orbctrl lists the services and agents that have attached to the instance of the Distributed Services Bus (aws_orb) running on the specified system.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(ORBCTRL)

This command has the following format:

```
orbctrl [-@ servicesHost]
```

-@ servicesHost

Identifies the system on which the services are running.

Default: local node

servicectrl-Component Dependencies Control Utility

Valid on UNIX/Linux, Windows, z/OS

servicectrl controls awservices configuration information by specifying the component name, path, and configuration options. servicectrl can also be used to control agents and services remotely. The awservices process must be active on the related host (local or remote).

Windows

servicectrl lets you define any Windows executable as a dependency for any Agent Technology component. For example, you can define a dependency such that OpenIngres must be running before the OpenIngres agent can start.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

This command has the following format:

```
servicectrl {install|installauto|remove|update|start|stop|status|version}
<coptions>
```

install

Installs the agent or service as a registered Agent Technology component.

installauto

Installs the agent or service and specifies that it must start automatically when awservices is started.

remove

Removes the agent or services as a registered Agent Technology component.

update

Removes the agent or service from the registry and then installs it.

start

Starts the agent or service in the background.

stop

Stops the agent or services.

status

Lists the operational status of the agent or service (that is, whether it is running or stopped).

version

Lists the name of the agent or service, its version information, and the name of its executable file.

Options

Note: These options have a double-dash(--) prefix.

--name=*component-name*

Specifies the name used to identify the component.

--instance=*instance-name*

Specifies the name for an instance of a component that has multiple instances with the same component name.

--remote=*remote-system*

Name of the host to which this command applies.

--image=*full-path-name-of-component*

Specifies the full path where the component is located.

--options=*default-options*

List of default options to be used when the component is started.

--type=*component-type*

Specifies the type of component. For Agent Technology components, the type is AWEXE. To install a component that is not a part of Agent Technology, the type is EXE. (Windows only)

--user=*user-name*

With password, specifies account under which the component is to be run.

--dependOn=*list-of-components*

Specifies components on which this component is dependent, and starts them when this component is started if they are not already running. Separate multiple component names with commas.

--autoStart=yes|no

Starts this component when awservices is started.

--andDependOn=yes|no

Stops a component even though components depending on it are still running. Stops the dependent components first.

install

Installs the agent or service as a registered Agent Technology component.

installauto

Installs the agent or service and specifies that it is to start up automatically when awservices is started.

remove

Removes the agent or service as a registered Agent Technology component.

update

Removes the agent or service from the registry and then installs it.

start

Starts the agent or service in the background.

stop

Stops the agent or service.

status

Lists the operational status of the agent or service (that is, whether it is running or stopped).

version

Lists the name of the agent or service, its version information, and the name of its executable file.

storectrl-Store Database Utility

Valid on UNIX/Linux, Windows, z/OS

storectrl provides a low-level command interface for the Agent Technology object stores. Use storectrl to display class definitions and object information stored in the Agent Technology stores:

Store	Managed by	Platform
aws_sadmin	aws_sadmin process	all platforms

Store	Managed by	Platform
AwNsm@<repository>	aws_dsm process	DSM on remote distributed system
objectStore	aws_store process	DSM on remote distributed system

You can also use storectrl to add properties, change property values, delete properties, and delete objects. These operations should not be attempted without the assistance of CA Technologies Support.

Windows

install-path\services\bin

UNIX/Linux, z/OS

install-path/services/bin

z/OS JCL

&CAI.MFNSM.CAW0JCL(STORECTL)

This command has the following format:

```
storectrl [ Store hostName micrObName ]
  Select propertyName=propertyValue [ propertyName=propertyValue ] End
        GetAllProperties
        GetProperties propertyName [ propertyName ] * End
        SetProperties propertyName=propertyValue [ propertyName=propertyValue ] *
End
        DeleteProperties propertyName [ propertyName ] * End
        Delete
```

[Store *hostName micrObName*]

Required for z/OS

hostName micrObName-Specify the hostname where the store resides and the particular store.

Required except when issued against local object store. See examples.

Default: objectStore (local)

Select *propertyName=propertyValue*
[*propertyName=propertyValue*] * End

Select one or more properties that match the specified values. Subsequent operations will be performed on the selected class or object.

GetAllProperties

Display all properties (can follow a Select).

SetProperties *propertyName=propertyValue*
[*propertyName=propertyValue*] * End

Change values for one or more existing properties (if property already exists) or create new properties and set values.

2-DeleteProperties *propertyName* [*propertyName*] * End

Delete selected class or object.

Example 1 (local z/OS system)

Display all attribute names for the MIB of loaded agent exagent stored in local aws_sadmin:

```
storectrl Store localhost aws_sadmin

        Select attrMib=exagent End
        GetProperties attrName End
```

Result:

Object : 0

attrName = exMonitorCnt

Object : 1

attrName = exHistStatus

Object : 2

attrName = exHistoryGroup

Object : 3

attrName = exFaultExcess

...

Note: The location of aws_sadmin must be specified even though it resides on the local system (because it is not the default, objectStore). "localhost" can be used to represent local host.

Example 2 (remote system with DSM)

Display property information on a DSM object, DemoAgt9:

```
storectrl Store USILCPCN AwNsm@USILCPCN Select moObjectName=DemoAgt9 End
GetAllProperties
```

Result:

Object : 0

__CLASS__ = AWMO_OBJECT

moObjectHost = USILCPCN

moObjectName = DemoAgt9

...

moObject_ccgVisible = 1

moObject_childPolicy = On

Example 3 (remote system with DSM)

Add a new property "pollInterval2" to object DemoAgt9:

```
storectrl Store USILCPCN AwNsm@USILCPCN Select moObjectName=DemoAgt9 End
SetProperties pollInterval2=300 End
```

Result, as shown by GetAllProperties:

Object : 0

__CLASS__ = AWMO_OBJECT

moObjectHost = USILCPCN

moObjectName = DemoAgt9

...

moObject_ccgVisible = 1

moObject_childPolicy = On

pollInterval2 = 300

Example 4 (remote system with DSM)

Delete the new property previously added to object DemoAgt9:

```
storectrl Store USILCPCN AwNsm@USILCPCN Select moObjectName=DemoAgt9 End
DeleteProperties pollInterval2 End
```

Result, as shown by GetAllProperties:

Object : 0

__CLASS__ = AWMO_OBJECT

moObjectHost = USILCPCN

moObjectName = DemoAgt9

...

moObject_ccgVisible = 1

moObject_childPolicy = On

Example 5 (local system with DSM)

Display class properties for DemoAgt9 (using local object store):

```
storectrl Select moClassName=DemoAgt9 End GetAllProperties
```

Result:

Object : 0

__CLASS__ = AWMO_CLASS

moClassName = DemoAgt9

...

moClass_initialState = Agent:UP

moClass_defaultState = Agent:UP

moClass_CriticalStr = Critical

Example 6 (local system with DSM)

Delete a class property of DemoAgt9 (using local object store):

```
storectrl Select moClassName=DemoAgt9 End DeleteProperties moClass_defaultState End
```

Result, as shown by GetAllProperties:

Object : 0

__CLASS__ = AWMO_CLASS

moClassName = DemoAgt9

...

moClass_initialState = Agent:UP

moClass_CriticalStr = Critical

Note: If after deleting a property (such as moClass_defaultState in this example) you later add the property back, it will appear at the end of the list when you display properties.

Chapter 3: Common Commands

This section contains the following topics:

ca_calendar-Start the Calendar Daemon

Applies to UNIX/Linux and Windows

Use the ca_calendar command to start the Enterprise Management calendar daemon. Normally the calendar daemon is executed as part of startup processing.

Enterprise Management functions such as Workload Management and Security Management use the calendar daemon to verify a date and time against a specified calendar. You can define calendars through the Enterprise Management command line or GUI interfaces. When started, the calendar daemon loads all calendars defined in the Enterprise Management database into the active calendar list. You can modify the active calendar list with the caladmin command.

This command has the following syntax:

```
ca_calendar [-f]
```

Startup Configuration

Execute the Enterprise Management Calendar daemon

To ensure that the Enterprise Management calendar daemon is executed during startup:

1. Verify that the Enterprise Management startup script, $CAIGLBL0000/scripts/rc, was properly integrated in your startup procedure.
2. Verify that the $CAIGLBL0000/scripts/rc file includes the following statement:
   `$CAIGLBL0000/cal/scripts/envset`

Performance Considerations

To minimize resource consumption, delete calendars from the active calendar list that are not in use.

The calendar daemon is automatically updated only when calendars are defined or altered through the GUI. You must add calendars defined or altered with the command line interface to the active calendar list using the caladmin command with the -a or -u option.

External Influences

CAIGLBL0000

CAICAL0000

CAICAL0001

CAICAL0002

CAICAL0003

Files

$CAIGLBL0000/cal/config/nodename/CA_CalDmon_Lock

Locking file that prevents more than one copy of the calendar daemon from executing concurrently. The locking file contains the process ID of the running calendar daemon. You can change it with the environment variable $CAICAL0003.

Warnings

If the calendar daemon encounters any environmental errors, it automatically shuts down. Other errors, such as missing or incomplete calendars, result in a calendar validation failure. The calendar validation failure causes all calendar analysis requests that name the affected calendar to return an invalid calendar status.

Check the SYSLOG file for any errors the calendar daemon encountered.

Examples

Start the calendar daemon if it is not already started:

```
ca_calendar
```

Shut down the currently executing calendar daemon to force a new calendar daemon to start, if necessary.

```
ca_calendar -f
```

caladmin-Configure or Shut Down the Calendar Service

Applies to UNIX/Linux and Windows

Use the caladmin command to modify the Enterprise Management calendar service. Use this command to add, update, and delete calendars from the active calendar list kept in storage by the calendar service. Also, use this command to shut down the calendar service.

Performance Considerations

To minimize resource consumption, calendars that are not in use should be deleted from the active calendar list.

Calendars created or modified with the command line interface are not reflected in the calendar service unless the caladmin command is used to modify the active calendar list or the calendar service is terminated and restarted.

Use the -a, -d, and -u options to modify the active calendar list for specific calendar entries.

caladmin (Windows)

This command has the following format:

```
caladmin  [-r]
                [-b]
                [-s]
                [-l]
                [-dcalendar]
                [-acalendar]
                [-ucalendar]
                [-v]
```

Environment Variables

%CAIGLBL0000%

Identifies the path where Enterprise Management is installed.

%CAICAL0000%

Contains information for debugging and tracing the calendar service. (For CA internal use only.)

caladmin (UNIX/Linux)

Note: This command must be executed on a UNIX/Linux system.

This command has the following format:

```
caladmin  [-r]
          [-b]
          [-s]
          [-l]
          [-dcalendar]
          [-acalendar]
          [-ucalendar]
          [-v]
```

-r

Refresh the active calendar list. The calendar service releases the current active calendar list and reloads all calendars defined in the Enterprise Management database.

-b

Rebuild binary files and refresh the active calendar list. The calendar service releases the active calendar list and rebuilds the binary files for all calendars defined in the Enterprise Management database. The rebuilt calendars are loaded into the active calendar list.

Important! *The* following *information applies to the -r and -b options.*

When instructed to refresh the active calendar list, the calendar service queues all calendar validation requests until the refresh process is complete.

Note: The refresh process can be overhead intensive and can slow down system performance. Therefore, we recommend that you use the -r or-b option judiciously.

-s

Shut down the calendar service.

Important! Shutting down the calendar service causes all requests for calendar validation to be returned as an “invalid day/time” condition. This has the potential to cause all logons, workload schedules and jobs, and so forth, to be rejected. Therefore, the -s option **should not** be used during normal production hours. The -s option should be issued just before a reboot or system shutdown.

-l

List the name of each calendar in the active calendar list. Check the system console or Windows Event Viewer file to view the list.

-dcalendar

Delete (remove) the named calendar from the active calendar list.

Note: Any subsequent requests for the deleted calendar result in calendar validation failure returning an “invalid day/time condition."

-acalendar

Add the named calendar (defined in the Enterprise Management database) to the active calendar list.

-ucalendar

Update the named calendar in the active calendar list. The calendar service reads the binary file for the named calendar and updates the copy of the calendar in the active calendar list.

-v

Save current in-core calendars' DSB file.

Environment Variables

CAIGLBL0000

Identifies the path where Enterprise Management is installed.

Examples

```
caladmin -r
```

Refreshes the active calendar list.

```
caladmin -b
```

Rebuilds binary files and refreshes the active calendar list

```
caladmin -s
```

Shuts down the calendar service.

```
caladmin -l
```

Produces a list of all calendars in the active calendar list.

```
caladmin -dxyzcal
```

Removes the calendar xyzcal from the active calendar list.

```
caladmin -aabccal
```

Adds the calendar abccal, defined in the Enterprise Management database, to the active calendar list.

```
caladmin -uupdcal
```

Updates the copy of the calendar updcal in the active calendar list.

staradmin-Administer stardaemon Process

Applies to UNIX/Linux only

On the UNIX system, the staradmin command is used to administer a running stardaemon process.

Note: You must have root status to use this program.

This command has the following format:

```
staradmin [-c]
          [-k]
          [-l]
          [-q]
          [-r[filename]
          [-s]
```

-c

Cleans up the structures, processes, and so on that are being used. Does not exit the program.

-k

Kills the daemon and all child processes. Exits the program.

-l

Provides a full status from the daemon. Lists all information provided by the -s option, plus the names and nodes of users logged into the stardaemon, and which outstanding processes are running.

-q

Stops the daemon, but lets all child processes finish. Exits the program.

-r *(filename)*

Refreshes the configuration from the optional file supplied (filename) or from the default location, if the file is not specified. Returns a status or error message from the refresh operation.

-s

Provides a brief status from the daemon. All output is sent to stderr. Lists the time the daemon started, the configuration file used and when it was last refreshed, the number of users logged into the stardaemon, and other information.

Environment Variables

CAIGLBL0000

Must be set in order for staradmin to run properly.

stardaemon-Run Enterprise Management stardaemon on UNIX

Applies to UNIX/Linux and z/OS only

The main executable for Enterprise Management for UNIX servers is stardaemon. On your UNIX system, it runs a background process under the root ID. On z/OS the stardaemon process runs under Unix Systems Services.

You may want to customize the behavior of the stardaemon by modifying the star.config file in the $CAIGLBL0000/star/etc directory.

This command has the following format:

```
unistart star
```

Activate CAICCI

Before stardaemon can function with STAR clients on other systems, the CAICCI Server daemon and CAICCI Remote Server daemon (ccirmtd) must be running. Make sure they are running prior to starting the stardaemon.

On z/OS, CAICCI normally starts with a system IPL.

On distributed machines and on z/OS, use this command to check whether the CAICCI Remote Server has been activated:

```
unifstat
```

In the table that appears, you should see entries for the CAICCI Server and the CAICCI Remote Server, as shown below.

Enterprise Management for UNIX Daemon Status Report

Daemon Name	Pid	Status
CA-Calendar Server	1178	running
CA-CCI Server	939	running
CA-CCI Remote Server	946	running

.
.
.

Start CAICCI

If the CAICCI daemons are not running on the Unix, Linux or Windows remote system, you should stop and restart Enterprise Management on those remote systems.

To stop Enterprise Management

Issue the following command:

```
$CAIGLBL0000/scripts/unishutdown all
```

Once executed, wait for the "Enterprise Management Shutdown Complete" message before attempting to restart.

To restart Enterprise Management

Issue the following command:

```
$CAIGLBL0000/scripts/unistart all
```

Environment Variables

CAIGLBL0000

Must be set for stardaemon to run properly.

TMPDIR

Specifies the location of temporary files.

CAISTARDEBUG

If set, causes stardaemon to execute in foreground, and debug messages are displayed from both the stardaemon and programs that it invokes. It is not recommended that the stardaemon be run with CAISTARDEBUG set except when debugging a problem.

star.config File Settings

The stardaemon reads the star.config file, then begins processing requests from clients. The following variables, if set in the star.config file, impact the way the stardaemon acts:

LOGIN_REQUIRED

If set to 1, then users must log in to the stardaemon through utilities provided on the client machine before accessing functions on the Enterprise Management STAR server.

Note: If LOGON_TIMEOUT is also specified, then it is possible for a user login to the stardaemon to expire, requiring the user to log in again to the stardaemon, even though they were successfully able to access functions moments before. At stardaemon termination, the current logins are saved to the star.save file. These logins are reloaded when the server is restarted.

TRANSACTION_TIMEOUT

Indicates the maximum time a process that was started by the stardaemon is allowed to run.

Requests from clients usually cause additional programs to be executed asynchronously from the stardaemon. These programs are defined in the starobj.map file. By setting a TRANSACTION_TIMEOUT value, you can prevent processes from "running away" or hanging indefinitely. A value of 0 (zero) means there is no time out.

unicntrl-Start Enterprise Management Functions

Applies to UNIX/Linux, Windows, z/OS

Use the unicntrl command to start Enterprise Management, an Enterprise Management function, or a common object. The command provides a convenient way to start one or more Enterprise Management functions that have been shut down without restarting Enterprise Management.

This command has the following format:

```
unicntrl command [keywords]
```

Applies to Windows only

```
[SERVER nodename]
[TRACE]
```

Command

Valid commands are:

STOP

Stops the specified CA NSM component on the node named in SERVER.

START

Starts the specified CA NSM component on the node named in SERVER. This is the default.

PAUSE Applies to Windows only

Pauses the specified CA NSM component on the node named in SERVER.

RESUME Applies to Windows only

Resumes running the specified CA NSM component on the node named in SERVER.

Default: START

Keywords

Identifies one or more of the following Enterprise Management functions or common objects. Use a space between multiple keywords.

Note: You might not have all components installed. Options for components not installed are ignored.

asm Applies to UNIX/Linux and Windows only -

File Management function

com Applies to UNIX/Linux and Windows only -

Common CA NSM components

opr Event Management function

prb **Applies to UNIX/Linux and Windows only** -

Problem Management function

rpt **Applies to UNIX/Linux and Windows only** -

Report Management function

sch **Applies to UNIX/Linux and Windows only** -

Workload Management function

sec **Applies to UNIX/Linux and Windows only** -

Security Management function

uni **Applies to Windows only -**

The default configuration found in the CAICFG.GLO file

all **Applies to UNIX/Linux and Windows only** -

The default configuration found in the CAICFG.GLO file

Default: uni | all

SERVER nodename

Applies to Windows only

Specifies the name of a remote CA NSM (Windows only) node you wish to control.

TRACE

Applies to Windows only

Starts a trace of the specified CA NSM component on the remote machine.

Files

%CAIGLBL0000%\bin\uncntrl.exe for windows

$CAIGLBL0000/bin/unicntrl for z/OS

Example

```
unicntrl start opr
```

Starts the Event Management function.

unicycle-Recycle Enterprise Management and Related Services

Applies to UNIX/Linux and z/OS

The unicycle command shuts down and restarts one or more Enterprise Management functions. unicycle is equivalent to performing a unishutdown followed by a unistart.

Note: Before you shut down or recycle any processes, you should consider that all Enterprise Management functions rely on the Event Manager, License Manager, and CAICCI processes. If any of these processes are stopped or cycled, you should shut down all of Enterprise Management to avoid system outages.

This command has the following format:

`unicycle {`*`name`* `[`*`name...`*`]|ALL}`

Name

Identifies one of the following Enterprise Management functions or common objects.

Note: You might not have all components installed. Options for components not installed are ignored.

cal Calendar

crit **Applies to UNIX/Linux and Windows only -**

Criteria Profile Manager

db **Applies to UNIX/Linux and Windows only -**

CAIUNIDB

enf **Applies to UNIX/Linux and Windows only -**

CAIENF server

gui **Applies to UNIX/Linux and Windows only -**

GUI main message router (CaMRouter)

help GUI help database server

opr Event Management function

prb **Applies to UNIX/Linux and Windows only -**

Problem Management function

rept **Applies to UNIX/Linux and Windows only -**

Report Management function

sche **Applies to UNIX/Linux and Windows only -**

Workload Management function

secu **Applies to UNIX/Linux and Windows only -**

Security Management function

tape **Applies to UNIX/Linux and Windows only -**

Tape Management function

snmp SNMP

ALL **Applies to UNIX/Linux and Windows only -**

All Enterprise Management functions, common objects, and services.

Environment Variables

CAIGLBL0000

Identifies the path where Enterprise Management is installed.

Files

$CAIGLBL0000/bin/unicycle

The unicycle executable.

Example

```
unicycle cal
```

Recycles only the Calendar function. Informational messages directed to stdout depict the progress.

Note: Issue this command from the root login ID.

unistart - Start Enterprise Management Functions

Applies to UNIX/Linux, and z/OS

The unistart command starts Enterprise Management, an Enterprise Management function, or a common object. The command provides a convenient way to start one or more Enterprise Management functions that were shut down without restarting Enterprise Management.

This command has the following format:

```
unistart {name [name...]|ALL}
```

Note: Issue this command from the root login ID.

Name

Identifies one of the following Enterprise Management functions or common objects. If you are using unistart to start components one at a time, ensure that you start the prerequisite components before using unistart.

cal Calendar

cci **Applies to UNIX/Linux and Windows only -**

Common Communications Interface (CAICCI).

crit **Applies to UNIX/Linux and Windows only -**

Criteria Profile Manager.

db **Applies to UNIX/Linux and Windows only -**

CAIUNIDB.

emsrvc Enterprise Management Services

enf **Applies to UNIX/Linux and Windows only -**

CAIENF server.

gui **Applies to UNIX/Linux and Windows only -**

GUI main message router (CaMRouter).

help GUI help database server.

opr Event Management function.

pdm **Applies to UNIX/Linux and Windows only -**

Service Desk.

prb **Applies to UNIX/Linux and Windows only -**

Problem Management function.

reptApplies to UNIX/Linux and Windows only -

Report Management function.

sche **Applies to UNIX/Linux and Windows only -**

Workload Management function.

secu **Applies to UNIX/Linux and Windows only -**

Security Management function.

snmp SNMP

star Remote logon function.

tape **Applies to UNIX/Linux and Windows only -**

Tape Management function.

ALL **Applies to UNIX/Linux and Windows only -**

All Enterprise Management functions, common objects, and services.

Files

$CAIGLBL0000/scripts/unistart

Script file for unistart command.

Example

```
unistart opr
```

Starts the Event Management function.

unifstat-Check Status of Enterprise Management

Applies to UNIX/Linux, Windows, z/OS

Use the unifstat command to display the status of the Enterprise Management functions, common objects, and services currently configured on the specified machine.

unifstat (UNIX/Linux)

This command has the following format:

```
unifstat
```

This command has the following format:

```
unifstat [-h]
         [-r]
         [-p]
         [-v]
         [-c component]
         [-u subcomponent]
         [-s node]
```

Note: If UNIFSTAT is issued without any options, the assumed default is UNIFSTAT -C * -U *. This displays the status of all components and subcomponents on the current machine.

-h

Displays syntactical help information for the unifstat command.

-r

Displays requisite information for the selected components/subcomponents. Both prerequisite and corequisite information is displayed. A component that is a prerequisite must reach an ACTIVE state before the dependent component can be started or resumed. A component that is a corequisite must maintain an ACTIVE state before the dependent component can be paused.

Note: This option should be specified only under the direction of CA Technologies Support.

-p

Displays process information associated with the selected components/subcomponents. An Active Process list and a Critical Process list are displayed (as appropriate) for each component/subcomponent.

Within each list the process ID and thread ID of the processes for the component are listed. Processes listed in the Critical Process list are essential to the operation of the component. If a Critical Process should die, the component would stop.

-v

Provides additional details about the information requested.

Note: This option should only be specified under the direction of CA Technologies Support.

-c *component*

Identifies the Enterprise Management components for which status information is desired. A trailing asterisk (*) may be used to select a group of components. Specify -c * to display all of the currently registered components for a machine. This list will show you the possible values, which may be specified for *component.*

Every registered component moves through several states during the course of Enterprise Management operation. There are three groups of states: terminal, transitory, and wait:

Terminal States

Terminal states are reached using the unicntrl command or through the control panel. The possible terminal states are:

ACTIVE

The component is active

INACTIVE

The component is inactive

PAUSED

The component is paused

Transitory States

Transitory states occur on the way to a terminal state. It is NOT normal for these states to persist for more than a minute or two. The possible transitory states are:

START_PENDING

The component is being started

PAUSE_PENDING

The component is being paused

RESUME_PENDING

The component has been resumed after a pause

TERM_PENDING

The component has received a STOP order

Wait States

Wait states may persist for a while, but are not indicative of a fully active system. The possible wait states are:

START_PENDING_REQUISITE

The START order for the component has been delayed because the component is a dependent and the requisite component is not at the necessary state.

RESUME_PENDING_REQUISITE

The RESUME order for the component has been delayed because the component is a dependent and the requisite component is not at the necessary state.

-u *subcomponent*

Identifies the subcomponents (of Enterprise Management components) for which status information is desired. A trailing asterisk (*) may be used to select a group of subcomponents. If the -u option is not specified, the status of all the subcomponents of the selected Enterprise Management components will be generated. For a list of all possible subcomponents, specify only the -c * option.

-s *node*

Identifies the machine for which you are requesting Enterprise Management status. If the -s option is not specified, the machine on which the unifstat command is issued is assumed.

Options

None.

Files

$CAIGLBL0000/bin/unifstat

Script file for unifstat command.

$CAIGLBL0000/scripts/unifstat.awk

The awk program source code.

unishutdown-Shut Down Enterprise Management and Related Services

Applies to UNIX/Linux and z/OS

The unishutdown command systematically shuts down all Enterprise Management functions. Any Enterprise Management work in progress is typically completed before the responsible daemon is terminated.

You can use this command to shut down one or more Enterprise Management functions without terminating all Enterprise Management processing.

This command has the following format:

```
unishutdown {name [name...]|ALL}
```

Note: Issue this command from the root login ID.

Name

Identifies one of the following Enterprise Management functions or common objects.

cal Calendar

cci **Applies to UNIX/Linux and Windows only -**

Common Communications Interface (CAICCI)

crit **Applies to UNIX/Linux and Windows only -**

Criteria Profile Manager

db **Applies to UNIX/Linux and Windows only -**

CAIUNIDB

emsrvc Enterprise Mgmt Services

enf **Applies to UNIX/Linux and Windows only -**

CAIENF server

gui **Applies to UNIX/Linux and Windows only -**

GUI main message router (CaMRouter)

help GUI help database server

opr Event Management function

pdm **Applies to UNIX/Linux and Windows only -**

Service Desk

prb **Applies to UNIX/Linux and Windows only -**

Problem Management function

reptApplies to UNIX/Linux and Windows only -

Report Management function

sche **Applies to UNIX/Linux and Windows only -**

Workload Management function

secu **Applies to UNIX/Linux and Windows only -**

Security Management function

tape **Applies to UNIX/Linux and Windows only -**

Tape Management function

snmp SNMP

ALL **Applies to UNIX/Linux and Windows only -**

All Enterprise Management functions, common objects, and services.

Environment Variables

CAIGLBL0000

Identifies the path where Enterprise Management is installed.

Files

$CAIGLBL0000/scripts/unishutdown

Shell script source for the unishutdown command.

Examples

```
unishutdown opr
```

Shuts down only the Event Management function. Informational messages directed to stdout depict the progress of the shutdown.

univer-Check the Version of Enterprise Management

Applies to UNIX/Linux, Windows and z/OS

This command returns the product and version you have installed.

Chapter 4: CAICCI Control Options

CAICCI control options can be specified in the CAIENF parameter file after the CAIENF parameters or in their own parameter file concatenated after the CAIENF parameter file referenced by the ENFPARMS DD. After startup, most CAICCI control options may be dynamically altered using the CAICCI or CAIENF operator command from any z/OS system console or operator command facility.

CAICCI generates informative messages in response to each control option. These messages and their variables are defined in the *Message Reference Guide.*

Control Options Summary - CAICCI

The following table has the CAICCI control options, operands, default values listed in alphabetic order.

Control Option	Operands	Default Values
CCI	(RECYCLE)	n/a (not applicable)
CCI	(STATUS,[sysid])	n/a
CCI	(TERM)	n/a
CCICT	(component_name,#buffers,buffer_size,member_name)	n/a
CONNECT	(sysid,...)	n/a
DISPLAY,CONNECT	n/a	n/a
DISPLAY,LINK	n/a	n/a
DISPLAY,NODE	n/a	n/a
DISPLAY,PROTOCOL	n/a	n/a
DISPLAY,RECEIVER	n/a	n/a
DISPLAY,RESOURCE	n/a	n/a
DISPLAY,SYSID	n/a	n/a
GATEWAY	(protocol,netparm,retry,sysid, maxru,start/stop,netparm2)	retry=10, maxru=4096, start/stop=START/SHUT
GENERIC	(protocol,generic-name)	n/a

Control Option	Operands	Default Values
HOSTNAME	(hostname[,FORCE])	No hostname defaults to the VMCF subsystem nodename.
LOGGER	(string,buffer1,buffer2,Y|N)	string=20, buffer1=12, buffer2=12, N
MAXRU	(nnnnn)	n/a
NODE	(protocol,netparm,retry,sysid, maxru,start/stop,netparm2)	retry=10, maxru=4096, start/stop=START/SHUT
NODEDATA	([DISPLAY|RESET,]protocol, sysid,netparm)	n/a
PERMIT	(NONE|ALL)	ALL
PROTOCOL	(protocol,netparm,retry,sysid,maxru,start/stop)	retry=10, maxru=4096, start/stop=START/SHUT
PROTSEC	(XCF|XES|XES,XCF|XCF,XES)	n/a
REMOVE	(SYSID,xxxxxxxx)	n/a
SYSID	(sysid)	n/a
SYSPLEX	(sysplexid)	n/a
VARY	(INACT|ACT,SYSID,sysid)	n/a

Notes:

- In the control option descriptions found in this guide, MAXRU is used as a control option and an operand for the GATEWAY, PROTOCOL and NODE control options. MAXRU is the maximum data packet size, specified in bytes, that is allowed to be transmitted between two CAICCI subsystems. You can redefine the maximum RU default value of 4096 using this control option.

  To determine the required MAXRU for hosts that are communicating using VTAM and are connected using NCP, see the MAXDATA operand of the PCCU macro, the BUFSIZE of the BUILD macro, and the INBFRS, MAXBFRU, and UNITSZ operands of the HOST macro.

  To determine the required MAXRU for hosts that are connected using CTC, see your VTAM start option LPBUF.

  Refer to the IBM reference manuals and consult with your VTAM/NCP Systems Programmer for more information.

- Use the CCI(RECYCLE) and CCI(TERM) control options only under the direction of CA Support.
- The CONNECT control option refers to definitions made using the GATEWAY and NODE control options to perform actual CAICCI linking.

- The GATEWAY and NODE control options establish CAICCI control blocks.
- The keyword SHUT is equivalent to the keyword STOP. Either may be coded to indicate that session termination occurs at CAICCI shutdown.
- With IPv6 addresses, you must use coded delimiters to identify the start and end of the address separate from the port number.

  Any of the following character pairs can be used as delimiters:

  ```
  start "["   end "]"
  start "<"   end ">"
  start and end "/"
  start and end "\"
  start and end "|"
  ```

  In the following example, coded brackets "[" and "]" delimit the start and end of the IPv6 address.

  ```
  [fd00:7a06:a20:100::11]:7001
  ```

CCI(RECYCLE)

Causes CAICCI to terminate all sessions with its remote CAICCI nodes and LU2 devices (PCs), followed by a shutdown of the CAICCI subsystem under CAIENF. Immediately afterwards, the CAICCI subsystem attempts to restart and re-establish VTAM connections with its remote subsystems that have NODE and CONNECT control statements defined for them. Remote CAICCI subsystems lacking NODE and CONNECT definitions and all PCs must re-establish the session with the local CAICCI from the remote CAICCI subsystems.

Important! This control option should be used only under the direction of CA Technologies Support.

Example (console)

```
CCI CCI(RECYCLE)
```

CCI(STATUS,[sysid])

Enter this control option to display the host CAICCI link status and resources. When this control option is invoked, CAICCI generates informative messages on the status of CAICCI host and link(s). Specifying "sysid" causes CAICCI to return the information for this specific system only.

Example (console)

```
CCI CCI(STATUS)
CCI CCI(STATUS,A97S)
```

Example (ENFPARMS)

```
CCI(STATUS)
CCI(STATUS,A97S)
```

CCI(TERM)

Causes CAICCI to terminate all sessions with its remote CAICCI nodes and LU2 devices (PCs), followed by a shutdown of the CAICCI subsystem under CAIENF. CAICCI subsystem will not attempt a restart.

Note: There is no way to restart the CAICCI subsystem under the current CAIENF started task. Once this option is issued, CAIENF must be stopped and restarted to reactivate the CAICCI subsystem.

Important! This control option should be used only under the direction of CA Technologies Support.

Example (console)

```
CCI CCI(TERM)
```

CCICT (component_name,#buffers,buffer_size,member_name)

The CCICT statement controls the Component Trace environment. For more information, see CAICCI Component Trace (see page 165).

Options are:

component_name

The Component Trace name. Default is CACCI

#buffers

Number of trace buffers. Specify a number between 2 and 128 inclusive. Default is 3.

buffer_size

The size, in K, of each buffer. Specify a number between 64 and 8192 inclusive (buffer sizes will be between 64k and 8M). Default is 512.

member_name

Name of the parmlib member for Component Trace to process when the Component Trace is initialized. There is no default value but a sample member CTECCI00 is provided for tailoring.

If a CCICT statement is not included in CCIPARMS, CAICCI is still defined to COMPONENT TRACE with the default values for the component_name, #buffers and buffer_size. All other trace options and values are set through the TRACE CT command when activating tracing.

CONNECT(sysid,...)

Use this control option to connect one or more remote CAICCI nodes (previously defined by a GATEWAY or NODE control option) to the local host.

This control option has the following operand:

sysid

The unique 1 to 8 character identifier of the remote CAICCI system to be connected. A maximum of seven CAICCI sysids may be connected using a single CONNECT statement.

This operand is **required**.

Example (console)

```
CCI CONNECT(A73CVC01,A93CVC01)
```

Example (ENFPARMS)

```
CONNECT(A73CVC01,A93CVC01)
```

DISPLAY,CONNECT

The DISPLAY,CONNECT option displays the CAICCI connections defined to this host.

Example (console)

```
CCI DISPLAY,CONNECT
```

Example (ENFPARMS)

```
DISPLAY,CONNECT
```

DISPLAY,LINK

The DISPLAY,LINK option displays the host CAICCI link status.

Example (console)

```
CCI DISPLAY,LINK
```

Example (ENFPARMS)

```
DISPLAY,LINK
```

DISPLAY,NODE

The DISPLAY,NODE option displays the CAICCI NODEs defined to this host

Example (console)

```
CCI DISPLAY,NODE
```

Example (ENFPARMS)

```
DISPLAY,NODE
```

DISPLAY,PROTOCOL

The DISPLAY,PROTOCOL option displays the CAICCI PROTOCOLs defined to this host.

Example (console)

```
CCI DISPLAY,PROTOCOL
```

Example (ENFPARMS)

```
DISPLAY,PROTOCOL
```

DISPLAY,RECEIVER

DISPLAY,RECEIVER displays all local CAICCI receivers active on the current host.

Example (console)

```
CCI DISPLAY,RECEIVER
```

Example (ENFPARMS)

```
DISPLAY,RECEIVER
```

DISPLAY,RESOURCE

DISPLAY,RESOURCE displays all CAICCI resources.

Example (console)

```
CCI DISPLAY,RESOURCE
```

Example (ENFPARMS)

```
DISPLAY,RESOURCE
```

DISPLAY,SYSID

DISPLAY,SYSID displays the CAICCI SYSID of the host.

Example (console)

```
CCI DISPLAY,SYSID
```

Example (ENFPARMS)

```
DISPLAY,SYSID
```

GATEWAY(protocol,netparm,retry,sysid,maxru,start/stop,netparm2)

The GATEWAY control option is used to define a remote CAICCI node to the local host. The default values are: retry=10, maxru=4096, start/stop=START/SHUT.

By specifying a GATEWAY rather than a NODE control option for a particular remote host, the remote host is subsequently passed all resource information from all other remote hosts that are connected to the local host. Likewise, all resource information received from the remote host is forwarded to all other remote hosts. This allows the remote host to communicate through the local host with all its other remote hosts without having a direct connection with them.

The following parameters must be entered with the GATEWAY control option:

- protocol
- netparm
- retry
- sysid

protocol

Specifies the desired communication protocol to be used. Possible values are:

LU0

LU0 is the communications protocol supported by the local host node to access the remote CAICCI node.

TCPIPGW

TCPIPGW is the communications protocol supported by the local host node to access the remote host node using IBM TCP/IP or CA TCPaccess Communications Server for z/OS.

TCPSSLGW

TCPSSLGW is the communications protocol supported by the local host node to access the remote host node using IBM TCP/IP or CA TCPaccess Communications Server for z/OS with Secure Sockets layer (SSL) support.

netparm[:port][;keyword=value...]

- For LU0 (or VTAM), netparm specifies the name defined in VTAMLST (as a cross-domain resource) that the remote CAICCI uses as its APPLID.
- For TCPIPGW or TCPSSLGW, netparm specifies the TCP/IP Hostname or address of the remote CAICCI host. This is a positional operand that requires two commas as a place holder if it is not specified. When not specified, the IP information is obtained from the remote CAICCI specified by the sysid operand.

  The IP address may be specified in IPv6 format. For information on IPv6 address identification, see the notes in CAICCI Control Option Summary (see page 131).

  :port

  The port operand specifies the port number on which the gateway will connect to the remote host. It is only valid when specifying a protocol of either TCPIPGW or TCPSSLGW. The port number needs to be specified if the listening port of the remote host is different than the port number specified with the PROTOCOL control option. If the port number is specified, it must be appended to the hostname or IP address prefixed with a colon(:).

  ;keyword=value

  The keyword operand specifies SSL Keywords and their values. It is only valid when specifying a protocol of either TCPIPGW or TCPSSLGW. Each keyword=value must be prefixed with a semi-colon (;).

  The following is a list of valid keywords, their abbreviations, and allowable values (case insensitive). Each keyword=value parameter must be prefixed with a semicolon (;) as a delimeter.

  - SECURE | SEC | S =

    Indicates whether or not to request an SSL (secured) connection (default is set from the PROTOCOL UNSECURED_CONNECT setting):

    YES | Y - Connection must be secure. If the remote system does not support or is not enabled for SSL, the connection request fails.

    ALLOW | A – An unsecured connection is allowed and requested. If the remote CAICCI **requires** an SSL connection, then a secure connection is permitted and made.

    DEFER | D – Connection type defers to the remote system: If the remote CAICCI supports and is enabled for SSL, then it connects securely; otherwise the connection is made unsecured.

    NO | N - Only unsecured connections allowed. If the remote system **requires** an SSL connection, the connection request fails.

retry

Specifies the re-poll time in minutes that CAICCI uses to attempt to re-establish a session with the remote CAICCI using the netparm specified. This time ranges from 1 to 59 minutes. This operand has a default value of 10. This is a positional operand that requires two commas as a place holder when the default value is to be used.

sysid

Specifies the unique 1 to 8 character identifier of the remote CAICCI system.

maxru

Specifies the maximum data packet size, specified in decimal bytes, that is allowed to be transmitted between the local CAICCI and this remote CAICCI node.

The default MAXRU value is 4096 or the value set by the MAXRU control option.

start/stop

Specifies when the session with the remote CAICCI is to be established and terminated. Acceptable values are:

- START/SHUT - Start link at CAICCI startup time. Drop link when CAICCI shuts down. START/SHUT is the default.
- START/TERM - Start link at CAICCI startup time. Drop link when the first CA application issues a CCI TERM.
- INIT/SHUT - Start link when first CA application issues a CCI INIT. Drop link when CAICCI shuts down.
- INIT/TERM - Start link when the first CA application issues a CCI INIT. Drop link when the first CA application issues a CCI TERM.

netparm2

For VTAM and LU0, specifies the logmode name for assigning a class of service (COS) to the session with the remote CAICCI node.

Important! The operands STOP and SHUT are fully compatible and can be substituted for each other.

Examples (console)

```
CCI GATEWAY(LU0,A73CVC01,01,A73SYSID,4000,START/SHUT)

CCI GATEWAY(TCPIPGW,IPGW73:1721,01,A73SYSID)

CCI GATEWAY(TCPSSLGW,,01,A93S)

CCI GATEWAY(TCPSSLGW,USILCA31;S=NO,1,A31ENF)
```

Examples (ENFPARMS)

```
GATEWAY(LU0,A73CVC01,10,A73SYSID,4000,START/SHUT)

GATEWAY(LU0,A73CVC01,,A73SYSID,4000,START/SHUT)
```

GENERIC(protocol,generic-name)

The GENERIC control option is used to define a generic network name to the local host for a given protocol. Providing a generic-name for a protocol is not required.

Parameters that must be entered for GENERIC are:

protocol

Specifies the communication protocol whose specific network connection is being assigned as a generic network resource. Currently, VTAM is the only communication protocol that supports generic resources, and only within a sysplex environment.

generic-name

Specifies the generic-name to be assigned to the specific network resource as defined in the PROTOCOL statement. If this parameter is omitted, the generic-name previously assigned, if any, is deleted.

Example (console)

```
CCI GENERIC(VTAM,A99CVC01)
```

Example (ENFPARMS)

```
GENERIC(VTAM,A99CVC01)
```

HOSTNAME(hostname[,FORCE])

Event Management commands require that the DNS host name of the target system be specified rather than its CAICCI SYSID. By default, CAICCI uses the nodename specified with the VMCF subsystem definition in SYS1.PARMLIB member IEFSSNxx as its default TCP/IP DNS host name. If the VMCF nodename is not a valid host name, the HOSTNAME control option can be used to assign the correct host name.

This control option is optional. If this control option is entered as a command when CAICCI is active, you receive the following message:

```
CAS9604W - CAICCI - Can not alter HOSTNAME when CCI Active
```

This control option has the following operands:

hostname

The 1 to 24 character hostname that is to be assigned as the default TCP/IP DNS host name for the local CAICCI system.

If this parameter is omitted, the VMCF subsystem nodename is assigned as the default TCP/IP DNS host name for the local CAICCI system.

FORCE

The FORCE operand allows the HOSTNAME control option to be entered as a command when CAICCI is active. Changing the host name does not affect CAICCI's internal operation. However, a local or remote CAICCI application may have obtained the previous hostname. Assigning a new host name midstream may cause this application to generate invalid output or operate incorrectly. A recycle of the application or CAICCI may be necessary.

Example (ENFPARMS)

```
HOSTNAME(USI297ME)
```

LOGGER(strings,buffer1,buffer2,reorg)

The LOGGER control option is used to activate the Assured Delivery feature of CAICCI and specify the number of VSAM strings and buffers that will be saved in the LOGGER database.

Parameters that must be entered for LOGGER are as follows:

strings

The number of VSAM strings. Calculate this number as follows: 2 x #systems + 5

The default is 20 strings.

buffer1

The number of VSAM data buffers. Calculate this number as follows: 2 x #systems + 3

The default is 12 data buffers.

buffer2

The number of VSAM index buffers. Calculate this number as follows: #systems but not < 12

The default is 12 index buffers.

reorg

Y or N. If Y is specified, LOGGER will reorganize the VSAM database at start-up of CCILGR.

The default is N.

Example (console)

```
CCI LOGGER(25,23,12,Y)
```

Example (ENFPARMS)

```
LOGGER(25,23,12,Y)
```

MAXRU(nnnnn)

The MAXRU control option alters the default maximum RU size from its initial value of 4096. The default value is used whenever the MAXRU operand is not specified on the GATEWAY, PROTOCOL or NODE control options.

nnnnn

Specifies the maximum data packet size, in decimal bytes, that is allowed to be transmitted between two or more CAICCI network nodes.

Important! If MAXRU is specified, it must be placed after the SYSID control option statement and prior to any GATEWAY, PROTOCOL and NODE control option statements.

Note: It is recommended that the default MAXRU value be set to the smallest path size in your network. Typically, the initial value of 4096 is sufficient unless one or more hosts are connected using an NCP. A default value of 256 is recommended in this network.

Example (ENFPARMS)

```
MAXRU(4096)
```

This control option may **not** be entered using the console.

NODE(protocol,netparm,retry,sysid,maxru,start/stop,netparm2)

The NODE control option is used to define a remote CAICCI node to the local host. The default values are: retry=10, maxru=4096, start/stop=START/SHUT.

The following parameters must be entered:

- protocol
- netparm
- retry
- sysid

Separate multiple options with commas.

protocol

Specifies the desired communication protocol to be used. Possible values are:

LU0

LU0 is the communications protocol supported by the local host node to access the remote CAICCI node.

TCPIPGW

TCPIPGW is the communications protocol supported by the local host node to access the remote host node using IBM TCP/IP or CA TCPaccess Communications Server for z/OS.

TCPSSLGW

TCPSSLGW is the communications protocol supported by the local host node to access the remote host node using IBM TCP/IP or CA TCPaccess Communications Server for z/OS with Secure Sockets layer (SSL) support.

netparm[:port][;keyword=value...]

- For LU0 (or VTAM), netparm specifies the name defined in VTAMLST (as a cross-domain resource) that the remote CAICCI uses as its APPLID.
- For TCPIPGW or TCPSSLGW, netparm specifies the TCP/IP Hostname or address of the remote CAICCI host. This is a positional operand that requires two commas as a place holder if it is not specified. When not specified, the IP information is obtained from the remote CAICCI specified by the sysid operand.

  The IP address may be specified in IPv6 format. For information in IPv6 addressing, see the notes in CAICCI Control Option Summary (see page 131).

  :port

  The port operand specifies the port number on which the gateway will connect to the remote host. It is only valid when specifying a protocol of either TCPIPGW or TCPSSLGW. The port number needs to be specified if the listening port of the remote host is different than the port number specified with the PROTOCOL control option. If the port number is specified, it must be appended to the hostname or IP address prefixed with a colon(:).

  ;keyword=value

  The keyword operand specifies SSL Keywords and their values. It is only valid when specifying a protocol of either TCPIPGW or TCPSSLGW. Each keyword=value must be prefixed with a semi-colon (;).

  The following is a list of valid keywords, their abbreviations, and allowable values (case insensitive). Each keyword=value parameter must be prefixed with a semicolon (;) as a delimeter.

  - SECURE | SEC | S =

    Indicates whether or not to request an SSL (secured) connection (default is set from the PROTOCOL UNSECURED_CONNECT setting):

    YES | Y - Connection must be secure. If the remote system does not support or is not enabled for SSL, the connection request fails.

    ALLOW | A – An unsecured connection is allowed and requested. If the remote CAICCI **requires** an SSL connection, then a secure connection is permitted and made.

    DEFER | D – Connection type defers to the remote system: If the remote CAICCI supports and is enabled for SSL, then it connects securely; otherwise the connection is made unsecured.

    NO | N - Only unsecured connections allowed. If the remote system **requires** an SSL connection, the connection request fails.

retry

Specifies the re-poll time in minutes that CAICCI uses to attempt to re-establish a session with the remote CAICCI using the netparm specified. This time ranges from 1 to 59 minutes. This operand has a default value of 10. This is a positional operand that requires two commas as a place holder when the default value is to be used.

A retry time of zero for a NODE will generate the messages:

```
CAS9604W-CAICCI-INVALID RETRY TIME SPECIFIED
```

```
CAS9604W-CAICCI-NODE ENCOUNTERED ERRORS
```

sysid

Specifies the unique 1 to 8 character identifier of the remote CAICCI system.

maxru

Specifies the maximum data packet size, specified in decimal bytes, that is allowed to be transmitted between the local CAICCI and this remote CAICCI node.

The default MAXRU value is 4096 or the value set by the MAXRU control option.

start/stop

The control words used to specify when the session with the remote CAICCI is to be established and terminated.

Acceptable values are:

START/SHUT

Start link at CAICCI startup time. Drop link when CAICCI shuts down. START/SHUT is the default.

START/TERM

Start link at CAICCI startup time. Drop link when the first CA application issues a CCI TERM.

INIT/SHUT

Start link when the first CA application issues a CCI INIT. Drop link when CAICCI shuts down.

INIT/TERM

Start link when the first CA application issues a CCI INIT. Drop link when the first CA application issues a CCI TERM.

Note: For z/OS, VM, and VSE, this operand should be specified as START/SHUT. Other operands available are shown but should not be selected unless specifically requested by the installation procedures of the CA solution you are installing.

netparm2

For LU0, this is the logmode name for assigning a class of service (COS) to the session with the remote CAICCI node.

Important! The operands STOP and SHUT are fully compatible and can be substituted for one another.

Examples (console)

```
CCI NODE(LU0,A73CVC01,01,A73SYSID,4000,START/SHUT)

CCI NODE(TCPIPGW,141.202.1.114:1721,1)

CCI NODE(TCPSSLGW,,01,A93S)
```

Examples (ENFPARMS)

```
NODE(LU0,A73CVC01,01,A73SYSID,4000,START/SHUT)

NODE(TCPIPGW,141.202.1.114:1721,1)

NODE(TCPSSLGW,USILCA11:1721;SECURE=D,1,ALLSENF)
```

NODEDATA([DISPLAY|RESET,]protocol,sysid,netparm)

The NODEDATA control option allows additional information to be appended to the netparm data of a specific GATEWAY, NODE, or PROTOCOL.

Netparm data on the GATEWAY, NODE, and PROTOCOL control options is restricted to 32 characters.

Both SSL and its numerous keyword=value options and IPv6 can require netparm data greater than 32 characters. This control option allows an additional 224 characters (256 total).

Each NODEDATA control option allows up to 52 characters to be appended so multiple NODEDATA commands for a particular protocol and sysid can be coded until the 256 character limit is reached.

A NODEDATA control option does not have to utilize all 52 characters before another NODEDATA control option is coded. Each NODEDATA control option appends whatever is specified onto the existing netparm.

[DISPLAY]

Displays netparm rather than modifying it.

If a specific protocol **and** sysid are provided, then only the netparm for that GATEWAY|NODE|PROTOCOL is displayed.

If only the protocol **or** the sysid is specified, then the netparm is displayed for each GATEWAY|NODE|PROTOCOL that matches the specified string.

If neither a protocol or sysid is provided, then the netparm of every GATEWAY|NODE|PROTOCOL is displayed.

[RESET]

Requests a RESET or clear of the netparm data for the specified GATEWAY|NODE|PROTOCOL for the specified sysid.

A protocol and sysid are required.

Once reset, a subsequent display request shows the netparm data to be ***NULL***.

protocol

Specifies the protocol used in selecting the specific GATEWAY|NODE|PROTOCOL control option for modifying or displaying its netparm.

A protocol must be specified when appending or resetting netparm data.

Possible values are:

LU0|TCPIP|TCPSSL|TCPIPGW|TCPSSLGW|XES|XCF|VTAM

sysid

Specifies the name of the sysid used in selecting the GATEWAY|NODE|PROTOCOL for appending, displaying, or resetting its netparm data.

If sysid matches the name of the local CAICCI, then only PROTOCOLs are searched; otherwise GATEWAYs and NODEs are searched.

A sysid must be specified when appending or resetting netparm data.

netparm

Specifies information to be appended to the existing netparm. A maximum of 52 characters can be coded. Strings may be continued across NODEDATA statements. For details on netparm parameter usage, see the PROTOCOL (see page 154) control option.

For information on IPv6 address identification, see the notes in CAICCI Control Option Summary (see page 131).

NODEDATA has the following abbreviations:

ND

Operates the same as NODEDATA.

DND

Displays netparm the same as using the [DISPLAY] option.

RND

Resets netparm the same as using the [RESET] option.

Examples (console)

```
CCI DND
CCI DND(protocol)
CCI DND(,sysid)
CCI DND(protocol,sysid)
CCI RND(protocol,sysid)
CCI NODEDATA(protocol,sysid,[####:####:####:####:####:####:####:####]:port#)
CCI ND(protocol,sysid,[####:####:####:####:####:####:####:####]:port#)
```

```
CCI NODEDATA(DISPLAY)
CCI NODEDATA(DISPLAY,protocol)
CCI NODEDATA(DISPLAY,,sysid)
CCI NODEDATA(DISPLAY,protocol,sysid)
CCI ND(DISPLAY)
CCI ND(DISPLAY,protocol)
CCI ND(DISPLAY,,sysid)
CCI ND(DISPLAY,protocol,sysid)
CCI NODEDATA(DISP)
CCI NODEDATA(DISP,protocol)
CCI NODEDATA(DISP,,sysid)
CCI NODEDATA(DISP,protocol,sysid)
CCI ND(DISP)
CCI ND(DISP,protocol)
CCI ND(DISP,,sysid)
CCI ND(DISP,protocol,sysid)
CCI NODEDATA(RESET,protocol,sysid)
CCI ND(RESET,protocol,sysid)
```

Example (ENFPARMS)

```
DND
DND(protocol)
DND(,sysid)
DND(protocol,sysid)
RND(protocol,sysid)
NODEDATA(protocol,sysid,[####:####:####:####:####:####:####:####]:port#)
ND(protocol,sysid,[####:####:####:####:####:####:####:####]:port#)
NODEDATA(DISPLAY)
NODEDATA(DISPLAY,protocol)
NODEDATA(DISPLAY,,sysid)
NODEDATA(DISPLAY,protocol,sysid)
ND(DISPLAY)
ND(DISPLAY,protocol)
ND(DISPLAY,,sysid)
ND(DISPLAY,protocol,sysid)
NODEDATA(DISP)
NODEDATA(DISP,protocol)
NODEDATA(DISP,,sysid)
NODEDATA(DISP,protocol,sysid)
ND(DISP)
ND(DISP,protocol)
ND(DISP,,sysid)
```

```
ND(DISP,protocol,sysid)
NODEDATA(RESET,protocol,sysid)
ND(RESET,protocol,sysid)
```

Important: In IPv6, the expanded addressing requires coded delimiters to specify the beginning and end of the address separate from the port number. In the following example, the coded brackets, "[" and "]", delimit the start and end of the IPv6 address.

Example (IPv6)

```
NODEDATA(TCPSSLGW,A11SENF,[fd00:7a06:a20:100::11]:7001)
```

PERMIT(ALLINONE)

Note: PERMIT(ALL|NONE) applies to LU0 only.

The PERMIT control option is used to restrict other systems from logging on to your system if NODE (or GATEWAY) and CONNECT statements have not been defined on your environment. The default is ALL.

This control option has the following operands:

ALL

Allows systems to log on. ALL is the default.

NONE

Prevents other systems from logging on.

Note: CA solutions running on PCs or TCP/IP will be able to log on if NONE is specified.

Example (console)

```
CCI PERMIT(ALL)
```

Example (ENFPARMS)

```
PERMIT(NONE)
```

PROTOCOL(protocol,netparm,retry,sysid,maxru,start/stop)

The PROTOCOL control option is used to specify the communications protocol supported by this local host node and its associated network parameters to access remote CAICCI nodes. Default: retry=10, maxru=4096, start/stop=START/SHUT.

All parameters must be entered.

protocol

Specifies the desired communication protocol to be used. Possible values are:

VTAM

Activates VTAM drivers to enable host-to-host connections using LU0.

XCF

Specifies that the cross system communication facility be used for host-to-host connectivity of those systems within a common sysplex.

XES

Specifies that the coupling facility be used for host-to-host connectivity of those systems within a common sysplex.

TCPIP

Activates host-to-PC connection using IBM TCP/IP or CA TCPaccess Communications Server for z/OS.

TCPIPGW

Activates host-to-host connection using IBM TCP/IP or CA TCPaccess Communications Server for z/OS.

TCPSSL

Activates host-to-PC connection using IBM TCP/IP or CA TCPaccess Communications Server for z/OS with Secure Sockets Layer (SSL).

TCPSSLGW

Activates peer-to-peer connection using IBM TCP/IP or CA TCPaccess Communications Server for z/OS with Secure Sockets Layer (SSL).

netparm[:port][;keyword=value...]

Specifies the network communication establishment parameters:

- For VTAM, this is the primary LU name defined in VTAMLST that CAICCI uses as its APPLID. This node should be defined and active to VTAM before this command is entered.
- For XCF, this is the 1 to 6 character identifier used to build the XCF member name. It does not need to be a unique identifier for each machine.
- For XES, this is the structure name predefined within the coupling facility used by CAICCI for cross-communication within a sysplex.
- For TCPIP, this is the PORT number. The default is 1202.
- For TCPIPGW this can be the subsystem name:port number. The PORT number default is 1721

- For TCPSSL and TCPSSLGW, you can specify the following optional port and keyword values:

  :port

  Specifies the listening port number for TCPSSL or TCPSSLGW. If the port number is specified, it must be appended to the hostname or IP address prefixed with a colon(:). The default port number for TCPSSL is 1202; the default port number for TCPSSLGW is 1721.

  ;keyword=value

  The following is a list of valid keywords, their abbreviations, and allowable values (case insensitive). Each keyword=value parameter must be prefixed with a semicolon (;) as a delimeter.

  - TCP=

    The TCP/IP stack name. The default is all active TCP/IP stacks.

  - UNSECURED_CONNECT | UNSECON | US =

    Indicates whether or not to accept non-SSL (unsecured) clients:

    NEVER | N - (default) Remote Hosts or PCs not supporting or enabled for SSL are denied a connection.

    ALLOW | A - All connections will be unsecured unless the PC supports and REQUIRES an SSL connection.

    NONSSL | NS - PCs not supporting SSL (pre-version 1.1.7) are allowed to connect unsecured. PCs supporting and enabled for SSL will connect secured.

    ONLY | O - Only unsecured connections allowed. PCs supporting and requiring SSL are denied a connection.

  - CLIENT_AUTH | CLAUTH | CA =

    Indicates whether PC Client Certificates should be authenticated:

    NO | N (default)

    YES | Y

    PASS | P - The client certificate is not authenticated but is still requested for user exit validation.

  - REMOTE_AUTH | RMAUTH | RA =

    Indicates whether PC Client Certificates should be authenticated:

    NO | N

    YES | Y (default)

    PASS | P - Client certificate is not authenticated but is still requested for user exit validation.

- CERT=

  Specifies the Server Certificate Label Name

  '*' - Use the certificate whose label is "CCIPC" (for TCPSSL) or “CCIGW” (for TCPSSLGW). If not found, use the certificate whose label is the local CAICCI sysid. If not found, use the certificate whose label is "CCI".

  'label' - Use the certificate whose name is label.

  '(null)' - (default) Use the SystemSSL default certificate.

Note: Embedded blanks within Certificate Label Names are not supported.

- KEYRING=

  Specifies the name of the external security keyring (Used in lieu of an HFS key database)

- SSL_VERSION | SSLV | SV =

  Specifies the version of System SSL that TCPSSL should use to request SSL services:

  1 - Version 1 (OS/390 version)

  2 - Version 2 (z/OS 1.2 version)

  '(null)' - Use highest available version (default)

- PROT=

  Specifies which security protocol(s) should be enabled:

  SSL - Only SSL Version 3 (default)

  TLS - Only TLS Version 1

  SSL/TLS | TLS/SSL | S/T | T/S | BOTH - Both SSL Version 3 and TLS Version 1 are enabled

- CIPHER_SUITE | CIPHERS | CIPHER | CI | CS =

  Specifies the choice of one or more SSL (Version 3) cipher suites in the order of usage preference, for CAICCI packet encryption in the form of 'xxyyzz...'

  The cipher suite values are:

  '01' - NULL MD5

  '02' - NULL SHA

  '03' - RC4 MD5 Export

  '04' - RC4 MD5 US

  '05' - RC4 SHA US

  '06' - RC2 MD5 Export

  '09' - DES SHA Export

  '0A' - 3DES SHA US

  '2F' - 128-bit AES SHA US

  '35' - 256-bit AES SHA US

  IBM - Use the System SSL default list: '0504352F0A090306020100'

  3DES - (default) Use the System SSL default list but put 3DES SHA US at the top of the list: '0A0504352F090306020100'

  AES128 | AES-128 - Use the System SSL default list but put 128-bit AES SHA US at the top of the list: '2F0504350A090306020100'

  AES | AES256 | AES-256 - Use the System SSL default list but put 256-bit AES SHA US at the top of the list: '3505042F0A090306020100'

- SSLTRACE | SSLT | ST =

  Specifies the name of the HFS file where the System SSL can write trace entries. (Specifying the file name turns on tracing!)

- SSLDUMP | SSLD | SD =

  Indicates whether SSL packets should be dumped to the Trace File (TRCPRINT):

  NO | N - (default)

  YES | Y

- CALLBACK_DLL | CBDLL =

  Specifies the module name of the DDL containing the user exit routine for validating client

  (and server) certificates.

retry

The re-poll time in minutes that CAICCI uses to attempt to re-establish a session with the specified network transport specified. This time ranges from 1 to 59 minutes and has a default value of 10.

This is a positional operand that requires two commas as a place holder when the default value is to be used.

A retry time of zero for a PROTOCOL will generate the messages:

```
CAS9604W-CAICCI-INVALID RETRY TIME SPECIFIED
```

```
CAS9604W-CAICCI-PROTOCOL ENCOUNTERED ERRORS
```

sysid

A unique 1 to 8 character identifier that is used for this CAICCI system. This identifier must be kept unique within the entire CAICCI system network. This operand is required, and must be the same as specified with the SYSID control statement.

maxru

The maximum data packet size, specified in decimal bytes, that is allowed to be transmitted between the local CAICCI and the remote CAICCI nodes.

The default MAXRU value is 4096 or the value set by the MAXRU control option statement.

Note: For any of the TCP/IP protocols, the MAXRU default of 4096 is conservative and should be changed to a higher value for more efficient operation.

start/stop

The control words used to specify when the LU-to-SSCP session is to be established and terminated. Acceptable values are:

START/SHUT

Start link at CAICCI startup time. Drop link when CAICCI shuts down. START/SHUT is the default.

START/TERM

Start link at CAICCI startup time. Drop link when the first CA application issues a CCI TERM.

INIT/SHUT

Start link when the first CA application issues a CCI INIT. Drop link when CAICCI shuts down.

INIT/TERM

Start link when the first CA application issues a CCI INIT. Drop link when the first CA application issues a CCI TERM.

Note: The last operand refers to when the ACB should be opened and when it should be closed. For z/OS, VM, and VSE, this operand should be specified as START/SHUT. Other operands available are shown but should not be selected unless specifically requested by the installation procedures of the CA solution you are installing.

Important! The operands STOP and SHUT are fully compatible and can be substituted for one another.

Example (console)

```
CCI PROTOCOL(VTAM,A97CVC01,01,A97SYSID,4000,START/STOP)

CCI PROTOCOL(TCPIPGW,SSID=ACSS:1721,1,USI273ME)
```

Example (ENFPARMS)

```
PROTOCOL(VTAM,A97CVC01,01,A97SYSID,4000,START/STOP)

PROTOCOL(TCPIP)

PROTOCOL(TCPIPGW,1721,1,USI273ME)

PROTOCOL(TCPSSLGW,1721;US=NS;CI=3DES;CERT=*,1,A97S)

PROTOCOL(TCPSSLGW,7001;CI='352F0A',01,A73S)
```

PROTSEC(XCF|XES|IXES,XCF|XCF,XES)

Use PROTSEC to specify XCF and/or XES type links as secure connections. This parameter does not enable encrypted data to flow on said connections, but rather to flag XCF and XES type links as secure physical connections. The PROTSEC parm must be specified on both systems involved in an XCF or XES connection for the link to be considered secure.

Note that the PROTSEC parm may not function as a CAICCI console command since it must be issued before a link becomes active. Most links become active at CAIENF startup time, making it necessary to insert PROTSEC before any NODE or CONNECT statements in the CAICCI parameter file. You can issue PROTSEC as a CCI command if you take action to ensure that the XCF link starts after the command is issued on both ends of the connection.

To make the connection after PROTSEC is issued as a console command

- Issue the PROTOCOL(XCF,...) statement as a command after the PROTSEC command. Make sure the PROTOCOL(XCF,...)is not contained in the CAICCI parameters statements.

or

1. Use the z/OS SETXCF STOP,PATHIN,DEVICE=xxx command to force a disconnection.
2. Issue the PROTSEC command on both systems involved.
3. Issue the z/OS SETXCF START,PATHIN,DEVICE=xxx command to reestablish the connection.

Options that must be entered are shown in the following table. Separate multiple options with a comma.

Operand	Description
XCF	Flag XCF links as secure physical links.
XES	Flag XES links as secure physical links.

Example (ENFPARMS)

```
PROTSEC(XCF)
PROTSEC(XES)
PROTSEC(XCF,XES)
PROTSEC(XES,XCF)
```

REMOVE(SYSID,xxxxxxxx)

The REMOVE control option is used to remove a NODE or GATEWAY entry. If a NODE or GATEWAY entry was entered incorrectly, the REMOVE command can be used to remove the control block built by the CONNECT statement.

This control option has the following operand:

xxxxxxxx

A unique 1 to 8 character identifier that is used for this CAICCI system. This identifier must be kept unique within the entire CAICCI system network.

This operand is required.

Example (console)

```
CCI REMOVE(SYSID,SYSIA97S)
```

SYSID(sysid)

Use SYSID to define a name for the local CAICCI system. This control option is required. If this control option is entered as a command when CAICCI is active, you receive the following message:

```
CAS9604W - CAICCI - Can not update SYSID while CCI Active
```

This control option has the following operands:

sysid

A 1 to 8 character identifier that is unique throughout the CAICCI network.

This parameter is required.

Example (ENFPARMS)

```
SYSID(A97SYSID)
```

SYSPLEX(sysplexid)

Indicates that the local CAICCI host is a member of the specified sysplex.

By defining two or more CAICCI sysids as parts of a sysplex, CA solutions residing on these hosts may be collectively targeted by their common sysplexid and allow CAICCI to distribute the workload among the individual CA solutions.

This command has the following operand:

sysplexid

A 1- to 8-character identifier. The name is not required to be the actual sysplex name in a sysplex environment but defaults to that name if this parameter is omitted. A sysplexid may be assigned to a host that is not physically within a sysplex.

VARY(ACT|INACT,SYSID,sysid)

The VARY command can be entered using the z/OS console to connect or disconnect a remote node.

Note: If this command is entered on CPU A for SYSID of CPU B, CPU B will attempt to reconnect to CPU A and fail every retry time interval.

This command has the following operands:

ACT

Connects to a remote node.

INACT

Disconnects from a remote node.

sysid

A 1- to 8-character identifier that is unique throughout the CAICCI network.

Example (console)

```
CCI VARY(INACT,SYSID,A97S)
```

Chapter 5: CAICCI Component Trace

This section contains the following topics:

CAICCI Component Trace Details

For information on starting and stopping a Component Trace External Writer and to connect or disconnect the writer from a component trace component, see the IBM manual, *z/OS MVS System Commands* (located under TRACE CT).

A sample PROC, CCIXWTR, is provided in CAI.CAW0PROC, which invokes the IBM external writer ITTTRCWR. For editing, see the IBM manual *Diagnostic Tools and Service Aids*. The PROC must be available for the MVS operator command TRACE CT. You may need to place PROC in SYS1.PROCLIB, IEFJOBS, or IEFPDSI).

Component Trace provides for automatically generating a TRACE CT command when component tracing is initialized. This is performed using a PARMLIB member that can be specified on the CCICT control statement to initialize component tracing. The PARMLIB member can indicate that the TRACE be set ON, define and connect an External Writer, select ASIDs and JOBNAMEs for tracing, and specify OPTIONS (described below).

A sample member, CTECCI00, is shipped in CAI.CAW0OPTN. If this member is used, it must be moved to an MVS PARMLIB data set and its name specified on the CCICT control statement. As delivered, if CCICT is added as a CAICCI parameter, then CTECCI00 will be used and as delivered, CTECCI00 will tell component trace to start the CCIXWTR started task. The CCIXWTR task must be customized to use a particular default output CTRACE data set.

CTECCI00 will also set TRACE ON, connect to its CCIXWTR External Writer and collect trace records from all CAICCI applications for all currently defined Event IDs.

Command Format

In addition to using the CCICT CAICCI parameter to start the component trace, you may also start the External Writer and trace manually. You can also restart a trace when a previous trace was stopped. Use the z/OS TRACE CT command for this.

These examples assume that the Component Trace name for CAICCI is defined to be CACCI.

To start an External Writer that is using the CCIXWTR task, issue this z/OS command:

TRACE CT,WTRSTART=CCIXWTR

To start a trace, use one of the following two commands:

TRACE CT,ON,COMP=CACCI

This command, when entered, generates a WTOR which must be responded to as such:

```
R nn,[,OPTIONS=()][,ASID=()] [,JOBNAME=()]
[,WTR=CCIXWTR|DISCONNECT]
```

ASID=(),[,JOBNAME=():

Specifies which applications are to be traced. If no ASIDs or JOBNAMEs are specified, all CAICCI applications are traced. If any ASID or JOBNAME is specified, only these applications will be traced. Up to 16 ASIDs and JOBNAMES can be specified.

WTR=CCIXWTR:

This parameter requests that the CAICCI Trace Component be connected to its External Writer CCIXWTR.

WTR=DISCONNECT:

This parameter requests that the CAICCI Trace Component be disconnected from its External Writer.

OPTIONS=():

The CAICCI Start/Stop routine accepts the following for OPTIONS:

- eid1,eid2,…
- EXTEND(#buffers)
- RESET(#buffers,buffer_size)
- DUMPON
- DUMPOFF
- PAUSE
- RESUME

eid1, eid2, …

Enables one or more Event IDs (EIDs). #CCICT macro calls that specify one of these EIDs will now commence capturing trace data. Currently implemented EIDs are:

APIE – Traces entry into the API

APIF – Traces API call failures

APIX – Traces API exit to the caller

SRBR – Traces execution through the Receiver SRB

SRBS – Traces execution through the Sender SRB

EXTEND(#buffers)

Extends the current trace data space by the specified number of buffers. Trace data within the existing buffers is unaffected. Tracing must have been previously set ON before this option can be used.

RESET(#buffers,buffer_size)

Resets the current trace data space to have the specified buffer_size (in Kilobytes) and #buffers. Trace data within the existing buffers is lost. If either #buffers or buffer_size is omitted, the current value is used. Omitting both values causes the existing trace buffers to be cleared. This option can be used at any time.

PAUSE and RESUME

PAUSE internally pauses tracing. RESUME resumes tracing. The current tracing criteria (#buffers, buffer_size, JOBNAMEs, ASIDs, EIDs) is not affected.

This command can be re-entered to modify the existing trace settings. However if ASIDs, JOBNAMEs or EIDs are specified, their previous settings are lost. The new values overwrite the existing values.

TRACE CT,,COMP=CACCI,PARM=PARMLIB_member

This command, when entered reads and runs PARMLIB_member from SYS1.PARMLIB. This is the same PARMLIB member that you specify with the CAICCI CCICT. Again, a sample member, CTECCI00, is shipped in CAI.CAW0OPTN.

To stop an External Writer that is using the CCIXWTR task, issue this z/OS command:

TRACE CT,WTRSTOP=CCIXWTR

To stop an active CAICCI trace, issue this z/OS command:

TRACE CT,OFF,COMP=CACCI

Chapter 6: CAIENF Control Options

CAIENF supports a variety of control options that allow you to tailor various CAIENF system functions. The commands that CAIENF supports fall into the following categories:

- System control options
- Event, data item, or application-specific commands
- Diagnostics

Initial CAIENF options can be specified using a parameter file referenced by the ENFPARMS DD statement in the CAIENF started task JCL, using the PARM= field on the EXEC JCL statement, or directly on the z/OS START command. After startup, most CAIENF control options can be dynamically altered at any time by entering the appropriate operator command from any z/OS console.

To set a CAIENF control option at startup time, code the desired command within the CAIENF parameter file (DDNAME 'ENFPARMS'), on the EXEC statement within the CAIENF procedure, or on the START command used to invoke CAIENF.

Control options entered on the z/OS START command have priority over those entered on the EXEC JCL statement. Parameters on the EXEC JCL statement in turn have priority over those entered on the parameter file. This scheme allows a parameter file option to be easily overridden at startup time, if desired.

The parameter file is a sequential file allocated to the CAIENF started task. The data set specified for the CAIENF parameter file must be LRECL=80 and RECFM=FB. Control options are specified free-form in columns 1 through 72 and are not required to appear in any special order. Note that when coding parameter file options, CAIENF considers any option that starts with an asterisk in column 1 to be a comment.

The following is a valid sample for a CAIENF parameter file (all lines begin in column 1):

```
*
* CAIENF parameters
*
ARCHIVE(0100)
CANCEL(NO)
DIAG(OFF)
EVENT(JOBTERM,RECORD)
SYSOUT(A)
TIMER(10,100)
TRACE(64)
  .
  .
  .
```

Most control options specified at startup time can also be dynamically altered with the CAIENF operator command. As an example, entering ENF ARCHIVE(AUTO) changes the ARCHIVE control option. Any z/OS console or operator command facility can be used to enter CAIENF commands.

This section contains the following topics:

Special DCM and EVENT Utility

The ENFUTIL utility has been provided to create the new DCM and EVENT control option statements required to upgrade from CA Common Services r11. This utility uses a CA Common Services r11 DB detail listing as input.

When creating the new control statements, this utility does not take into account any DCMs or EVENTs that are no longer valid in the current release of the product supplying the DCM.

Use the following JCL to obtain a CA Common Services r11 detail listing.

```
//CAS9DB   EXEC PGM=CAS9DB,REGION=4096K,TIME=1440
//STEPLIB  DD DSN==YOUR.COMMON.SERVICES.R11.CAILIB,DISP=SHR  ===> UPDATE
//DBOUT    DD DISP=(NEW,CATLG),
//         DSN=YOUR.R11.ENFDB.LISTING,                  ===> UPDATE
//         UNIT=3390,VOL=SER=??????,SPACE=(CYL,(1,1)),
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=6118)
//DBIN     DD *
LIST DB(*) DETAIL
/*
```

Important! You must update the STEPLIB statement with the DSN of your CA Common Services r11 CAILIB before you submit the JCL.

Once you have obtained a detail listing, see the CA Common Services SAMPJCL member ENFUTIL for instructions on how to edit the JCL. The output from the ENFUTIL job can be placed in the member or data set referenced by ENFPARMS DD in your CAIENF startup procedure.

Control Options Summary - CAIENF

The CAIENF control options are summarized in the following table.

Note: Numbers in the Footnote column following the option indicate notes after the table.

Control Option	Footnote	Short Name	Default
APPL		AP	N/A
ARCHIVE(hhmm)	2, 5	BA	0100
AUTOCMDS	1	-	N/A
CANCEL(opt)		CAN	NO

Control Option	Footnote	Short Name	Default
CASE(opt)	3	CAS	M
CATALOG(opt)	2	CAT	YES
DB2STAT[,dsn1,...]		N/A	N/A
DCM(object,ddname)	4	N/A	N/A
DCOL(data_name)		DC	N/A
DETAIL(nn)	2	DET	7
DIAG(opt)		DI	OFF
DSNAME(dsn)	2	DSN	ENF.ARCHIVE
DUMP		DU	N/A
ENDIF	4	-	N/A
ENFCT		-	-
EVENT(name,opt)		EV	N/A
IF	4	-	N/A
LABEL(option)	2	LA	SL
MAP		-	N/A
MAXQTIME		MAXQT	10
MSGEVENT		MS	N/A
NODB	4	-	N/A
PURGE(event)		PUR	N/A
RECORD(opt)		REC	YES
REFRESH(module)		REF	N/A
REINIT		REIN	N/A
RESETSTA		RES	N/A
RETPD(nnnn)	2	RET	30
SCREEN(event,data,operand,mask)		SCR	N/A
SELECT(event,data,operand,mask)		SEL	N/A
SPACE(nnn,nnn)	2	SPA	0,0
STATUS(opt)	0	STA	N/A
SVCDUMP		SVC	N/A

Control Option	Footnote	Short Name	Default
SYNCDUMP		-	No
SYSOUT(class,dest)		SY	A,LOCAL
TASKLIB(dsn{,opt_vol}|CLOSE)		-	CLOSE
TIMER(min,max)		TI	500,6000
TRACE(nnnn[,loc])		TR	32
UNIT(opt)	2	UN	TAPE

Notes:

- (0) The abbreviation for STATUS has been changed from ST to STA.
- (1)This command is valid only when used with the CAIENF START command:

  ```
  (S ENF,,,control_option).
  ```

- (2) This command is required for CAIENF database backup.
- (3) If only CAIRIM is running, the system default is U (uppercase). This command affects all CAS9 messages.
- (4) This command is valid only at CAIENF startup time.
- (5) The ARCHIVE control option replaces BACKUP from previous releases. BACKUP may still be coded, but its functionality may be different in future ENF releases. BACKUP should be changed to ARCHIVE.

APPL

This command produces a summary of all currently active applications by responding with an informational message. The message presents each application currently listening for CAIENF events, and the number of events for which each application is currently listening. The message also presents the status of each application as one of the following six possible conditions.

WAIT

The application is waiting for events from CAIENF.

ACT

The application is processing CAIENF events.

WAIT DIS

The application is waiting for events from CAIENF, but has been disabled by the ENF dispatcher.

WAIT RECV

The application is waiting while recovering CAIENF events.

ACT DIS

The application is processing CAIENF events, but has been disabled by the ENF dispatcher.

ACT RECV

The application is processing CAIENF recovered events.

AUTOCMDS

If specified as part of a CAIENF START command (S ENF,,,AUTOCMDS), this control option executes all CAIENF auto commands in the ENFCMDS file. If CAIENF is brought down, you can re-execute all CAIENF auto commands in the ENFCMD file when you recycle CAIENF without re-IPLing your system.

The AUTOCMDS control option can only be specified with the CAIENF Start command (S ENF,,,AUTOCMDS).

Note: This command cannot be specified as a parameter in the CAIENF parameter file.

ARCHIVE(hhmm) or BACKUP(hhmm)

Controls the time of day that the automatic database archive and compression routine is scheduled; the default is 0100. The following table shows the result of ARCHIVE commands with different available options.

ARCHIVE takes the place of BACKUP for this release of CAIENF. BACKUP can still be coded and will perform the same as ARCHIVE, but you will receive a warning message with instructions to use ARCHIVE instead. BACKUP may be modified in the future.

Examples

ARCHIVE(0600)

Automatic database backup is invoked at 6:00 AM.

ARCHIVE(AUTO)

Automatic database backup is invoked when the CAIENF database reaches 80% utilization. Runs once daily.

ARCHIVE(AUTO-*nn*)

nn is a percentage between 50 and 95, inclusive. The database is automatically backed up when the database reaches *nn*% utilization.

ARCHIVE(NOW)

Forces an immediate backup.

ARCHIVE(OFF)

Automatic database backup is not active.

Notes:

- CA recommends that the variable time option in the format **hhmm** be specified because it makes the CAIENF ARCHIVE more predictable. However, be advised that if you specify the variable time option and the database is full at the time of ARCHIVE, the database may need to be reformatted when the ARCHIVE finishes.
- ARCHIVE(OFF) can be specified if neither timed nor automatic ARCHIVE is desired.
- ARCHIVE(NOW) allows you to force an unscheduled archive at the point the ARCHIVE command is issued.

The PURGE operator command can be used to clear event data that is not critical to running your CA Technologies solutions.

CANCEL(option)

Controls whether CAIENF can be canceled or not; the default is NO.

This command has the following options:

YES

Allows CAIENF to be canceled at any time.

NO

Does not allow CAIENF to be canceled at any time unless a CANCEL(YES) command is entered.

Note: Forcing or canceling the CAIENF address space has adverse effects on products that use CAIENF services. This should be avoided and used only when there are no other options available during a crisis situation or it may force an unscheduled IPL. During CAIENF shutdown, the address space becomes cancelable. This allows CAIENF to be cancelled if shutdown does not complete.

CASE(option)

Controls whether messages generated by CAIENF are translated to uppercase or displayed in mixed case. The default is mixed case unless CAIRIM is the only component running, in which case the default is uppercase.

Valid options are:

U

Translates all messages to uppercase.

M

Displays all messages in mixed case.

CATALOG(option)

Controls dynamically allocated backup data sets; the default is YES. If cataloging of backup data sets is not desired, specify CATALOG(NO).

DB2STAT[,dsn1,dsn2,...,dsn7]

Allows the display of CAIENF/DB2 thread table usage statistics. To show all subsystems, issue the command without a DB2 subsystem name (dsn). Up to seven subsystem names can be included in the command.

Example

From the console issue:

```
ENF DB2STAT,DSN1,DSN2
```

DCM(object,<ddnamel*>)

DCM configuration statements specify the name and location of pre-compiled Data Control Modules. These modules contain information on the events being monitored by the CAIENF address space. In previous CAIENF versions, DCMs were installed into the CAIENF database using an independent batch utility program (CAS9DB). With the elimination of CAS9DB in R12, DCMs are defined directly to CAIENF through DCM configuration statements and are installed (or verified) at CAIENF startup.

Note: When upgrading from CA Common Services r11, a utility can be used to create the new DCM control option statements required for CA Common Services Version 12.0. For more information, go to Special DCM and EVENT Utility (see page 171).

object is the required PDS member name of the DCM object.

ddname is the optional DDName of the PDS data set where the DCM object resides. If ddname is omitted, DDName “CAIDCM” (defining the default DCM object data set) must be present in the CAIENF started task JCL, and this library will be used to load DCM objects. ddname may also be specified as a single asterisk, indicating the DCM object resides in the CAIENF started task JOBLIB/STEPLIB concatenation, or the LPALST/LNKLST concatenation.

Example

DCM(CAS9DCM2)

DCM(CAS9DCM4)

DCM(CARRDCM0,USSLIB)

DCM(SYSVDCM,*)

In these examples, CAS9DCM2 and CAS9DCM4 will be loaded from the data set defined by DDName “CAIDCM” (default). CARRDCM0 will be loaded from the data set defined by DDName “USSLIB”. SYSVDCM will be loaded from the CAIENF started task STEPLIB/JOBLIB concatenation, or from the z/OS LNKLST/LPALST concatenation if it is not found in STEPLIB/JOBLIB.

Usage Notes

The ddname parameter allows the loading of DCMs from CA product target libraries without having to include them in the CAIENF started task JOBLIB/STEPLIB concatenation.

If ddname is specified, the search for the DCM object is limited to the specified data set(s) to prevent inadvertent loading of a "system" copy of a DCM.

DCM data sets require APF authorization. A DCM load will fail (error code X'306) if the DCM data set is not authorized.

DCM configuration statements follow the same syntax rules as any other CAIENF configuration statement or initialization parameter, and they must be placed in the CAIENF started task ENFPARMS data set (that is, they cannot be placed in the ENFCMDS data set).

If any type of error occurs during DCM processing, an error message is posted to the CAIENF log describing the failure (along with any applicable return/reason codes), and the CAIENF started task abnormally terminates immediately. Also, if no DCM statements are specified in the ENFPARMS data set, the CAIENF started task abnormally terminates.

DCOL(data_name)

The DCOL option controls processing for data collection routines.

- If (data_name) is not entered, the control option presents a summary of all of the data collection routines currently active.
- If just (data_name) is entered, a detailed description of the data collection routines is presented.

Example (console)

```
ENF DCOL
ENF DCOL(DATEL)
```

Example (ENFPARMS)

```
DCOL
```

DETAIL(nn)

Specifies the number of days that CAIENF event data is kept on the CAIENF database. nn can be specified as a one or two digit number from 1 to 99. The default is 7.

DIAG(options)

Controls various CAIENF diagnostic options; the default is OFF.

Valid options are:

OFF

Turns off all diagnostic options.

ALL

Requests all diagnostic functions.

DB

Requests diagnostic data from the database.

AS

Requests diagnostic data from the CAIENF address space modules.

EV

Requests diagnostic data from the event interface.

AP

Requests diagnostic data from the application interface.

Important! DIAG(AP) must be entered after the TRACE(nnn,CSA) option, otherwise it is not recognized and the following error appears: CAS9248E-CA-END DIAG(AP) only valid with TRACE in CSA.

DSNAME(dsn)

Controls dynamically allocated backup data sets. The data set name specified is used as a prefix; the default is ENF.ARCHIVE. The time and date of the backup are appended in the form: Dyyddd.Thhmm to the data set name.

The DSNAME prefix for dynamically allocated backup data sets can only be up to 16 characters in length including periods.

The following example would not be accepted because it contains 18 characters:

```
ENF.ARCHIVE.BACKUP
```

The following example would be accepted, since it contains 16 characters:

```
ENF.ARCHIVE.BKUP
```

DUMP

Provides a formatted dump of various CAIENF control blocks and data areas for use in problem diagnosis. The dump is written to a dynamically allocated SYSOUT data set according to the specifications in the SYSOUT control statement. The dump can be taken to a DASD or tape data set by including a DD statement named ENFDUMP in the CAIENF started task JCL.

ENDIF()

Closes a set of control options started by an IF statement. The control options within the IF/ENDIF pair are executed only when the conditions in the IF statement are true.

ENFCT (component_name,#buffers,buffer_size,member_name)

The ENFCT statement controls the Component Trace environment. For more information, see CAIENF Component Trace (see page 219).

Options are:

component_name

The Component Trace name. Default is CAIENF

#buffers

Number of trace buffers. Specify a number between 2 and 128 inclusive. Default is 2.

buffer_size

The size, in K, of each buffer. Specify a number between 64 and 8192 inclusive (buffer sizes will be between 64k and 8M). Default is 512.

member_name

Name of the parmlib member for Component Trace to process when the Component Trace is initialized. There is no default (no trace sets will be enabled). A sample member CTEENF00 is provided for tailoring.

EVENT(event_name,option)

Controls processing for a specific event.

- If (event_name,options) is not entered, then the command presents a summary of all the events currently active.
- If just (event_name) is entered, then a detailed description of the event is presented.

Only one option can be specified on a single EVENT statement. If more than one option is required, specify multiple EVENT statements using different option operands on each. Valid options that can be entered on an EVENT statement are:

INACT

Deactivates the event.

ACT

Enables an INACTIVE event.

DUMP

Writes an SVC dump the next time the event occurs.

REC

Requests that data from this event be recorded on the database.

NOREC

Suppresses recording of data for this event on the database.

RP=nn

The number of days the event is to be retained on the database. This keyword can be used to override the DETAIL setting for a particular event. The value can be from 1 to 99.

PURGE=Y|N

PURGE=Y sets recorded events to PURGE only when their retention period has been met. PURGE=N sets recorded events to archive and then be purged after their retention period has been met. The PURGE flag is used during CAIENF archive processing. For more information, see the ARCHIVE control option.

The default is PURGE=N, however the default can be overridden by a DCM event definition. Base CAIENF events that are defined by DCMs distributed with CAIENF such as CAS9DCM0, are all set with a default of PURGE=N.

Note: When upgrading from CA Common Services r11, a utility can be used to create the new EVENT control option statements required for CA Common Services r12. For more information, go to Special DCM and EVENT Utility (see page 171).

IF(condition & condition ...)

Controls CAIENF command processing based on system variables. If the condition is true, then all control options following the IF command are executed. Execution stops when an ENDIF command is encountered. If the condition is false, the set of commands between the IF and ENDIF commands is purged and a message is sent to the console.

Note: Do not use blanks between variables and operators. Any blank encountered will be deciphered as the end of the IF statement.

Options are:

Option	Description
condition	A condition consists of a variable and a value joined by an operator. For example, variable=value.
variable	Replace *variable* with one of the following items: **SYSPLEX** an IBM-defined variable **SYSNAME** an IBM-defined variable **SYSCLONE** an IBM-defined variable **CCISYSID** a CAIENF startup option **SMFID** the System Management Facility ID
operator	One of the following operators is required. = Equal **¬=** Not Equal **!=** Not Equal **<=** Less Than or Equal **>=** Greater Than or Equal **>** Greater Than < Less Than.
value	Replace *value* with the appropriate value for the variable you have selected. Values are in EBCIDIC and must be enclosed in single quotes; if the quotes are omitted, *value* is treated **first** as a variable and **then** as a value. Valid values for each variable are listed in the following table.
SYSPLEX	The name of your sysplex. The length of the name can be up to eight bytes of character data; if not specified, the default LOCAL is used. The name is defined in either member COUPLExx or LOADxx in SYS1.PARMLIB.

Option	Description
SYSNAME	The name of your system. The length of the name can be up to eight bytes of character data; if not specified, the processor ID is used as the default. The name is defined in either the IEASYMxx or IEASYSxx sysname operand. **Note:** If the same SYSNAME appears more than once in the IF command, you must specify SYSCLONE for each duplicate SYSNAME.
SYSCLONE	An abbreviation for the name of the system, up to two bytes of character data in length. If not specified, defaults to the last two characters of SYSNAME. SYSCLONE is defined in the IEASYMxx member of SYS1.PARMLIB.
CCISYSID	The value of the CAICCI system ID for this system. The value length can be up to eight bytes of character data. CCISYSID is defined in member ENFPARMS of SYS1.PARMLIB **Note:** CCISYSID must be specified before using this variable in ENFPARMS.
SMFID	The SMF system ID. The value length can be up to four bytes of character data. If not specified in SMPFRMxx, it defaults to the four-digit processor model number. SMFID is defined in the SID operand of member SMFPRMxx in SYS1.PARMLIB.
User defined	The name of a variable defined by you that can be up to eight bytes of character data in length. User defined variables are defined in member IEASYMxx of SYS1.PARMLIB. **Note:** See the IBM *INIT and Tuning Reference* for instructions on how to define these symbolics. When using symbolics you define in your ENFPARMS, **do not** code the "&". and ending ".".

The following is an example showing an IBM Static System Symbolic definition. In this example, the variable is specified as RELEASE, not &RELEASE.

```
SYMDEF(RELEASE='520')
```

logical operator (& |)

A logical operator joins each set of conditions. Use one of these logical operators:

&

Signifies AND logic. Both conditions must be met for the statement to be true.

|

Signifies OR logic. Either condition can be met for the statement to be true.

The following is an example showing AND logic:

```
SYSPLEX='PLEX01'&SYSNAME='HP92'
```

In this example, the system named HP92 must be found on the PLEX01 sysplex for this condition to be true.

Examples

The following examples show how to specify the IF and ENDIF commands:

```
IF(SYSPLEX='PLEX01'&SYSNAME='HP92')
  SELECT(JOBPURGE,JOBNUM,EQ,J*)
ENDIF()
IF(SYSPLEX=PLEX01'&SYSNAME='HP94'|SYSNAME='HP97')
  SELECT(STEPTERM,JOBNUM,EQ,J*)
  ENDIF()
IF(SYSPLEX='PLEX01'&SYSNAME='HP97')
  SCREEN(DSCLOSE,ACCESS,EQ,INPUT)
  ENDIF()
IF(SYSNAME='HP91'|SYSNAME='HP92'|SYSNAME='HP94')
  SCREEN(JOBTERM,JOBNAME,EQ,JRDR)
  ENDIF()
IF(SYSNAME='HP91'|SYSNAME='HP92'|SYSNAME='HP94')
  PROTOCOL(TCPIP)
  ENDIF()
```

LABEL(option)

Controls dynamically allocated backup data sets. If a tape device is selected for dynamic allocation as an archive data set, this control option allows you to specify standard label (SL) or non-labeled (NL); SL is the default.

MAP

Produces the location of key CAIENF modules and control blocks for debugging purposes.

MAXQTIME(nn)

Specifies the time limit, in minutes, that an application with events queued to it is allowed to listen for the events before the application is disabled if disable mode is turned on, or when an SVCDUMP will be taken in warning mode. The minimum value is 0 and the maximum is 999; the default is 10; a value of 0 disables the time limit.

Note: A value that is too low may result in CAIENF applications with a low dispatching priority being erroneously disabled. A value that is too high may result in high virtual storage utilization.

MSGEVENT(options)

The MSGEVENT control option is used to define the messages that CAIENF is to intercept and generate JOBFAIL events for.

If (action,prefix,type,desc,namecol,nbrcol) is not entered, then the command presents a summary of all installation defined messages that CAIENF is to intercept and generate JOBFAIL events for.

The control option syntax and description are as follows:

```
MSGEVENT(action,prefix,type,desc,namecol,nbrcol)
```

Defines a message that CAIENF is to intercept and generate JOBFAIL events for. A MSGEVENT statement allows you to create JOBFAIL events for messages issued by your installation exits. For example, a JES2 JCL converter internal text scan exit.

The MSGEVENT statement may be repeated once for each message (up to a maximum of 16 messages) that CAIENF is to intercept and generate JOBFAIL events for.

Options are:

action

DEFINE to define a JOBFAIL message or DELETE to delete a previously defined JOBFAIL message.

prefix

The message prefix. Must be at least 5 characters long and can be up to 16 characters long.

The first character must be alphabetic or numeric. The remaining characters must be alphabetic, numeric, or one of the special characters ($, #, @).

Must not be one of the default message prefixes that CAIENF always generates JOBFAIL events for (for example, IEF251I).

This option is required for all actions.

type

The JOBFAIL event type. Can be up to 8 characters long. This is a user defined type; there are no predefined values.

The first character must be alphabetic or numeric. The remaining characters must be alphabetic, numeric, or one of the special characters ($, #, @).

This option is required for the DEFINE action and is not allowed for the DELETE action.

desc

The JOBFAIL event description. Can be up to 40 characters long. If the description contains spaces or any special characters, the entire string must be enclosed in single quotes.

This option is required for the DEFINE action and is not allowed for the DELETE action.

namecol

The starting column number of the job name in the intercepted message. The value specified must be a decimal value from 0 to 128.

If the value specified is zero, the originating job name is used. Otherwise, the job name is set to the 8 bytes of message text beginning at the column specified.

Note that the originating job name may be spaces if the WTO is issued in SRB dispatching mode.

This option is required for the DEFINE action and is not allowed for the DELETE action.

nbrcol

The starting column number of the job number in the intercepted message. The value specified must be a decimal value from 0 to 128.

If the value specified is zero, the originating job number is used. Otherwise, the job number is set to the 8 bytes of message text beginning at the column specified.

Note that the originating job number may be spaces if the WTO is issued in SRB dispatching mode.

This option is required for the DEFINE action and is not allowed for the DELETE action.

JOBFAIL Messages

HASP105

HASP107

HASP119

HASP310

HASP396

HASP543

HASP608

HASP890

HASP913

IAT2628

IAT4801

IAT6815

ICH408I

IEF251I

IEF450I

IEF452I

IEF453I

IEF722I

IEFC452I

TSS7100E

TSS7145E

NODB

NODB bypasses initialization and preparation of the CAIENF event recording database (such as CA Datacom/AD). Use this option when the CAIENF database is not installed or is not to be used. If CAIENF is operating without a database, events are processed real-time and CAIENF event logging does not occur. Furthermore, CAIENF event management services such as checkpoint and archive/retrieval, are not available without a database.

NODB is a startup-only option. It cannot be issued as a command once CAIENF is started.

Do not confuse NODB with RECORD(NO). Even though RECORD(NO) disables the logging of events to the database, RECORD has no effect on database availability or operability. If NODB is omitted and RECORD(NO) is specified as a startup option, CAIENF still performs database initialization and preparation. This allows a RECORD(YES) command to be issued at a more suitable time.

Important! Specifying NODB along with startup option RECORD(YES) is a conflict that results in the posting of message CAS9216E and CAIENF startup termination.

PURGE(event)

Deletes **all** data for the specified event. Index structures are also deleted and rebuilt.

Note: Data purged using this command can not be recovered.

RECORD(option)

Globally controls CAIENF event logging.

- If RECORD(NO) is specified, no database logging is attempted for any event (although CAIENF applications can still process events real-time).
- RECORD(YES), the default, allows logging to take place according to the options specified on the EVENT and SELECT control options.

REFRESH(module)

Dynamically reloads global CAIENF modules at any time after initialization. This allows new copies of modules to be installed without shutting down CAIENF (after applying maintenance, for example).

You can issue a REFRESH control option with strings of up to seven modules, using commas as delimiters. For example,

```
REFRESH (CAS9ERR,CAS9EVNT,CAS9LCAM,CAS9MSG,CAS9MSGT,CAS9SRVC)
```

Important! You should REFRESH only those modules for which you are directed to do so. Instructions for refreshing modules comes from either CA Technologies Support or by a cover letter accompanying a PTF tape.

For information on refreshing modules from a data set other than the CAIENF STEPLIB or LNKLIST concatenation, see the TASKLIB control option.

REINIT

Allows all CAIENF global modules to be refreshed at startup time.

Note: The REFRESH command can be specified to reload a specific module without needing to shutdown CAIENF. The REINIT command can only be specified with the CAIENF Start command (S ENF,,,REINIT). This command cannot be specified as a parameter in the CAIENF parameter file.

RESETSTA

Causes most of the CAIENF statistic counters and response-time measurement areas to be cleared. The following list shows counters and measurement areas that are cleared by RESETSTA:

- database response rate
- event response rate
- number of database calls
- number of errors and abends
- number of operator commands executed

RETPD(nnnn)

Controls dynamically allocated backup data sets. Specifies the retention period for the data set; the default is 30. Note that dynamically allocated archive data sets are kept track of within the CAIENF database; if RETPD(0) is specified, CAIENF does not know when a data set is scratched. This may cause unpredictable results when attempting to restore selected data.

SCREEN(event_name,[NONE] | [data_name,operand,mask,grp_id])

Controls CAIENF event creation for a specific event by screening out (excluding) events with certain data field values. This control option should be used with care, since all CAIENF listening applications will no longer receive the excluded events. The SCREEN command can be useful when presented with certain special performance problem circumstances and is often used based on the advice of CA Technologies Support.

Options are:

event_name

Can be any defined event.

NONE

(Optional) Used to remove existing SCREEN settings. After executing this option, new settings can be issued.

data_name

(Optional) Must be a data name associated with the specified event.

operand

(Optional) Can be EQ, NE, GT, NG, LT or NL

mask

(Optional) A masking pattern that can include the * (match any length string).

grp_id

(Optional) A value used to group commands with AND logic. There is no limit to the number of commands that can be grouped, and there is no limit to the number of groups that can be designated. However, grouping is done separately for each event type. That is, you cannot group different types of event statements in a single group.

Examples

```
SCREEN(JOBTERM,ACCOUNT,NE,P*)
```

Screens out JOBTERM events where the ACCT field does not start with P (for instance, screen out jobs which do not have a production account number).

```
SCREEN(DSCLOSE,JOBNUM,EQ,TSU*)
```

Screens out DSCLOSE events where the JOBNUM field starts with TSU (that is, screens out data set activity for TSO users).

```
SCREEN(DSCLOSE,VOLUME,EQ,MVS*)
SCREEN(DSCLOSE,IOCOUNT,GT,1000)
```

Screens out DSCLOSE events where the VOLSER starts with MVS, or greater than 1000 I/O operations are done.

```
SCREEN(DSCLOSE,NONE)
```

All settings to screen out events for DSCLOSE are removed. You can then enter new SCREEN settings.

```
SCREEN(DSCLOSE,VOLUME,EQ,VSE*,GP0)
SCREEN(DSCLOSE,IOCOUNT,GT,500,GP0)
SCREEN(DSCLOSE,VOLUME,EQ,MVS*,GP1)
SCREEN(DSCLOSE,IOCOUNT,GT,1000,GP1)
SCREEN(STEPTERM,JOBNAME,EQ,PRD*,GP1)
SCREEN(STEPTERM,CLASS,EQ,M,GP1)
SCREEN(STEPTERM,STEPNAME,EQ,FTP@@@$)
```

The above statements screen out DSCLOSE events where the VOLSER starts with VSE **and** greater than 500 I/O operations are done **or** where the VOLSER starts with MVS **and** greater than 1000 I/O operations are done.

The statements also screen out STEPTERM events for jobs that have names that start with the characters PRD **and** have a jobclass of M **or** where the stepname is FTP@@@$.

As is evidenced from these examples, multiple SCREEN statements for the same event can be specified. The event is screened out if the event data matches any of the set criteria.

SELECT(event_name,data_name,operand,mask,[grp_id])

Controls CAIENF event logging (recording on the CAIENF database) for a specific event. Options are listed and described as follows:

event_name

Can be any defined event.

data_name

Must be a data name associated with the specified event.

operand

Can be EQ, NE, GT, NG, LT, or NL.

mask

A masking pattern that can include the * (match any length string) and ? (match single character.)

grp_id

(Optional) A value used to group commands with AND logic. There is no limit to the number of commands that can be grouped, and there is no limit to the number of groups that can be designated. However, grouping is done separately for each event type. That is, you cannot group different types of event statements in a single group.

Examples

Command examples are as follows:

```
SELECT(JOBTERM,ACCOUNT,EQ,P*)
```

Logs only JOBTERM events where the ACCT field starts with P (for instance, logs only jobs with production account numbers).

```
SELECT(DSCLOSE,JOBNUM,NE,TSU*)
```

Logs only DSCLOSE events where the JOBNUM field does not start with TSO (for instance, does not log data set activity for TSO users).

```
SELECT(DSCLOSE,VOLUME,NE,MVS*)
SELECT(DSCLOSE,IOCOUNT,GT,1000)
SELECT(DSCLOSE,DSN,EQ,SYS1.*)
```

Logs DSCLOSE events where the VOLSER does not start with MVS, or greater than 1000 I/O operations are done, or the data set name starts with “SYS1.”.

```
SELECT(DSCLOSE,VOLUME,EQ,VSE*,GP0)
SELECT(DSCLOSE,IOCOUNT,GT,500,GP0)
SELECT(DSCLOSE,VOLUME,EQ,MVS*,GP1)
SELECT(DSCLOSE,IOCOUNT,GT,1000,GP1)
SELECT(STEPTERM,JOBNAME,EQ,PRD*,GP1)
SELECT(STEPTERM,CLASS,EQ,M,GP1)
SELECT(STEPTERM,STEPNAME,EQ,FTP@@@$)
```

The above statements cause logging of DSCLOSE events where the VOLSER starts with VSE **and** greater than 500 I/O operations are done **or** where the VOLSER starts with MVS **and** greater than 1000 I/O operations are done.

The statements also cause logging of STEPTERM events for jobs that have names that start with the characters PRD **and** have a jobclass of M **or** where the stepname is FTP@@@$.

As evidenced by the examples above, multiple SELECT statements for the same event can be specified. It should be noted that the event is logged if the event data matches any of the above criteria.

SETCCP([restart_max,][pause_time])

Sets the parameters for managing CAICCI spawned services. SETCCP can be issued at any time.

restart_max

(Optional) Sets the restart limit. This limit prevents CAICCI from repeatedly restarting failed services. The default limit is 3.

Once a service has reached its restart maximum, it will no longer be automatically restarted by CAIENF/CAICCI, but can be reenabled using the STARTCCP command.

pause_time

(Optional) Sets the time to pause after a z/OS STOP before a z/OS CANCEL is issued. The default is 10 seconds.

At CAIENF termination, the CAICCI spawn service (CASNMVSR) checks to see if an active service is to be terminated using the z/OS STOP command (the preferred method), or dropped using the z/OS CANCEL command (designated as the "default" method to maintain compatibility with previous CAIENF implementations).

If CASNMVSR issues a z/OS STOP command against the service's started task, then pauses for pause_time seconds. If the task is still active after the pause, a z/OS CANCEL command is issued to force removal of the service from the system.

Note: pause_time is a system-wide specification that applies to ALL services spawned by CAICCI. Each service however, can be configured with its own individual pause time by using SPNPARMS or the CASNESVC module that preempts the default or system-wide setting.

Example:

SETCCP(5,20)

Sets the system-wide restart limit to 5 and sets the system-wide pause-time to 20 seconds.

SPACE(nnn,nnn)

Controls dynamically allocated backup data sets; the default is 0,0. If DASD is used as the archive media, specifies a primary and secondary space allocation amount in cylinders.

Note: CAIENF always allocates DASD data sets with the RLSE operand so any unused space is freed.

STARTCCP(started_task_name | *,[FORCE])

Manually re-enables started tasks that have been placed out-of-service due to problems with the task that eventually caused it to exceed the restart maximum limit.

STARTCCP can be issued only after CAIENF is fully initialized (it cannot be specified in ENFPARMS).

started_task_name

A required parameter that specifies either:

- The 1-8 character started-task name associated with the CAICCI service to be re-enabled.
- An asterisk, indicating that started tasks for all out-of-service services are to be re-enabled.

FORCE

(Optional) Re-enables a dormant service even if it is **not** in an out-of-service state.

STARTCCP provides two distinct advantages over simply restarting a service using a console-issued z/OS START command:

- PARM information, which may be quite lengthy, is retained from the initial starting of the service and is automatically re-inserted by STARTCCP.
- Multiple services may be re-enabled with a single STARTCCP command.

STARTCCP is provided only as an emergency tool to re-enable faulty started tasks that have been placed out-of-service due to exceeding the restart_max limit.

A started task re-enabled with STARTCCP is **not** fully managed by CAICCI and will **not** be automatically restarted if it terminates unexpectedly. STARTCCP started tasks do however undergo proper disposal at CAIENF shutdown.

Examples:

STARTCCP(CCITCP)

STARTCCP(CCITCP,FORCE)

STARTCCP(*)

STATUS(opt)

Provides a quick summary of processing since CAIENF was started. Also provides version, gen level, and service level of every installed CA Common Services for z/OS.

Note: If no option is defined, the command provides a summary of processing since CAIENF was started. It also provides version, gen level, and service level of every installed CA Common Services for z/OS.

CCIR

Displays all CAICCI resources and their status.

CCIL

Displays all CAICCI links and their status.

Status information is provided as console messages, including the following:

CAS9250I CAIENF STATUS DISPLAY FOR SYSTEM XXXX

CAS9075I SERVICE(name) VERS(version level) GENLVL(genlevel) SERVICE LEVEL(service level)

CAS9251I INITIALIZED AT XXXXXXXX ON XXXXXXXX

CAS9252I INTERCEPTS(NNN) COLLECTORS(NNNN) APPLS(NNN) CSA(NNNK)

CAS9253I EVENTS(NNNNNNNN) DB I/O(NNNNNNNN) ABENDS(NNN) COMMANDS(NNNN)

CAS9254I EVENT RESP(NNNN) DB RESP(NNNN) SUPP(NNNNNNNN) LOST(NNNNNN)

CAS9255I DIAGNOSE(XXXXXXXXXX) STATUS(XXXXXXXX) XXXXXXXX

CAS9256I SYSOUT(X) DESTINATION(XXXXXXXX TRACE(NNNN) EXIT(XXX)

CAS9257I RECOVERY(XXXXXXXX) RECOVERY QUEUE(NNNNNN)

CAS9267I CICS STATUS: (AUTO-INSTALL FEATURE XXXXXXXX)

CAS9268I XXXXXXXX(XXXXXXXX) - APPL(XXXXXXXX) ASCB(NNNNNNNN) XXXXXXXX

CAS9269I NNN CICS ADDRESS SPACES MONITORED

Note: As part of the STATUS command output, message CAS952I displays the amount of common storage currently used by CAIENF, not the amount ESCA, if any.

Information on these messages can be found in the *CA Common Services for z/OS Message Reference Guide*.

SVCDUMP

Takes a machine readable dump to one of the system dump data sets using the MVS SDUMP service. Dumps taken using this command are titled ENF SVCDUMP COMMAND. A system dump data set (SYS1.DUMPxx) must be available prior to issuing this command.

SYNCDUMP(YES/NO)

This parameter can be included in your CAIENF parameters. It can also be issued on the console using a MODIFY command, after CAIENF has initialized. Default is No.

Note: CA will direct clients when necessary to include this parameter.

SYSOUT(class,dest)

Controls the SYSOUT class and destination used when CAIENF allocates a SYSOUT data set for dump or message processing; the default is A.

If a pre-allocated DASD or tape data set is desired, an ENFDUMP DD statement can be included in the CAIENF started task JCL. This file is then used to hold dumps and other information generated by CAIENF.

TASKLIB(dsn{,opt_vol}|CLOSE)

When issued, this control option indicates which library to use to refresh all CAIENF global modules. This allows you to apply maintenance on CAIENF without having to recycle or perform an IPL.

Using this control option, you can choose which library to REFRESH after maintenance is applied.

A primary benefit of this control option is that you can test the maintained modules for data center conflicts before placing maintenance changes in the CAIENF STEPLIB or system LNKLIST concatenation. However, if you use the benefits of this control option to test maintenance, you should CLOSE the specified alternate DSN and REFRESH the default libraries once you are confident that there are no systems conflicts.

Valid options for this control option are:

dsn

The library from which you want to REFRESH all CAIENF modules.

opt_vol

If the specified library is not cataloged, you may specify an optional volume where the desired library is located.

CLOSE

Turns off the TASKLIB specifications. Upon issuing the TASKLIB(CLOSE) control option, further refreshing is performed on the CAIENF STEPLIB or system LNKLIST concatenation.

TIMER(min,max)

Controls the threshold values for time-driven database updating. The database control subtask waits between updates for a duration somewhere between the minimum and maximum values, depending on overall system load. Values entered are in 1/100-second increments; permissible range for this command is from 1 (for a value of .01 second) to 9999 (for a value of 99.99 second). The default is 500,6000.

Note: A value that is too low causes extra overhead in the database control subtask. A maximum value that is too large may result in high virtual storage utilization.

TRACE(nnnn[,loc])

Controls the size of the CAIENF internal trace table. nnnn is a one- to four-digit number specifying the size of the trace table in 1024-byte segments. If loc is not specified, then the trace table is allocated from storage within the private area of the CAIENF address space. If desired, certain global events can be traced by specifying loc as CSA, which places the trace table in CSA.

Note: The trace table is printed as part of the ENF DUMP command output.

Example

```
TRACE(786,CSA)
```

UNIT(option)

Specifies the unit type to be used for a dynamic allocation request to backup a data set. Any valid unit name can be entered and CAIENF honors it for dynamic allocation requests. The default is TAPE.

Options are as follows:

UNIT(TAPE)

Backup data set is allocated to unit device type TAPE.

UNIT(3390)

Backup data set is allocated to unit device type 3390.

UNIT(SYSDA)

Backup data set is allocated to SYSDA.

View a List of Event Names

Some control options require an event name as an operand. You can view a list of the event names currently defined to the CAIENF database.

To view a list of event names

Issue an EVENT command from any z/OS system console:

```
F ENF,EVENT
```

The CAIENF command processor supports abbreviations of any command in the form of its shortest unique prefix. For instance, the STATUS command can be entered as STATUS, STAT, or STA. However, STATUS cannot be entered as S or ST, since S and ST are not unique identifiers.

CAIENF/CICS Control Options

CAIENF/CICS control options are specified at CAIENF startup time. They can be dynamically altered using the CAIENF operator command. Any z/OS system console or operator command facility can be used to enter CAIENF/CICS commands.

Note: Each CAIENF/CICS control option, like all CAIENF control options, must be coded on a separate line in the CAIENF parameter file.

Control Options Summary - CAIENF/CICS

The following is a list of the CAIENF/CICS control options.

Control Option	Default
CICS(START|STOP,jobname,prodcode)	N/A
CICS(REFRESH,jobname)	N/A
MODE(CICS,ON|NONE)	ON
CICSREL(*nn,nn,nn...*)	N/A
CICS(SETSDUMP,jobname,prodcode)	N/A
CICS(SETCDUMP,jobname,prodcode)	N/A
CICS(NOSDUMP,jobname,prodcode)	N/A
CICS(NOCDUMP,jobname,prodcode)	N/A
CICS(NOESTAE,jobname)	N/A
CICS(QUERY,jobname,prodcode)	N/A

Note: CICSREL is valid only at CAIENF startup time.

CICS(START|STOP,jobname,prodcode)

This control option enables or disables the CAIENF/CICS processing for a CA Technologies product within a specific CICS region; however, you must not issue this command until the targeted CICS region is active. CICS is active when message DFHSI1517 is displayed.

Important!

The START|STOP control option cannot be specified in the CAIENF parameter file. It must be issued as an operator command after CAIENF is active.

START/STOP cannot be specified in the CAIENF auto commands file.

You cannot use the START|STOP commands against the CICS components of the CA Top Secret and CA ACF2 security products. CAIENF can be enabled in the region by specifying SEC=YES as a SIT option.

Options are as follows:

START

Enables the CAIENF/CICS interface.

STOP

Disables the CAIENF/CICS interface.

jobname

The jobname of the specific CICS region you want CAIENF/CICS installed in.

prodcode

The product code for the specific product you want to install in a CAIENF/CICS region.

Note: jobname refers to the started task JOBNAME if CICS is running as a started task and refers to the batch JOBNAME if CICS is running as a batch job.

Examples

To activate the CAIENF/CICS intercepts for CA JARS (assuming the product code for CA JARS is UX62) where the CICS jobname is CICS630A, issue the following command:

```
CICS(START,CICS630A,UX62)
```

To disable the CAIENF/CICS intercepts where the CICS region started task or jobname is CICS630A, issue the following command:

```
CICS(STOP,CICS630A)
```

CICS(REFRESH,jobname)

This control option can be used if the CICS routines were loaded into each CICS Private Area. If used, this control option refreshes the CAIENF/CICS interface for a particular job. CAIENF automatically determines the correct CICS version for the specified jobname and reloads the required intercepts.

The following option applies:

jobname

Specify this option if you want CAIENF/CICS intercepts refreshed for a specific job or CICS region.

Example

The following example refreshes the CAIENF/CICS modules in region CICS630A.

```
ENF CICS(REFRESH,CICS630A)
```

MODE(CICS,ON|NONE)

Controls whether CAIENF/CICS automatically installs intercepts into each CICS region. The default is ON.

Options are:

ON

Allows CAIENF/CICS to automatically install intercepts at CICS systems start.

NONE

Prevents CAIENF/CICS from automatically installing intercepts into active CICS or as CICS regions start.

Note: If MODE(CICS,NONE) is specified, you need to manually enter a CICS(START,jobname,prodcode) for each CICS region in which CA Technologies solutions operate.

If MODE(CICS,NONE) is issued while CAIENF is active, then only the regions activated after the issuance of this command are not intercepted by CAIENF/CICS. CICS regions that were active prior to the issuance of this command continue to be intercepted. If you wish to deactivate a CICS region at this point, the CICS region must be recycled.

Examples

To activate the CAIENF/CICS intercepts in all CICS regions, code the following:

```
MODE(CICS,ON)
```

To prevent the CAIENF/CICS intercepts from being installed into all active CICS regions during CAIENF startup, code the following:

```
MODE(CICS,NONE)
```

CICSREL(nn,nn,nn,...)

Specifies the releases of CICS for which CAIENF/CICS modules are to be loaded into ECSA. Issuing this command without an option causes all of the CAIENF/CICS Intercept modules to be loaded into ECSA.

Options are:

nn,nn,nn,...

Specify one or more CICS or CICS Transaction Server (CTS) releases by replacing *nn* with one of the following values.

For Release	Specify this for *nn*:
CTS 3.1	64
CTS 3.2	65
CTS 4.1	66
CTS 4.2	67

Note: Specify this control option in the ENFPARMS file. Specified modules are then loaded into ECSA at CAIENF startup and reside there until the next IPL. More than one instance of this command can be issued for specifying the intercept modules for more than one CICS release. This command can accept up to seven release level values.

CICS(SETCDUMP,jobname,prodcode)

This control option causes CAIENF/CICS to write a standard CICS transaction dump when the specified jobname or specified product in the CICS region abends. If you do not issue this control option, and the specified jobname or product abends, CAIENF/CICS recovers without writing a transaction dump.

Options are:

jobname

Specify this option if you want CAIENF/CICS to write a transaction dump when a specific jobname abends.

prodcode

Specify this option if you want CAIENF/CICS to write a transaction dump when a specific product abends.

Example

The following example causes the CAIENF/CICS to write a transaction dump when either job CICS0 or product J161 abends.

```
ENF CICS(SETCDUMP,CICS0,J161)
```

Note: This command produces transaction dumps for all errors that occur after the command has been issued. Unlike the SETSDUMP command, this command does not have to be reissued to produce more than one dump.

CICS(SETSDUMP,jobname,prodcode)

This control option can be used to cause CAIENF/CICS to write system dump (SVC dump) when the specified jobname or specified product in the CICS region abends. If you do not issue this control option, and the specified jobname or product abends, CAIENF/CICS recovers without writing an SVC dump.

Options are:

jobname

Specify this option if you want CAIENF/CICS to write an SVC dump when a specific jobname abends.

prodcode

Specify this option if you want CAIENF/CICS to write an SVC dump when a specific product abends.

Example

Issuing the following command causes CAIENF/CICS to write an SVC dump when either job CICS0 or product J161 abends.

```
ENF CICS(SETSDUMP,CICS0,J161)
```

Note: This command produces the SDUMP only once after the command has been issued. If a second abend should occur, it will be treated as if the command was never issued.

If a dump is needed, a second SETSDUMP command should be issued after an SDUMP has been produced, from the first SETSDUMP command issued.

CICS(NOSDUMP,jobname,prodcode)

This control option can be used to cancel the SVC dump request for a specified jobname and product made by issuing the CICS(SETSDUMP,jobname,prodcode) command.

Options are:

jobname

Specify this option if you want to cancel the request for CAIENF/CICS to write an SVC dump when a specific jobname abends.

prodcode

Specify this option if you want to cancel the request for CAIENF/CICS to write an SVC dump when a specific product abends.

Example

The following example causes CAIENF/CICS to cancel the request to write an SVC dump when either job CICS0 or product J161 abends.

```
ENF CICS(NOSDUMP,CICS0,J161)
```

CICS(NOCDUMP,jobname,prodcode)

This control option can be used to cancel the standard CICS transaction dump request for a specified jobname and product made by issuing the CICS(SETCDUMP,jobname,prodcode) command.

Options are:

jobname

Specify this option if you want to cancel the request for CAIENF/CICS to write a transaction dump when a specific jobname abends.

prodcode

Specify this option if you want to cancel the request for CAIENF/CICS to write a transaction dump when a specific product abends.

Example

The following example causes CAIENF/CICS to cancel the request to write a transaction dump when either job CICS0 or product J161 abends.

```
ENF CICS(NOCDUMP,CICS0,J161)
```

CICS(NOESTAE,jobname)

Issuing this command cancels all CICS recovery when a specified jobname or CICS region abends.

Important! The CICS(NOESTAE,jobname) command should only be used under the direction of CA Technologies Support.

CICS(QUERY,jobname,prodcode)

This control option can be used to query the status of the CAIENF/CICS environment to help diagnose a problem associated with installing a CA Technologies solution in a CICS region. The basic command with no options indicates whether CAIENF/CICS is active, whether the AUTOINSTALL feature has been set, which DCMs have been installed in the CAIENF database, and whether the initialization modules associated with those DCMs have been loaded. Alternatively, a particular region and solution can be specified.

Options are:

jobname

Specify this option to inquire about the status of the CICS region represented by the eight-character jobname.

prodcode

Specify this option to inquire about the status of the product represented by the four-character product code.

Examples

Issue the CICS(QUERY) command without any options to display global information about the status of CAIENF/CICS as it appears from the CAIENF address space.

```
ENF CICS(QUERY)
```

The response will show if CAIENF/CICS is active, whether the AUTOINSTALL feature is active, depending on how the CAIENF parameter MODE(CICS,ON|NONE) was set, and also the current status of the DCMs defined to the CAIENF database for CA Technologies solutions, if any. This information includes for each solution the product code, the name of the associated initialization routine, the address where the initialization routine is loaded, and the status (active or inactive) of the solution.

Note: If CAIENF was unable to locate the routine during initialization, the address shown is 00000000. Check the LINKLST or CAIENF STEPLIB for the routine.

To inquire about the status of a specific product with product code K050 being installed in a specific region, CICSV410, issue the following command. The response indicates whether the product is installed in the region and if so, if it has been activated.

```
ENF CICS(QUERY,CICSV410,K050)
```

CAIENF/CICS SPAWN Control Options

CAIENF/CICS SPAWN control options are specified at CAIENF startup time. They may be dynamically altered using the CAIENF operator command. Any z/OS system console or operator command facility can be used to enter CAIENF/CICS SPAWN commands.

Note: Each CAIENF/CICS SPAWN control option, like all CAIENF control options, must be coded on a separate line in the CAIENF parameter file.

Control Option Summary - CAIENF/CICS SPAWN

The following is a list of the CAIENF/CICS SPAWN control options.

Control Option	Default
CICS(START,jobname,SPWN)	N/A
CICSPAWN(*nn,nn,nn*)*	N/A
MODE(CICSPAWN,NONE|ON)	NONE

***** This command is only valid at CAIENF startup time.

CICS(START,jobname,SPWN)

Enables or disables the CAIENF/CICS SPAWN processing within a specific CICS region; however, you must not issue this command until the targeted CICS region is active. CICS is active when message DFHSI1517 is displayed.

Important!

The START|STOP control option cannot be specified in the CAIENF parameter file. It must be issued as an operator command after CAIENF is active.

START/STOP cannot be specified in the CAIENF auto commands file.

Options are:

START

Enables the CAIENF/CICS SPAWN interface.

Jobname

The jobname of the specific CICS region you want CAIENF/CICS SPAWN installed in.

SPWN

The code to be installed in a CAIENF/CICS SPAWN region.

Example

To activate CAIENF/CICS SPAWN in a region where the CICS jobname is CICS630A, issue the following command:

```
CICS(START,CICS630A,SPWN)
```

CICSPAWN(nn,nn,nn,...)

Specifies the releases of CICS for which CAIENF/CICS SPAWN modules are to be loaded into ECSA. Issuing this command without an option causes all of the CAIENF/CICS SPAWN intercept modules to be loaded into ECSA. This command can accept up to seven release level values.

Options are:

nn,nn,nn,...

Specify one or more CICS or CICS Transaction Server release levels for which SPAWN will be activated. You can specify up to six levels in a list separated by commas.

For Release:	**Specify this for *nn*:**
CTS 3.1	64
CTS 3.2	65
CTS 4.1	66

Note: Specify this control option in the ENFPARMS file. Specified modules are then loaded into ECSA at CAIENF startup and will reside there until the next IPL. More than one instance of this command may be issued for specifying the intercept modules for more than one CICS release.

If CICSPAWN is not specified in the ENFPARMS file, none of the CAIENF/CICS SPAWN intercept modules will be loaded or deleted.

MODE(CICSPAWN,NONE|ON)

Controls whether CAIENF/CICS SPAWN automatically installs into each CICS region. The default is NONE.

Options are:

NONE

Prevents CAIENF/CICS SPAWN from automatically installing intercepts into active CICS or as CICS regions start.

ON

Allows CAIENF/CICS SPAWN to automatically install intercepts as CICS systems start.

Note: If MODE(CICSPAWN,NONE) is specified, you need to manually enter a CICS(START,jobname,SPWN) for each CICS region in which CA Technologies solutions operate. This command can accept up to seven release level values.

If MODE(CICSPAWN,ON) is issued while CAIENF is active, then only the regions activated after the issuance of this command are installed by CAIENF/CICS SPAWN. CICS regions that were active before this command was issued are not intercepted. If you wish to activate SPAWN for a CICS region at this point, the CICS region must be recycled.

Example

To activate the CAIENF/CICS SPAWN intercepts in all CICS regions, code the following:

```
MODE(CICSPAWN,ON)
```

To prevent the CAIENF/CICS SPAWN intercepts from being installed into all active CICS regions during CAIENF startup, code the following:

```
MODE(CICSPAWN,NONE)
```

CAIENF/DB2 Control Options

CAIENF/DB2 control options are specified at CAIENF startup time. They can be dynamically altered using the CAIENF operator command. Any z/OS system console or operator command facility can be used to enter CAIENF/DB2 commands.

Control Options Summary - CAIENF/DB2

The following is a list of the CAIENF/DB2 control options.

Control Option	Description
DB2(MAXTHRD,nnnn)	Set the thread table size.
DB2REL(nn)*	Load the CAIENF/DB2 Intercept module into ECSA.
MODE(DB2,ON|NONE)	Set automatic intercept installation.

***** This command is valid only at CAIENF startup time.

DB2(MAXTHRD,nnnn)

Allows you to control the thread table size for CAIENF/DB2

nnnn

Specifies the number of entries in the thread table.

Default: 2100

Max: 9999

DB2REL(nn)

Loads the CAIENF/DB2 module for a specific CAIENF/DB2 release into ECSA. Issuing this command without an option loads all of the CAIENF/DB2 Intercept modules into ECSA.

nn

Specify 23 to load any supported DB2 release.

Note: Specify this control option in the ENFPARMS file. Specified modules are then loaded into ECSA at CAIENF startup. More than one instance of this command can be issued for specifying the intercept modules for more than one DB2 Release.

MODE(DB2,ON|NONE)

Controls whether CAIENF/DB2 automatically installs intercepts into each DB2 subsystem. The default is ON.

ON

Allows CAIENF/DB2 to automatically install intercepts as DB2 systems start.

NONE

Prevents CAIENF/DB2 from automatically installing intercepts as DB2 subsystems start.

Note: If MODE(DB2,NONE) is issued while CAIENF is active, then only the subsystems activated after the issuance of this command are not intercepted by CAIENF/DB2. DB2 subsystems that were active prior to the issuance of this command continue to be intercepted. If you wish to deactivate a DB2 subsystem at this point, the DB2 subsystem must be recycled.

Example

To activate the CAIENF/DB2 intercepts in all DB2 subsystems, code the following:

```
MODE(DB2,ON)
```

To prevent the CAIENF/DB2 intercepts from being installed into starting DB2 subsystems after CAIENF startup, code the following:

```
MODE(DB2,NONE)
```

Imbedded Multi-User Facility (IMUF) Option

One of the major benefits of CAIENF r12 and later is the switch from an older relational database technology to CA Datacom/AD for optional event recording and possible future event playback. Previous versions of CAIENF used a database technology based on CA-Universe to record CAIENF event activity. CA Datacom/AD provides many benefits over CA-Universe technology.

CA-Universe operated as an internal subtask to the CAIENF main task, and all CA-Universe database activity was performed "under the covers" completely within the CAIENF address space.

CA-Datacom/AD, through its Multi-User Facility (MUF), is generally run as a batch job or started task in its own address space. User communication with a MUF is provided through the CA Datacom/DB Application Programming Interface (API). The CAIENF address space uses this API to manage its event recording database and under typical CAIENF operation, this requires the target MUF (the one CAIENF will be communicating with) to have been manually started and active in a separate address space before CAIENF is started.

Requiring CAIENF users to start up another address space to service the CAIENF database may be viewed as an imposition since additional started task JCL must be maintained and security settings planned for.

To address these concerns, CAIENF is designed with an optional feature that approximates the same "under the covers" database operation as previously implemented with CA-Universe.

This feature is called Imbedded MUF (IMUF) and it removes the necessity of starting up a separate MUF address space to support CAIENF's database management requirements.

IMUF Implementation and Startup

An IMUF is configured and customized exactly as it would have been configured and customized for standard (external) MUF operation as described in the *CA Datacom/AD Installation and Maintenance Guide*.

IMUF is not "MUFLite". The same rules and regulations that apply to external MUF operation, apply to IMUF operation as well.

IMUF can be MUF-plexed or shared with other CA solution address spaces without any impact on CAIENF operation, though there is likely to be little, if any, benefit on behalf of CAIENF by doing so.

In a mutli-MUF environment, IMUF must be uniquely identified through the CA-Datacom/AD DBSIDPR element like any other MUF.

All of the job steps and DD statements that normally would have been included in the JCL for the external MUF started task procedure, must be included in the CAIENF started task procedure for IMUF operation.

A small amount of "handshaking" logic has been added to both CAIENF and CA-Datacom/AD to support IMUF operation, but essentially, the IMUF is unaware that it is running as an CAIENF subtask.

There are only two discernable differences between standard MUF operation and IMUF operation:

- In IMUF mode, the MUF is attached by CAIENF, within its own borders, rather than by a z/OS initiator.
- Since CAIENF is in control of the address space, the CAIENF command processor will have already seized control of the address space operator communication "handle" by the time IMUF is launched. This means the IMUF will be unable to directly listen for, and respond to, operator modify (F) or stop (P) commands intended for the IMUF task. However, CAIENF and the IMUF task negotiate a "handshake" during startup that, in addition to synchronizing CAIENF/IMUF startup and shutdown procedures, allows CAIENF to forward MUF-specific operator commands to IMUF through the CAIENF command processor.

  For instructions on how to use the IMUFCMD, see The CAIENF Imbedded Multi-User Facility Command (see page 217).

To activate the IMUF feature, place a single "IMUF" statement anywhere in the ENFPARMS PDS member or data set of the CAIENF started task procedure. This directs CAIENF to load the appropriate support routines and initialize the IMUF environment. Datacom/MUF startup/status messages will be posted to the CAIENF job log along with CAIENF startup/status messages.

Partial example illustrating placement of the IMUF statement:

```
//ENFPARMS   DD  *
……
DCM(CAS9DCM0)
DCM(CAS9DCM2)
……
IMUF
RECORD(YES)
……
/*
```

Other Considerations

CAIENF must run as an APF task because z/OS security restrictions won't allow an APF-authorized task to invoke an unauthorized task. Users have the choice of running an external MUF as APF-authorized or non-authorized, but IMUF must run APF-authorized. When the MUF is running in its own address space, it is under no CAIENF-imposed APF restrictions. It is only in CAIENF IMUF mode that a MUF requires APF authorization.

The MUF load libraries (CAAXLOAD and CUSLIB) that participate in IMUF operation must be APF-authorized because they will now be commingled through JOBLIB/STEPLIB or LNKLST with CAIENF APF-authorized libraries.

Note: The CA Datacom/AD auxiliary data sets (CXX, FXX, LXX, PXX, and RXX) do not require APF authorization.

Running the IMUF APF-authorized should not be an issue since it is strongly recommended that MUFs always be APF-authorized to maximize CA Datacom/AD's performance enhancement capabilities, and to insure that the full range of CA Datacom/AD's features are available.

If the IMUF statement is omitted from ENFPARMS, CAIENF will start up in standard mode of operation where it expects (and requires) an external MUF address space to be already started and prepared to accept CAIENF database requests.

The CAIENF Imbedded Multi-User Facility Command

If the imbedded Multi-User Facility (IMUF) is being used, you can issue CA Datacom/AD MUF console commands to the IMUF task by using the CAIENF IMUFCMD command. The format of the IMUFCMD command is:

```
IMUFCMD('muf_command_string')
```

where muf_command_string is 1 to 40 characters comprising a MUF command that is routed to the IMUF task.

For information on MUF console command structure and usage, see the appropriate CA Datacom/AD guide.

If the MUF command contains one or more command parameters, *muf_command_string* **must** be framed within single quotes. Improper framing will cause a malformed command string to be sent to the IMUF task.

Examples:

```
IMUFCMD(Status)

IMUFCMD('Display DBMUFPR A0')
```

In the first example, no framing characters are necessary because the MUF STATUS command requires no parameters.

IMUFCMD command responses are posted to the CAIENF job log.

The IMUFCMD command can be abbreviated to IMUFC.

CAIENF SNMP Monitor Control Options

The TRACE and DEBUG facilities provide additional diagnostic information. In general, the TRACE facility traces the general flow and the DEBUG facility provides additional information about the functions it performs.

Control Options Summary - ENFSNMP

The following is a list of the CAIENF SNMP Control Options.

Control Option	Description
TRAce	ON|OFF. Turns the tracing on or off.
DEbug	ON|OFF. Turns the debugging on or off.

Notes:

- Only the capitalized letters are required to invoke the function. For example, 'F enfsnmpm,TRA OFF' turns off tracing
- All ENFSNMPM traces or debugging messages are routed to CAW1LOG DDname. ENFSNMPM CAW0PROC includes this DDNAME.

Chapter 7: CAIENF Component Trace

This section contains the following topics:

CAIENF Component Trace Details

Refer to the IBM manual, *z/OS MVS System Commands* (located under TRACE CT) for information on starting and stopping a Component Trace External Writer and to connect or disconnect the writer from a component trace component. A sample PROC, ENFXWTR, is provided in CAI.CAW0PROC, which invokes the IBM external writer ITTTRCWR. Refer to the IBM manual *Diagnostic Tools and Service Aids* for editing. The PROC must be available for the MVS operator command TRACE CT. You may need to place PROC in SYS1.PROCLIB or IEFJOBS or IEFPDSI).

Component Trace provides for automatically generating a TRACE CT command when component tracing is initialized. This is performed via a PARMLIB member that can be specified on the IBM macro to initialize component tracing. A sample member, CTEENF00, is shipped in CAI.CAW0OPTN. If this member is used, it must be moved to an MVS PARMLIB data set and its name specified on the ENFCT auto command. As delivered if ENFCT is added as an CAIENF parameter, then CTEENF00 will be used and as delivered CTEENF00 will tell component trace to start the ENFXWTR started task. The ENFXWTR task must be customized to use a particular default output CTRACE data set.

Command Format

In addition to using the ENFCT CAIENF parameter to start the component trace, you may also start the trace manually. You can also start a trace when a previous trace was stopped. Use the z/OS TRACE CT command for this.

This command has the following syntax:

```
TRACE CT,ON,COMP=component
```

The following code generates a WTOR:

```
R nn,[,OPTIONS=()][,ASID=()] [,JOBNAME=()] [,WTR=writer|DISCONNECT]
```

The Start/Stop routine accepts the following as options:

```
PAUSE
RESUME
ENAble|ENAble(trcid) option 1
DISable|DISable(trcid) option 1
MODify(trcid)  option 1
LIST|LIST(trcid)
STATUS
```

Option 1:

```
EIDG(eidg, eidg, ...eidg)
XEIDG(eidg, eidg, ...eidg)
XAJ
LIMIT(max)
```

Where trcid is the trace set id and eidg is the event ID group.

PAUSE and RESUME

PAUSE will internally pause tracing. RESUME will resume tracing. This allows TRACE CT commands to be issued to update trace sets while not internally collecting trace data.

ENABLE and DISABLE

If a trace set ID is specified and if it exists, it is updated and the status changed to enabled or disabled. If the trace set ID does not exist, a new trace set is created and its status is set to enabled or disabled. If the trace set ID is not specified, a trace set ID is generated, a new trace set created, and its status set to enabled or disabled. Specifying ASID= or JOBNAME= will rebuild the filtering criteria.

Modify

The trace set ID must exist. The specified trace set is updated. Specifying ASID= or JOBNAME= will rebuild the filtering criteria.

LIST

If the trace set ID is not specified, all defined trace sets are displayed.

STATUS

Displays the status as to whether tracing is active or inactive, paused or not paused, and the events being traced.

EIDG (eidg, eidg, ...eidg)

For ENABLE, DISABLE, and MODIFY, this will designate the Event ID Groups (for instance, E1 and P1 are groups) to be traced.

XEIDG(eidg, eidg,...eidg)

For MODIFY, this will designate the Event ID Groups to be removed from the list of Event ID Groups to be traced.

XAJ

For ENABLE, DISABLE and MODIFY this indicates that the ASID= and JOBNAME= specifications are excluded ASIDs and JOBNAMEs. Specifying XAJ without ASID= or JOBNAME= resets any filtering based on address space.

LIMIT(max)

For ENABLE, DISABLE, and MODIFY, this will indicate the maximum number of times the data will be collected for that trace set. A value of zero indicates no limit is in effect (can undo a previous specification). The maximum value is 16,000,000. By default, no limit is in effect.

This command syntax can also be used to start the trace manually if you wish to use the options in the CTEENF00 parms member:

```
TRACE CT,ON,COMP=CAIENF,PARM=CTEENF00
```

Event ID Groups

The following table lists the Event ID groups:

ID	Event ID Group
E1	Event Creation for events other than STEPTERM and DSCLOSE
E2	Event Creation for the STEPTERM event
E3	Event Creation for the DSCLOSE event
P1	Event dispatch, delivery and freeing for events other than STEPTERM and DSCLOSE
P2	Event dispatch, delivery and freeing for the STEPTERM event.
P3	Event dispatch, delivery and freeing for the DSCLOSE event.
L1	Init or term request, abnormal termination of a listener request, DATA request and request for checkpointing
C1	Checkpoint processing and checkpoint reload.

To stop an active trace that is using the ENFXWTR task, issue these z/OS commands:

```
TRACE CT,OFF,COMP=CAIENF
TRACE CT,WTRSTOP=ENFXWTR
```

Chapter 8: CAIENF Utilities

This section describes a utility provided with CA Common Services for z/OS.

ENFC is a command utility which is invoked from a REXX Program to perform the CAIENF LISTEN function, returns data back in CLIST or REXX variables.

This section contains the following topics:

ENFC

This section describes the TSO CLIST Application Interface (ENFC). It provides details on the ENFC LISTEN function, including command syntax and descriptions, as well as standard CLIST and compound REXX variables.

The ENFC command is used in a TSO CLIST to perform the LISTEN function and get the results back in CLIST variables. ENFC supports both the old standard CLIST language and the REXX language.

Note: The LISTEN function runs on MVS 3.1.3 and above.

ENFC LISTEN Function

The LISTEN function allows a CLIST to pause until one or more system events occur.

```
ENFC LISTEN EVENT(event1[,event2,...,event9])
                        [CONT(YES|NO)]
                        [FROM(yydddhhmmssth)]
```

EVENT(event1, ...)

Specifies the event types (from one to nine) for which you wish to wait. These events must be defined in the CAIENF database.

CONT(YES|)NO)

Indicates the action to be taken after control returns to the CLIST.

YES

Keeps the LISTEN active after control returns to the CLIST. Note that if you specify YES, you must issue an ENFC TERM command when you finish LISTENing so that event data is not unnecessarily queued up.

NO

Starts a new LISTEN every time ENFC is called. However, any events that occurred between the time of the first and second LISTENs are not returned to you.

FROM

Specifies a recovery date and time for events that have been logged. Beginning at this date and time, CAIENF returns the logged events for which you have been listening.

LISTEN Processing

Control is returned to the CLIST only after one of the events in the event list has occurred or after an error is encountered. When control is given back, the following CLIST variables are set:

ENFRC

The return code from the LISTEN function. If the value is zero, the function completed successfully. If it is nonzero, the variable ENFMSG contains an error message.

ENFDRC

The detail return code from the LISTEN function. This contains further information about the error if an error was indicated in ENFRC.

ENFMSG

This is a character string that contains a message indicating the completion status of the LISTEN request. This variable should always have a value, even when the LISTEN completes successfully.

ENFEVENT

This variable is set to the name of the event that occurred.

ENFDNUM

The number of data elements returned.

ENFDNAME

A character string containing all the names of the returned data elements, eight bytes each. For example, if three data elements, such as JOBNAME, JOBNUM, and COMPCODE, are returned, then:

&ENFDNUM. = 3, and

&ENFDNAME. = JOBNAME JOBNUM COMPCODE.

The &ENFDNAME. variable may then be used with the &SUBSTR. function to extract the names of the data elements.

dataname

The value of each data element is assigned to a variable with the same name as the data element. For example, if the three data elements JOBNAME, JOBNUM, and COMPCODE were returned, the variable

&JOBNAME. could equal CICSPROD,

&JOBNUM. could equal STC00100, and

&COMPCODE. could be 602.

Compound REXX Variables

ENFC also sets a compound REXX variable with the values of the data elements that were returned. The following variables are set:

ENFDATA.enfevent.0

The number of data elements returned for the event. The value for *enfevent* that was set previously is substituted in the variable name. Thus, if the event that occurred was JOBTERM, you could also use the name ENFDATA.JOBTERM.0.

ENFDATA.enfevent.n

An array of names of the data elements that were returned, where *n* can range from 1 to the number returned in ENFDATA.enfevent.0.

ENFDATA.enfevent.dataname

The value of the data element that was returned. For example, if a JOBTERM event occurred and returned the three data elements:

- JOBNAME = CICSPROD,
- JOBNUM = STC00100, and
- COMPCODE = 602,

The following variables are set:

Variable	Value
ENFDATA.JOBTERM.0	3
ENFDATA.JOBTERM.1	JOBNAME
ENFDATA.JOBTERM.2	JOBNUM
ENFDATA.JOBTERM.3	COMPCODE
ENFDATA.JOBTERM.JOBNAME	CICSPROD
ENFDATA.JOBTERM.JOBNUM	STC00100
ENFDATA.JOBTERM.COMPCODE	602

Also note that for REXX users, all the variables that are set for use by the old CLIST language are also set for REXX users. Thus, REXX variables JOBTERM, JOBNUM, and COMPCODE as well as ENFDNUM and ENFDNAME are set.

Note: If you press Attention (PA1) while a LISTEN is active, ENFC performs its own ENFC TERM function and returns the following to the CLIST:

- ENFRC = 8
- ENFDRC = FF
- ENFMSG = Attention received

CAS9DCMR - DCM Mapper

This is a stand-alone utility program that is invoked through standard z/OS JCL, as follows:

Note: Sample JCL is located in .CAW0JCL(CAS9DCMR).

```
//jobname  JOB  job card information
//STEP1   EXEC  PGM=CAS9DCMR,REGION=0M,
//                    PARM='DCM(dcm1,dcm2,…,dcmn)'
//*
//STEPLIB DD DSN=common-services-CAW0LOAD,DISP=SHR
//SYSPRINT  DD  SYSOUT=*
//SYSPUNCH DD  DSN=&&DROPSQL,
//                      DISP=(NEW,PASS),
//                      UNIT=SYSDA,
//                      SPACE=(TRK,(5,1)),
//                      DCB=(LRECL=80,BLKSIZE=800,RECFM=FB)
//DCMLIB     DD  DSN=dcm-library-dataset-name,DISP=SHR
//SYSIN       DD  *
utility control statements
//
```

The only utility control statement allowed is the DCM statement which identifies which DCMLIB members are to be mapped:

```
DCM(dcm1,dcm2,…,dcmn)
```

There can be more than one DCM control statement.

Any statement that contains an asterisk (*) in column 1 is treated as a comment. It is printed on the input statement report, but the information in the card is ignored.

The DCM control statements can be provided from the following different sources:

1. If only one control statement is needed, the SYSIN DD statement can be omitted, and the required DCM control statement can be passed from the PARM field on the EXEC card.
2. Multiple control statements can be provided by the SYSIN DD statement.
3. Control statements can be passed in the PARM field and from the SYSIN DD statement. In this case, the statement passed in the PARM field is processed first, and then all the ones contained in the SYSIN data stream.

The output from the program is divided into four parts:

1. A listing of all the control statements processed and any error messages associated with them or the DCM members referenced
2. A map of the Event Tables defined in the DCM.
3. A map of all the fields contained in each of the Event Tables.
4. A series of comments and DROP TABLE SQL commands that can be passed to the DBSQLPR utility provided with CA Datacom/AD. These statements are written to the SYSPUNCH DD statement if it is present. If the SYSPUNCH statement is omitted from the run, no error message is produced nor are the DROP TABLE statements generated.

   Note: DROP TABLE SQL commands should only be used to REPLACE a DCM, as advised in the *Best Practices Guide*.

   Note: If the CAIENF DCM (CAS9DCM0) is chosen to report on, no DROP TABLE SQL statements will be output for this particular DCM. This is because some of the CAS9DCM0 event tables are required for CAIENF to function properly. If CAS9DCM0 ever needs to be replaced, the replacing code from either a PTF or a new CCS release, will provide specific instructions and sample JCL to perform the replacement.

Chapter 9: CA-GSS Statements and Commands

This section contains the following topics:

Notation Conventions

This section explains the general notation conventions used in syntax for the CA-GSS services. Conventions specific to a service are explained in the section discussing that service.

These general conventions are used in syntax:

- When parameters are optional, they are enclosed in brackets:

  ```
  [parameter]
  ```

- When a parameter is required but may take one of several predefined values, your choices are enclosed in braces:

  ```
  {value1|value2}
  ```

- Multiple choices sometimes appear on one line and are separated by a vertical bar (as shown above). When your choices are more complex, they appear stacked on top of one another.
- When the entire command syntax will not fit on one line, the syntax is continued on a second line that is indented two spaces. Likewise, if the command syntax carries over onto a third line, the third line of syntax is indented an additional two spaces, and so on.

Example

Assume there is a batch maintenance command called GO, and this is its syntax diagram:

```
GO BY {CAR|TRAIN} {TO WORK AT time}             [WITH name]
                  {OUT FOR DINNER TO restaurant}
                  {OVER THE BRIDGE}
 [BECAUSE reason] BEFORE DOING action [AND THEN PHONE HOME]
                                      [OR JUST STAY WHERE YOU ARE]
```

This syntax diagram indicates the following things about the GO command:

- You must specify the BY parameter, and you must give it a value of either CAR or TRAIN.
- You must specify one of the following: TO WORK AT *time*, OUT FOR DINNER TO *restaurant*, or OVER THE BRIDGE.
- You may include the WITH parameter and a value for *name*.
- You may include the BECAUSE parameter and a value for *reason*.
- You must include the BEFORE DOING parameter and a value for *action*.
- You may include one of the following: AND THEN PHONE HOME or OR JUST STAY WHERE YOU ARE.

Service Routines

This section explains each service routine for CA-GSS.

$SRV_COMMAND

The $SRV_COMMAND routine issues the passed argument as an ISERVE operator command and returns the results on the stack.

Syntax

Use this syntax:

```
CALL $SRV_COMMAND argument
```

Parameters

The $SRV_COMMAND service routine takes this parameter:

argument

Command to be issued.

Returned Data

Data is returned on the stack. The format is identical to that displayed on the operator console. Error messages are also reported on the stack, as in a response to an operator command.

Return String

The return string is null. Errors are reported on the stack.

$SRV_DSTAB_STAT

The $SRV_DSTAB_STAT routine returns information about all ISETs and their associated data sets.

Syntax

Use this syntax:

```
CALL $SRV_DSTAB_STAT
```

Parameters

No operands are expected and none should be passed.

Returned Data

Data is returned on the stack. Each line represents one ISET. The first character in each record is the delimiter character. This character is used to separate the returned fields as listed next.

1. Delimiter character.
2. ISET name.
3. Data set name.
4. Version of editor or compiler that last updated the ISET. This field will be blank unless ISERVE allocated and opened the ISET since initialization.
5. CPU time expended by IMOD tasks whose root IMOD was loaded from this ISET. The format of this field is *hh:mm:ss.tho*, with leading zeros and field separators removed.
6. ISET description.
7. Access status for the ISET (read or write).
8. Subsystem ID that this ISET is associated with, if available.

Note: All fields may not be completed when this service routine returns.

Example

```
limit = call srv_dstab_stat
   do i = 1 to limit
   parse pull 1 d +1 . (d) iset (d) dsn (d) version (d) cpu (d) .
   end i
```

$SRV_ENQ_INFO

The $SRV_ENQ_INFO routine returns information about all IMOD enqueues that are currently held.

Syntax

Use this syntax:

```
CALL $SRV_ENQ_INFO
```

Parameters

No operands are expected and none should be passed.

Returned Data

Data is returned on the stack. Each line represents one enqueue for one IMOD. The first character in each record is the delimiter character. This character is used to separate the returned fields as listed below:

1. LOCK or ENQUEUE.
2. QNAME or LOCK-NAME. This field is always eight characters, padded with blanks. This field may contain embedded delimiter characters.
3. RNAME. This field is always 256 characters, padded with blanks. This field may contain embedded delimiter characters.
4. OWN or WAIT.
5. Reserved.
6. IMOD ID associated with the request.

Return String

The return string is null.

Usage Note

Since both the QNAME and RNAME fields may contain embedded delimiters, care should be exercised in parsing the returned string. See the following Example for a recommended method.

Example

```
call $srv_enq_info
limit = queued( )
 do i = 1 to limit
   parse pull 1 d +1 type (d) . +1 qname +8 . +1 rname +256   ,
         own (d) level (d) . (d) imodid (d) .
   end i
```

$SRV_LOG_STAT

The $SRV_ILOG_STAT routine returns information about ILOG files.

Syntax

Use this syntax:

```
CALL $SRV_ILOG_STAT
```

Parameters

No operands are expected and none should be passed.

Returned Data

Data is returned on the stack. Each line represents one ILOG subfile. The first character in each record is the delimiter character. This character is used to separate the returned fields, as listed below:

1. ILOG file number.
2. ILOG subfile number.
3. DDname associated with the ILOG subfile
4. Status of the file:

   Not-Avail

   Subfile is not available for use.

   Dump_req'd

   Subfile must be dumped and/or reset before it is available.

   Full-Avail

   Subfile is full but has already been reset.

Active

Subfile is currently in use.

Empty

Subfile is empty and available for use.

Not-Open

Subfile has not been opened or has been closed.

5. Total records currently in the subfile.
6. Percent of subfile that is currently being used.
7. Date stamp of the first record in the subfile. The format is *yy/mm/dd*.
8. Time stamp of the first record in the subfile. The format is *hh:mm:ss*.
9. Date stamp of the last record in the subfile. The format is *yy/mm/dd*.
10. Time stamp of the last record in the subfile. The format is *hh:mm:ss*.

Return String

This routine returns the number of records added to the stack.

Example

```
call $srv_ilog_stat
limit = result
do i = 1 to result
   parse pull 1 d +1 ilog (d) subfile (d) ddname   ,
         (d) status (d) records (d) percent   ,
         (d) sdate (d) stime (d) edate (d) etime (d)
   end i
```

$SRV_IMOD_INFO

The $SRV_IMOD_INFO routine returns information about a specific IMOD.

Syntax

Use this syntax:

```
CALL $SRV_IMOD_INFO imod
```

Parameters

The $SRV_IMOD_INFO service routine takes this parameter:

imod

The name of the IMOD to be reported on.

Returned Data

Data is returned in the returned string. The first character in the string is the delimiter character. This character is used to separate the returned fields as listed below:

1. IMOD name.
2. Reserved.
3. Reserved.
4. Use count. This is the number of IMOD tasks currently executing this copy of the IMOD.
5. User ID of the TSO user or the console ID from which this IMOD was last loaded into the system.
6. Date of last modification. The format is *yy/dd/mm*.

   Note: Simply recompiling an IMOD does not constitute a modification.
7. Compiler version last used.
8. Reserved.
9. ISET name from which the IMOD was last loaded.
10. Data set name from which the IMOD was last loaded.
11. *Internal* if the IMOD is an integrated part of ISERVE; null otherwise.
12. *Callable* if the IMOD can be called as a subroutine; null otherwise.
13. *Active* if the IMOD is currently executable; null otherwise.
14. Total CPU time expended by this IMOD since it was last loaded.

Example

```
call $srv_imod_info 'test_user'

parse var result 1 d +1 name (d) . (d) . (d) count (d) id (d)  ,

       cdate (d) version (d) . (d) iset (d) dsname (d) int (d)  ,

       callable (d) act (d) cputime (d) .
```

$SRV_IMOD_LIST

The $SRV_IMOD_LIST routine returns a list of all IMODs currently loaded.

Syntax

Use this syntax:

```
CALL $SRV_IMOD_LIST
```

Parameters

No operands are expected and none should be passed.

Returned Data

Data is returned on the stack, one IMOD name per record. The IMODs are in collating sequence.

Return String

A null string is returned by this routine.

$SRV_JOB_INFO

The $SRV_JOB_INFO routine searches active system routines to find one or more jobs that meet the specified name criteria. Information on the matched jobs is returned on the stack.

Syntax

Use this syntax:

```
CALL $SRV_JOB_INFO {jobname|*}
```

Parameters

The $SRV_JOB_INFO service routine takes these parameters:

jobname

The name of a job (JOB), started task (STC), or time-sharing user (TSU).

***

All jobs, started tasks, and time-sharing users are selected.

Returned Data

Data is returned on the stack, one address space per record. The first character in each record is the delimiter character. This character is used to separate the returned fields, as listed below:

1. ASID number, returned as a decimal number.
2. ASCB address, returned as a decimal number.
3. Jobname.
4. JOB, STC, or TSU.
5. Task ID, for started task only.
6. Current STEP name.

Return String

This routine returns the number of lines of data that it added to the stack. A returned value of 0 may be interpreted as "job not found".

$SRV_MVS_DATA

The $SRV_MVS_DATA routine returns information about the currently executing z/OS.

Syntax

Use this syntax:

```
CALL $SRV_READ_IMOD name iset
```

Parameters

The $SRV_MVS_DATA service routine takes these parameters:

LPA

Returns information pertaining to data sets in LPALIB.

LINKLIST

Returns information pertaining to LINKLIST data sets.

CATALOG

Returns information pertaining to the catalogs defined on your system.

Returned Data

The following data is returned on the stack:

1. SMF ID.
2. Time of the last IPL in *hh:mm:ss* format.
3. Date of the last IPL in *mm/dd/yy* format.
4. CPU ID.
5. IPL volume.
6. IPL unit address.
7. Master catalog name.
8. z/OS product version.
9. z/OS FMID.
10. CPU model number.
11. IPL nucleus member.
12. IOCONFIG.

$SRV_IDR_LIST

The $SRV_IDR_LIST routine reads a load module and returns IDR information on the stack.

Syntax

Use this syntax:

```
CALL $SRV_IDR_LIST ddname module
```

Parameters

The $SRV_IDR_LIST service routine takes these parameters:

ddname

DDname of the library that contains the load module.

module

Name of the load module to be listed.

Returned Data

If the request was successful, RC is set to 0 and the IDR data is returned on the stack, one IDR record per stack record. All fields are blank delimited. The first field is IDR. The second field indicates the type of IDR and is one of the following:

LINK

Link Editor IDR.

ZAP

IDR assigned by AMASPZAP utility.

USER

IDR included as the result of a link editor IDENTIFY statement.

TRANSLATOR

IDR created by the assembler or compiler.

The remaining fields depend upon the type of IDR and are listed in the following table.

IDR Type	Field	Description
LINK	Date	Date of the last link edit in the form *mm/dd/yy*.
	Version	Linkage editor version user ID.
	ID	ID assigned (by IBM) to the linkage editor.

IDR Type	Field	Description
ZAP	Date	Date the zap was applied in the form *mm/dd/yy*.
	CSECT	CSECT that was zapped.
	Data	Eight-byte data field specified at the time of the zap. If no data was supplied, the string NO_IDENT is returned.
USER	Date	Date that the IDR was inserted, in the form *mm/dd/yy*.
	CSECT	CSECT to which the data applies.
	Data	Data from the IDENTIFY statement. This data may contain blanks.
TRANSLATOR	CSECT	Name of the CSECT.
	Length	Length of the CSECT.
	Date	Date of translation in the form *mm/dd/yy*.
	Version	Version of the translator.
	Name	Name of the translator. This is a numeric sequence for IBM translators.
	Date	Date of pre-compilation (if pre-compiled).
	Version	Version of the pre-compiler (if any).
	Name	Name of the pre-compiler (if any).

Return Codes

The RC special variable will contain one of the following return codes:

Code	Meaning
0	The request was successfully processed.
121	The library could not be opened.
122	The load module could not be found.

$SRV_PDS_LIST

The $SRV_PDS_LIST routine returns a listing of all members contained in a PDS. If requested, the user-data field is returned as ISPF statistics.

Syntax

Use this syntax:

```
CALL $SRV_PDS_LIST ddname [ISPF]
```

Parameters

The $SRV_PDS_LIST service routine takes these parameters:

ddname

DDname to allow access to the PDS.

ISPF

If specified, the returned user-data field is formatted as ISPF statistics.

Returned Data

Successful completion is indicated by the RC special variable being set to 0 and a null string returned as the result. Errors are indicated by a non-zero value in RC and an error message returned as the result.

Each PDS directory entry is returned on the stack. The first character of each record is a delimiter used to parse the individual fields. The order of the fields returned is as follows:

1. Member name.
2. ALIAS or TRUENAME, depending upon the entry.
3. TTR of start of member (printable hexadecimal).
4. Number of TTRs in the user data (whole number, meaningful for load libraries only).
5. User data.

If ISPF was specified as an option and the user-data appears to be valid ISPF statistics, the user data field is converted to the following blank-delimited fields:

1. The version and modification level specified as *vv.mm*. The *vv* and *mm* values are whole numbers in the range of 0 through 255.
2. The creation date for this member. The format is *mm/dd/yy*.
3. The date of last modification. The format is *mm/dd/yy*.
4. The time of last modification. The format is *hh:mm* or *hh:mm:ss* where *hh* ranges from 0 through 24, *mm* ranges from 0 through 59, and *ss* ranges from 0 through 59. The *hh*, *mm*, and *ss* values are whole numbers.

   Note: ISPF began storing change seconds with Release 3.3. Prior releases only stored hours and minutes.
5. The current size in lines (logical records) of the member. This is a whole number in the range of 0 through 65535.
6. The size in lines (logical records) of the member when it was created. This is a whole number in the range of 0 through 65535.
7. The number of times this member has been modified. This is a whole number in the range of 0 through 65535.
8. The user ID of the user who last updated (or created) this member. This is a seven-character field (the eighth character, if any, is truncated), right-padded with blanks.

$SRV_READ_IMOD

The $SRV_READ_IMOD routine returns the source code for an IMOD on the stack.

Syntax

Use this syntax:

```
CALL $SRV_READ_IMOD name iset
```

Parameters

The $SRV_READ_IMOD service routine takes these parameters.

name

Name of the desired IMOD.

iset

Name of the ISET that contains the IMOD. This ISET must be defined to the ISERVE address space and the ISET itself must be accessible under the user ID of the caller.

Usage Notes

- The source code is always loaded from the ISET, regardless of whether or not the source already resides in memory.
- The source code is returned regardless of its status (for instance, embedded compiler errors, non-production status, and so on).
- Compiler error messages are stripped out and not returned.

Returned Data

Successful completion is indicated by the RC special variable being set to 0 and a null string returned as the result. Errors are indicated by a non-zero value in RC and an error message returned as the result.

The source text is returned in the current stack, one line per record.

$SRV_SYS_STAT

The $SRV_SYS_STAT routine returns ISERVE status information.

Syntax

Use this syntax:

```
CALL $SRV_SYS_STAT
```

Parameters

No operands are expected and none should be passed.

Returned Data

1. CA-GSS subsystem name.
2. JES NJE node name.
3. SMF ID.
4. GRS system name.
5. CA-GSS job name.
6. CA-GSS started task ID.
7. Local system GoalNet node.
8. Local system GoalNet LU name.
9. Elapsed time in *hh:mm:ss* format.
10. Amount of CPU time this task has used.
11. Number of IMODs available.
12. Number of IMODs executed.
13. Number of REXX EXECs executed.
14. Number of instructions executed.

$SRV_SYS_STATX

Like the $SRV_SYS_STAT routine, the $SRV_SYS_STATX routine returns ISERVE status information. The data returned by the $SRV_SYS_STATX routine is more readily accessible by REXX programs, but it is not printable.

Syntax

Use this syntax:

```
CALL $SRV_SYS_STATX
```

Parameters

No operands are expected and none should be passed.

Returned Data

The following data is returned on the stack:

1. CA-GSS subsystem name.
2. JES NJE node name.
3. SMF ID.
4. GRS system name.
5. CA-GSS job name.
6. CA-GSS started task ID.
7. Local system GoalNet node.
8. Local system GoalNet LU name.
9. Elapsed time in *hh:mm:ss* format.
10. Amount of CPU time this task has used.
11. Number of IMODs available.
12. Number of IMODs executed.
13. Number of REXX EXECs executed.
14. Number of instructions executed.

$SRV_TPCF

The $SRV_TPCF routine interfaces with the CA MIA Tape Sharing Preferencing Control Facility and returns information about one or more tape drives. Information on the selected devices is returned on the stack.

Syntax

Use this syntax:

```
CALL $SRV_TPCF {loaddr-hiaddr} [,{loaddr-hiaddr}]
               {addr}          [,{addr}]
```

Parameters

The $SRV_TPCF service routine takes these parameters:

addr

The address of a tape device. The address is in printable hexadecimal format; for example, 8C0.

loaddr

The beginning, low address of a range of tape devices. The address is in printable hexadecimal format; for example, 5D0.

hiaddr

The ending, high address of a range of tape devices. The address is in printable hexadecimal format; for example, 5DF.

Usage Notes

- Use of this routine requires the presence of the TPCF ADDRESS environment. This is distributed with CA-Multi-image Manager and requires a definition to CA-GSS in the initialization parameters.
- Multiple individual addresses or multiple ranges of addresses can be specified by separating them with commas.
- The entire address specification is a single argument and should not contain embedded blanks.

The user data field, returned as a stack-record field, may contain embedded delimiter characters. Note how the parse statement is used in the example.

Returned Data

Data is returned on the stack, one device per record. The first character in each record is the delimiter character. This character is used to separate the returned fields, as listed below:

1. Unit name (the device address) as a printable hexadecimal value.

2. Volume serial number, if any.
3. Preference.
4. O if overgenned, N if not under TPCF control, else a null field.
5. Device type and features.
6. Status of device on this system.
7. If reserved for a job, the JOBNAME.
8. Owning system name, if allocated.
9. Owning JOBNAME, if allocated.
10. User information string. This field is always eight characters long and can contain binary data. It can also include one or more delimiter characters as data.
11. TPCF return code.
12. Reserved, null.
13. Reserved, null.
14. Reserved, null.
15. Reserved, null.
16. Name of System 1.
17. Flags for System 1.
18. 18 - 47: Name and flag fields for Systems 2 through 16.

Return String

This routine returns a blank string to indicate successful completion. Otherwise, error text is returned.

Note: Even if error text is returned, some data may be added to the stack.

Example

A complete example is contained in the EXAMPLE ISET under the name TPCF.

```
call $srv_tpcf '5c0-5cf'
do i = 1 to queued()
   parse pull 1 d +1 unit (d) volser (d) pref (d) stat1   ,
               (d) devt (d) stat2 (d) resjob (d) sysname   ,
               (d) jobname (d) user +8 . +1 retcode   ,
               (d) flags (d) . (d) . (d) rest
 /*  additional processing  */
   end i
```

Special Purpose IMODs

This section lists and explains each special purpose IMOD for CA-GSS.

$USER_ILOG_FULL

The $USER_ILOG_FULL IMOD is invoked as a subroutine call from $INT_ILOG_FULL whenever an ILOG subfile is filled. Upon return, the ILOG subfile is placed in RESET status and is eligible for reuse, as needed.

Arguments

Two arguments are passed to the IMOD - the ILOG file number, and the subfile number within the ILOG. These values are numeric and are separated by blanks in standard REXX form.

Return Values

$USER_ILOG_FULL can return a text string to the caller. The value of this string determines the final disposition of the ILOG subfile as shown in the following table.

Value	Explanation
RESET	The ILOG subfile is reset and is reused when needed.
null	The ILOG subfile is not reset. You need to reset the subfile manually before CA-GSS can re-use it. If any text string other than RESET is returned, a null string is assumed.

Usage Notes

- This IMOD is also entered once for each ILOG subfile that is found to be full during CA-GSS initialization processing.
- When you are logging large volumes of data, you need to provide a $USER_ILOG_FULL IMOD so that CA-GSS can automatically switch or reset ILOGs that become full. If you do not provide the IMOD, CA-GSS automatically resets the full ILOG.

Example

A complete example of $USER_ILOG_FULL can be found in the EXAMPLE ISET.

INITIALIZATION

The INITIALIZATION IMOD is used during CA-GSS initialization as its own IMOD task. Other processing continues simultaneously.

Arguments

No arguments are passed to this IMOD.

Initial Stack Records

Prior to invoking the INITIALIZATION IMOD, CA-GSS processes the CAW0OPTN data set. The text from each USER statement in this data set is passed, using the stack, to the INITIALIZATION IMOD.

Return Values

If any results are returned by this IMOD, they are ignored.

TERMINATION

The TERMINATION IMOD is invoked as an IMOD task during ISERVE address space termination and occurs when a STOP command is received. Processing of the TERMINATION IMOD proceeds until it either completes or a STOP FORCE command is entered. STOP FORCE immediately terminates all processing. This IMOD is not entered during abnormal shutdown.

Arguments

No arguments are passed to the IMOD.

Return Values

If any results are returned by this IMOD, they are ignored.

Compiler Directive Statements

This section describes the CA-GSS compiler directive statements.

#DESC

Syntax

#DESC *text*

The contents of the *text* operand of the first #DESC statement appears in the description field of the IMOD selection panel under the IMOD editor. The text of all #DESC statements appears in batch mode IMOD listings.

#CALLABLE

Syntax

#CALLABLE

You must include this directive if you want to call this IMOD from another IMOD as an external subroutine.

#SOURCE

Syntax

#SOURCE

Including the #SOURCE directive causes ISERVE to load the source statements into memory along with the object code. This allows you to use the REXX TRACE instruction and SOURCELINE() function; both useful for debugging. It does, however, increase your use of storage.

#REFORMAT

Syntax

If the #REFORMAT directive is encountered by the compiler, it reformats your source IMOD. This directive can appear multiple times in an IMOD. The scope of #REFORMAT is from the point where inserted until the end of file or next #REFORMAT directive.

The operands are:

margin

Specifies the left-most column for REXX code and comments. All lines will be adjusted to start in this column or to the right of it. Lines that contain a non-blank in column 1 will not be adjusted. The values for *margin* are 2 through 50.

indent

Specifies an indent amount that is added to the *margin* value for each level of nesting (DO loops) to determine the starting column for the line. Lines that contain a non-blank in column 1 will not be adjusted. The values for *indent* are 1 through 10.

When using #REFORMAT,

- One of the most difficult programming errors to locate is a missing or extra END statement. The #REFORMAT directive makes it easy to locate the problem.
- Syntax errors in REXX statements can cause loss of indents. This is corrected when the syntax is corrected and the IMOD is recompiled.
- Source lines that contain a non-blank character in column 1 are not reformatted.
- Indenting causing truncation (source lines are limited to 255 characters) is reduced to the point where truncation is avoided.
- Compiler directives must begin with the pound sign (#) in column 1 and the first letter of the directive must begin in column 2.

Code Batch Maintenance Commands

The batch maintenance commands are shown in the following format:

command parameters

command

Name of the batch maintenance command.

parameters

Parameters for the command.

Syntax Coding Rules

Follow these guidelines when you code batch maintenance commands:

- To indicate that a record is a comment, use an asterisk (*) as the first non-blank character.
- To indicate that a command is continued on the next record, end the line to be continued with a single hyphen (-) separated from other text by at least one blank on both sides.
- Use blanks to delimit operands.
- Blank records are ignored.

Batch Maintenance Command Overview

The following table describes each command for maintaining IMODs in batch mode.

Command	What It Does
ALLOC_DSN	Allocates a new non-VSAM sequential file or PDS.
ALLOC_ILOG	Allocates a new ILOG subfile (VSAM Linear Data Set).
ALLOC_ISET	Allocates a new ISET file (VSAM KSDS).
CHANGE	Changes the executable status of an IMOD.
CLOSE	Explicitly closes a file.
COMPILE	Compiles an IMOD.
COPY	Copies an IMOD from a sequential file to an IMOD file or from one IMOD file to another.
DELETE	Deletes an IMOD.
DUMP	Dumps an IMOD object code (for CA analysis).
DYNALLOC	Dynamically allocates a file to SRVMAINT.
EXTRACT	Extracts an IMOD from an IMOD file and writes it to a sequential file.
FLDC	Specifies the fixed-length pattern-matching symbol.
INITIALIZE	Initializes an IMOD file (ISET) for use by CA-GSS.
LINECOUNT	Sets the number of lines to be printed on a page.
LIST_IMOD	Lists an IMOD to a print file.
LIST_ISET	Lists the contents of an IMOD file in directory format.
MAXCC	Sets the maximum completion code for continuation.

Command	What It Does
NAME_LIST	Creates a list of IMOD names for other operations.
PACKAGE	Packages IMODs into a load module.
PDSLOAD	Loads PDS members into an ISET.
RENAME	Renames an IMOD.
SCAN	Searches IMODs for specific character strings.
SCRATCH	Deletes an entire ISET (VSAM KSDS).
UPGRADE	Compiles all IMODs in an ISET.
VIO	Specifies the unit name for VIO work files.
VLDC	Specifies the variable-length pattern-matching symbol.

ALLOC_DSN

Use the ALLOC_DSN command to create a sequential file or a new PDS.

Syntax

Use this format for the ALLOC_DSN command:

```
ALLOC_DSN DSNAME dsname UNIT unit [VOLUME volume] [DDNAME ddname]

  SPACE {blk|CYL|TRK},prim[,sec[,dir] [RECFM recfm] [LRECL lrecl]

    [BLKSIZE blksize]
```

Parameters

The ALLOC_DSN command takes these parameters:

dsname

Name of the data set to be allocated. This parameter is required.

volume

Unit address or symbolic name of the device where the data set is to be allocated. This parameter is required.

ddname

DDname assigned to the newly allocated data set. If specified, this value may be used by the SRVMAINT program to direct output to the new data set.

SPACE

The values that follow this required parameter determine how much space is reserved for the data set.

blk

Indicates that the values specified for *prim* and *sec* are for data blocks of the size *blk*.

CYL

Indicates that the values specified for *prim* and *sec* are in cylinder increments.

TRK

Indicates that the values specified for *prim* and *sec* are in track increments.

prim

Amount of space initially reserved for the data set.

sec

If specified and not zero, specifies a secondary amount of space that is allocated whenever the data set is full.

dir

If specified and not zero, specifies the number of 256-byte blocks reserved for a PDS directory. A non-zero value for *dir* allocates a PDS.

recfm

Record format to be assigned to the data set. Permitted values are: F, FA, FM, FB, FBA, FBM, V, VA, VM, VB, VBA, VBM, and U. The default is FB.

lrecl

Logical record length assigned to the data set. The default is: 80 for RECFM=F; 255 for RECFM=V; 0 for RECFM=U

blksize

Block size assigned to the data set. The default is *lrecl* X 10.

Usage Note

There are many additional attributes that you may assign to a data set. If you require any of these, you should use a JCL allocation strategy.

ALLOC_ILOG

Use the ALLOC_ILOG command to allocate an ILOG subfile.

Syntax

Use this format for the ALLOC_ILOG command:

```
ALLOC_ILOG NAME clname [VOLSER volser] {CYL|TRK} prim sec
```

Parameters

The ALLOC_ILOG command takes these parameters:

clname

Cluster name, from 1 to 38 characters.

volser

Volume serial number of the volume to contain the data set.

CYL

Space allocation (*prim* and *sec*) is specified in cylinders.

TRK

Space allocation (*prim* and *sec*) is specified in tracks.

prim sec

Space, in cylinders or tracks, for primary and secondary extents.

ALLOC_ISET

Use the ALLOC_ISET command to allocate an IMOD file. ALLOC_ISET automatically invokes the INITIALIZE command after successful allocation.

Syntax

Use this format for the ALLOC_ISET command:

```
ALLOC_ISET [DDNAME ddname] NAME clname [VOLSER volser] {CYL|TRK} prim sec
```

Parameters

The ALLOC_ISET command takes these parameters.

ddname

DDname to assign. Be sure the name you specify does not duplicate a *ddname* in the JCL. Needed only if you wish to perform additional operations on the newly allocated ISET.

clname

Cluster name, from 1 to 38 characters.

volser

Volume serial number of the volume to contain the data set.

CYL

Space allocation (*prim* and *sec*) is specified in cylinders.

TRK

Space allocation (*prim* and *sec*) is specified in tracks.

prim sec

Space, in cylinders or tracks, for primary and secondary extents.

Usage Note

After you allocate a file, you can perform operations on the new file by using the *ddname* that you assigned.

CHANGE

Use the CHANGE command to change the status of an IMOD.

Syntax

Use this format for the CHANGE command:

```
CHANGE IMOD {imodname|&LISTn} IN ddname STATUS {PROD|TEST|FORCE}
```

Parameters

The CHANGE command takes these parameters:

Parameter	Description
imodname	Name of the IMOD.
&LIST*n*	Name of the list containing the names of the IMODs to be changed. See "NAME_LIST" for details on using name lists.
ddname	DDname that identifies the ISET containing the IMOD.
PROD	Assigns production status to the IMOD.
TEST	Assigns test status to the IMOD.
FORCE	Assigns force-load status to the IMOD.

CLOSE

Use CLOSE to close a file explicitly and free the associated storage. SRVMAINT opens files as needed and closes them only when required or at the end of all processing.

Syntax

Use this format for the CLOSE command:

```
CLOSE ddname
```

Parameters

The CLOSE command takes this parameter:

ddname DDname of the file to be closed.

COPY

Copies an IMOD from a sequential file to an IMOD file or from one IMOD file to another.

Syntax

Use this format for the COPY command:

```
COPY IMOD {name1|&LISTn} TO dd1 FROM dd2 [REPLACE] [NEWNAME name2]
```

Parameters

The COPY command takes these parameters:

Parameter	Description
name1	Name of the IMOD to copy. This name is assigned to the new IMOD unless you specify NEWNAME.
&LIST*n*	Name of the list containing the IMODs to be compiled.
dd1	DDname of the target IMOD file.
dd2	DDname of the input file, either an IMOD file or a sequential file.
REPLACE	Copy operation replaces an existing IMOD of the same name. If an IMOD of the same name does not already exist, it is added.
name2	Name to assign to the new IMOD. If *name1* is specified as a list, you cannot specify *name2*. The default is *name.*

Usage Note

To copy an IMOD to a sequential file, use the EXTRACT command.

COMPILE

Use the COMPILE command to compile an IMOD.

Syntax

Use this format for the COMPILE command:

```
COMPILE IMOD {imodname|&LISTn} IN ddname [VIO unit] [STATUS status]
```

Parameters

The COMPILE command takes these parameters:

Parameter	Description
imodname	Name of the IMOD.
&LIST*n*	Name of the list that contains the names of the IMODs to be compiled. See "NAME_LIST" for information on using name lists.
ddname	DDname of the IMOD file.
unit	JCL UNIT name to be used for a VIO work file. The default is the value last specified by either the VIO operation or COMPILE operation with VIO specification.
status	The status of the IMOD. Specify one of the following: **FORCE** Force-load status. **PROD** Production status. **TEST** Test status. **SAME** Leaves the status of the IMOD unchanged; the default.

Usage

The COMPILE command dynamically invokes the SRVCOMP module. This module must be accessible using STEPLIB, JOBLIB, or LINKLIST.

DELETE

Use the DELETE command to delete an IMOD.

Syntax

Use this format for the DELETE command:

```
DELETE IMOD {imodname|&LISTn} FROM ddname
```

Parameters

The DELETE command takes these parameters:

Parameter	Description
imodname	Name of the IMOD to delete.
&LIST*n*	Name of the list that contains the names of the IMODs to be deleted. See NAME_LIST" for information on using name lists.
ddname	DDname of the file containing the IMOD.

DUMP

Use the DUMP command to dump an IMOD file to a print file for CA Technologies Support to analyze.

Syntax

Use this format for the DUMP command:

```
DUMP IMOD {imodname|&LISTn} TO ddname1 FROM ddname2 [LINECOUNT nn]
```

Parameters

The DUMP command takes these parameters:

Parameter	Description
imodname	Name of the IMOD to dump.
&LIST*n*	Name of the list that contains the names of the IMODs to be dumped. See "NAME_LIST" for information on using name lists.
ddname1	DDname of the print file.
ddname2	DDname of the IMOD file.

Parameter	Description
nn	Number of lines to be printed on each page of the report. **Default**: Current default.

DYNALLOC

Use the DYNALLOC command to dynamically allocate existing disk files to SRVMAINT.

Syntax

Use this format for the DYNALLOC command:

```
DYNALLOC DDNAME [ddname] DSN [dsn] [disp]
```

Parameters

The DYNALLOC command takes these parameters:

Parameter	Description
ddname	DDname to be assigned to the file.
dsn	Data set name of the file to be allocated.
disp	Data set disposition when allocated. Specify one of the following: **OLD** DISP=SHR status. **SHR** DISP=OLD status; the default.

Usage Notes

- If the data set is not immediately available, the request fails.
- Ensure that the specified ddname does not duplicate any coded in the JCL.

EXTRACT

Use the EXTRACT command to extract an IMOD from an IMOD file and write the IMOD to a sequential file or a PDS.

Syntax

Use this format for the EXTRACT command.

```
EXTRACT IMOD {imodname|&LISTn} TO ddname1 FROM ddname2

  [ADD|REPLACE] [MEMBER name] [ISPF] [SEQNUM] [DIAG]
```

Parameters

The EXTRACT command takes these parameters.

Parameter	Description
imodname	Name of the IMOD to extract.
&LIST*n*	Name of the list that contains the names of the IMODs to be extracted. See "NAME_LIST" for information on using name lists. If the target data set is sequential, all IMODs in the list will be concatenated in the target data set. If the target data set is a PDS, each IMOD is copied to a separate member. Before beginning any extract, a list of member names is internally generated. The member names are the IMOD names when the IMOD names are eight characters or less. If an IMOD name exceeds eight characters, the member name will be the last eight characters. If the generated member name list contains any duplicates, the last character is replaced with uppercase *A*. If this still results in duplication, the last character is replaced with *B*, *C*, and so forth, until a unique name is produced. If the alphabet is exhausted and the name is still duplicated, the extract for that IMOD is suppressed. The ADD/REPLACE operand applies to the generated member names.
ddname1	DDname identifying the source IMOD file.
ddname2	DDname identifying the target file.
ADD	For a PDS operation, specifies that members are to be added only. If the member already exists, the extract is suppressed. This is the default.
REPLACE	For a PDS operation, specifies that members are always to be added. If the member already exists, its content is replaced.

Parameter	Description
name	For a PDS operation on a single IMOD, the *name* operand permits you to specify the member name that is assigned in the target PDS. If you omit this operand, the IMOD name is used as the member name. If the IMOD name exceeds eight characters, the last eight characters are used.
ISPF	When extracting to a PDS, requests that ISPF-compatible statistics be maintained in the target data set.
SEQNUM	Adds the IMOD's sequence numbers as comment fields to the start of each record.
DIAG	Copies compiler diagnostic messages along with the source.

Usage Note

If a record is too long to be written to the output data set, it is truncated and a warning is written to the SYSPRINT file.

FLDC

Use the FLDC command to specify a character to be used as the Fixed-Length symbol during pattern matching. When found in a pattern, the FLDC character matches a single character of any value (including blank).

Syntax

Use this format for the FLDC command:

FLDC *char*

Parameters

The FLDC command takes this parameter:

char Any character, except blank. The default is *.

INITIALIZE

Use the INITIALIZE command to initialize (or reinitialize) an IMOD file. This updates the ISET header record to the current CA-GSS version number.

Syntax

Use this format for the INITIALIZE command:

```
INITIALIZE ISET ddname [ERASE]
```

Parameters

The INITIALIZE command takes these parameters.

Parameter	Description
ddname	DDname of the file to initialize.
ERASE	If specified, all existing IMODs are deleted. The file must be allocated with DISP=OLD for this option to be valid. ERASE is the default for ISETs that have not been previously accessed.

LINECOUNT

Use the LINECOUNT command to set a new default for the number of lines printed on each report page (including SYSPRINT).

Syntax

Use this format for the LINECOUNT command:

```
LINECOUNT lines
```

Parameters

The LINECOUNT command takes this parameter.

lines

Number of lines to be printed on each report page.

LIST_IMOD

Use the LIST_IMOD command to list the contents of an IMOD to a print file.

Syntax

Use this format for the LIST_IMOD command:

```
LIST_IMOD {imodname|&LISTn} TO ddname1 FROM ddname2
  [LINECOUNT lines]
```

Parameters

The LIST_IMOD command takes these parameters.

Parameter	Description
imodname	Name of the IMOD to be listed.
&LIST*n*	Name of the list that contains the names of the IMODs to be listed. See "NAME_LIST" for information on using name lists.
ddname1	DDname of the print file.
ddname2	DDname of the IMOD file containing the IMOD.
lines	Number of lines to be printed on each page. This *lines* value remains in effect for the output *ddname* until it is overridden. Default: Current default.

LIST_ISET

Use the LIST_ISET command to list the contents of an IMOD file to a print file.

Syntax

Use this format for the LIST_ISET command:

```
LIST_ISET ddname1 TO ddname2 [LINECOUNT lines]
```

Parameters

The LIST_ISET command takes these parameters:

Parameter	Description
ddname1	DDname of the IMOD file to be listed.

Parameter	Description
ddname2	DDname of the print file.
lines	Number of lines to be printed on each page. The *lines* value remains in effect for the output *ddname* until it is overridden. Default: Current default.

MAXCC

Use the MAXCC command to set the maximum completion code for continuation.

Syntax

Use this format for the MAXCC command:

```
MAXCC value [RESET]
```

Parameters

The MAXCC command takes these parameters:

Parameter	Description
value	Maximum completion value for an operation. If this value is exceeded, all following operations are skipped. The default is 4.
RESET	Resets the maximum completion value to the value specified. This will prevent a code from a previous operation from aborting a run

Usage Note

Once operations are being suppressed due to errors, MAXCC will also be suppressed.

NAME_LIST

In addition to operating on individual IMODs, many maintenance commands may operate on lists of IMODs. Use the NAME_LIST command to create and modify name lists.

Syntax

Use this format for the NAME_LIST command:

```
NAME_LIST &LISTn [CLEAR] [INCLUDE iset /pattern/]
                         [INCLUDE PDS ddname /pattern/]
                         [INCLUDE NAME name]
  [EXCLUDE pattern] [COPY &LISTn] [MERGE &LISTn] [PRINT]
```

Parameters

The NAME_LIST command takes these parameters.

Parameter	Description
&LIST*n*	Name of the list that is the target for the NAME_LIST operation. Three lists are available: &LIST1, &LIST2, and &LIST3.
CLEAR	Deletes the current contents of the list.
INCLUDE	Includes additional names in the list. Names are added only when they match the specified pattern.
iset	DDname of the ISET from which IMOD names are to be fetched.
ddname	DDname of the PDS from whose directory names are to be fetched.
name	Specifies a single name to be added to the list.
pattern	Comparison pattern to be applied against an IMOD name. The pattern may consist of both characters and placeholders. An asterisk (*) will match any single character. An ampersand (&) will match 0 through *n* characters. The pattern must be enclosed by a pair of arbitrary delimiter characters (shown as slashes (/) in the format). Any character that does not appear in the pattern except blank, VLDC, and FLDC may be used as the delimiter.
EXCLUDE	Any name in the list that matches the pattern will be deleted from the list.
COPY	Contents of the specified list replaces the target list.

Parameter	Description
MERGE	Contents of the specified list is merged into the target list. Duplicate names are discarded.
PRINT	Contents of the target list is printed.

Usage Notes

- Lists are maintained in ascending collating sequence (alphabetical order).
- Duplicate names are automatically deleted.
- Operands are processed left to right, as they are encountered.

PACKAGE

The PACKAGE command converts IMODs into z/OS object decks.

Syntax

Use this format for the PACKAGE command:

```
PACKAGE {IMOD {name|&LISTn} TO ddname1 FROM ddname2}
        {COMPLETE TO ddname1 ENTRY imodname}
        {PARMS string}
```

Parameters

The PACKAGE command takes these parameters.

Parameter	Description
name	Name of the IMOD to package.
&LIST*n*	Name of the list containing the IMODs to be packaged.
ddname1	DDname identifying the source ISET.
ddname2	DDname identifying the package PDS. This PDS should be empty and must have RECFM=FB, LRECL=80, and BLKSIZE 800.
imodname	Name of the IMOD to receive control when package is invoked.
string	Parameter string to be passed to the initialization phase or the package. The content may be any parameter string that may be passed to the SRVBATCH program using the SRVPARM DD statement.
IMOD	Adds an IMOD or list of IMODs to be added to the package.

Parameter	Description
COMPLETE	Completes the package. No additional IMODs may be added to it.
PARMS	Text that follows is used for runtime initialization. Can be repeated on subsequent cards.

Usage Note

Following creation of the package PDS you must execute assembly and link-edit steps.

PDSLOAD

Use the PDSLOAD command to load IMODs from a PDS to an IMOD file.

Syntax

Use this format for the PDSLOAD command:

```
PDSLOAD MEMBER {member|&LISTn} TO ddname1 FROM ddname2 [ADD|REPLACE] [IMOD name]
```

Parameters

The PDSLOAD command takes these parameters:

Parameter	Description
member	Name of the PDS member to copy to the ISET.
&LIST*n*	Name of list containing members to be loaded. Each member is copied to a separate IMOD of the same name.
ddname1	DDname identifying the source PDS file.
ddname2	DDname identifying the target ISET.
ADD	IMODs are added only. If it exists, load is suppressed; the default.
REPLACE	Always adds IMODs. If IMOD already exists, replaces its content.
name	Target IMOD name. Cannot be used if MEMBER specifies a list. If omitted, the IMOD *name* is the same as the member name.

RENAME

Use the RENAME command to rename an IMOD.

Syntax

Use this format for the RENAME command:

```
RENAME IMOD imodname IN ddname NEWNAME newname
```

Parameters

The RENAME command takes these parameters:

Parameter	Description
imodname	Name of the IMOD to be renamed.
ddname	DDname of the file containing the IMOD.
newname	New name to be assigned to the IMOD.

SCAN

Use the SCAN command to search one or more IMODs for the presence or absence of text strings.

Syntax

Use this format for the SCAN command.

```
SCAN IMOD imodname IN ddname [FLDC char] [VLDC char]

  ON {MATCH|NOMATCH} {ADD &LISTn}    PATTERN / patt / [AND|OR / patt /]
                     {DELETE &LISTn}
                     {LIST_LINE}

    [ON {MATCH|NOMATCH} {ADD &LISTn}    PATTERN / patt / [AND|OR / patt/]
                        {DELETE &LISTn}
                        {LIST_LINE}
```

Parameters

The SCAN command takes these parameters.

Parameter	Description
imodname	Name of the IMOD to be scanned.
&LIST*n*	Name of the list containing the IMODs to be scanned.
ddname	DDname of the file containing the IMOD (or IMODs if &LIST*n* is specified).
FLDC	Specifies the Fixed-Length Do not Care character used in patterns that follow, either until it encounters another FLDC parameter or reaches the end of SCAN. If FLDC is not specified, uses the value in effect from the execution of a previous FLDC command. FLDC character matches exactly one character, including blank.
char	Any single character, except blank, to be used as a placeholder.
VLDC	Specifies the Variable-Length Do not Care character to be used in the patterns that follow, either until another VLDC parameter is encountered or the end of the SCAN command is reached. If VLDC is not specified, the value in effect from the execution of a previous VLDC command is used. The VLDC character matches any character string of any length (including a null string).
ON	Signifies the start of a scan operation. You may perform multiple scan operations simultaneously by providing additional specifications.
MATCH	Action will be taken if the patterns are matched.
NOMATCH	Action will be taken if the patterns are not matched.
ADD	IMOD name is to be added to the specified &LIST*n*.
DELETE	IMOD name is to be deleted from the specified &LIST*n*.
LIST_LINE	REXX source code line is to be listed if the specified criteria is met.
PATTERN	String that follows is the first pattern to be applied.
patt	Comparison string to be applied to each physical line of the IMOD. This string may contain characters and placeholders. Each pattern must be enclosed by a pair of delimiter characters (shown as slashes (/) in the format). Any character that does not appear in the pattern except blank, VLDC, and FLDC may be used as the delimiter.

Parameter	Description
AND	Pattern that follows must match the same physical IMOD line as all the other patterns specified in this SCAN command for the match condition to be met. The AND parameter and its accompanying pattern may be specified as many times as needed.
OR	If the pattern that follows is matched, the match condition is met whether the first pattern was matched or not. Specify the OR parameter and its accompanying pattern as many times as needed.

Usage Notes

- Scans are against physical source lines. Continuations of source statements are not recognized.
- Variable substitution is not done. Only the physical source code is examined.
- String comparisons are not case-sensitive.
- If wildcard characters are not included in your pattern string, a match will not occur unless the entire source line matches the pattern.
- The MATCH or NOMATCH action is performed if the match flag is turned on following the application of all comparisons to a source line.
- Comparisons are performed in the order that you specify them.
- The match flag is turned on or off after applying the first pattern. Each additional pattern is compared and the match flag is turned on or off according to the following tables:

And	Match	No Match
Flag ON	ON	OFF
Flag OFF	OFF	OFF
OR	**Match**	**No Match**
Flag ON	ON	ON
Flag OFF	ON	OFF

SCRATCH

Use the SCRATCH command to delete an IMOD or ILOG file.

Syntax

Use this format for the SCRATCH command:

```
SCRATCH dsname
```

Parameters

The SCRATCH command takes this parameter:

dsname

Data set name of the IMOD or ILOG data set to be deleted.

UPGRADE

Use the UPGRADE command to compile all IMODs in an ISET.

Syntax

Use this format for the UPGRADE command.

```
UPGRADE FILE ddname [ALL]
```

Parameters

The UPGRADE command takes these parameters:

ddname

DDname of the ISET.

ALL

Compiles all IMODs in the ISET. If ALL is omitted, only IMODs that were last compiled with a previous (or later) release of the compiler are compiled.

Usage Note

The UPGRADE command dynamically invokes the SRVCOMP module. This module must be accessible using STEPLIB, JOBLIB, or LINKLIST.

VIO

Use the VIO command to specify the unit name for VIO work files.

Syntax

Use this format for the VIO command:

```
VIO name
```

Parameters

The VIO command takes this parameter:

name

Unit name to be used for VIO work data sets. The default is VIO.

VLDC

Use the VLDC command to specify a character to be used as the Variable-Length Do not Care symbol during pattern matching. When found in a pattern, the VLDC character matches a string of any length, including null.

Syntax

Use this format for the VLDC command:

```
VLDC char
```

Parameters

The VLDC command takes this parameter:

char

Any character, except blank. The default is &.

IMOD Editor Commands

This section explains the IMOD editor commands.

B

Use the B command to display but not change the contents of an IMOD or to select an ISET in read-only mode.

Ways to Issue It

Specify B in the input field next to the name of the IMOD on the ISET or IMOD panel.

Usage Note

When you issue a B command, the IMOD editor invokes the ISPF BROWSE facility and opens the Edit panel. Only commands provided by the ISPF BROWSE facility are available.

C

Use the C command to compile an IMOD.

Ways to Issue It

Specify C in the input field next to the IMOD name on the IMOD panel.

Usage Notes

- To tell whether a compile succeeded, look at the first character in the STATUS field of the IMOD panel. C indicates that the IMOD compiled successfully; * indicates that a compile error occurred.
- If an error occurs, standard REXX error messages are added to the source code. These messages are preceded by a non-displayable character (X'00'), which you can locate with either the ISPF FIND P'.' command or CA-GSS' ERROR edit macro. If you do not manually delete these messages, they are automatically deleted during the next compile.

D

Use the D command to delete an IMOD.

Ways to Issue It

Specify D in the input field next to the IMOD name on the IMOD panel.

Usage Note

Before the IMOD is deleted, you are prompted to confirm the delete request. At the prompt, specify YES to delete the IMOD. To cancel the deletion, press the PF3 key or specify anything other than YES.

DATASETS

Use the DATASETS command to display a panel that shows information about all data sets that the IMOD editor is using.

Ways to Issue It

Go to the command line of the ISET or IMOD panel and issue this command:

```
DATASETS
```

Usage Note

To return to the panel that you were viewing, press the Enter key.

EDIT

Use the EDIT command to edit an IMOD using the PDF editor or to select an ISET in update mode.

Ways to Issue It

Use either of these methods to issue the EDIT command:

- Specify E in the input field next to the name of the IMOD on the ISET or IMOD panel.
- Go to the panel command line and issue this command:

  `Edit` *`imodname`*

Usage Notes

When you want to create an IMOD, you must issue the EDIT command from a panel command line.

- Although you can use standard ISPF editing commands to edit an IMOD, do not use the NUM ON command or enter line numbers manually-REXX will treat line numbers as syntax errors.
- To save an IMOD and return to the IMOD panel, press the PF3 key. After you have edited it, you are ready to compile it, assign it production status, and load it.

F

Use the F command to change an IMOD status to force-load. When you do this, the status is changed to production and any previously loaded IMOD with the same name is replaced.

Ways to Issue It

Specify F in the input field next to the IMOD name on the IMOD panel.

Usage Notes

- When an IMOD has force-load status, it can be executed by the ISERVE address space.
- The difference between the F command and the P command is that F replaces any previously loaded IMOD that has the same name-P will not.

G

Use the G command to compile an IMOD, change its status to production, and load it.

Ways to Issue It

Specify G in the input field next to the IMOD name on the IMOD panel.

Usage Note

If an IMOD does not compile successfully, its status will not be changed to production and it will not be loaded.

L

Use the L command to load a particular IMOD into a CA-GSS address space.

Ways to Issue It

Specify L in the input field next to the IMOD name on the IMOD panel.

Usage Notes

- An IMOD that already exists will be replaced if it is reloaded.
- Only IMODs with both compiled and production (or force-load) status can be loaded.
- Before you issue the L command, make sure the SSNAME field contains the address name for the target ISERVE.
- You can use the PF1 key (or issue a HELP command) to get more information about the message that is displayed as a result of the L command.

LINK

Use the LINK command to link to a new CA-GSS address space.

Ways to Issue It

Go to the ISET panel command line and issue this command:

```
LINK ssid
```

ssid is the subsystem ID for a CA-GSS address space on the local z/OS image.

Usage Notes

- When you link to a new CA-GSS address space, the IMOD editor creates a menu that lists all ISETs that are available to that address space.
- After linking to a new CA-GSS address space, the IMOD editor begins using the default data sets defined for that address space (using the EDITOR initialization parameter). As a result, it may begin using different ISPF libraries, CLIST libraries, etc. A new IMOD compiler may also be loaded.

  Note: The IMOD editor cannot change ISPF data sets if you are currently in split-screen mode.

LOCATE

Use the LOCATE command to position the screen to a particular ISET or IMOD.

Ways to Issue It

Go to the ISET or IMOD panel command line and issue this command:

```
Locate name
```

name is the ISET or IMOD name (or leading fragment).

Usage Notes

- Case is ignored when matching the name to the value of the LOCATE command.
- Do not specify a string with quotes.

MEMBER

Use the MEMBER command to use a different menu of ISETs.

Ways to Issue It

Go to the ISET panel command line and issue this command:

`MEMBER member [DEFAULT]`

member

Name of the member (or * for the system default member).

DEFAULT

Saves the member name in your ISPF profile so that it is automatically used the next time you start the IMOD editor.

P

Use the P command to change the status of an IMOD to production.

Ways to Issue It

Specify P in the input field next to the IMOD name on the IMOD panel.

Usage Notes

- When an IMOD has production status, it can be executed by the ISERVE address space.
- The difference between the P command and the F command is that P will not replace any previously loaded IMOD that has the same name-F will.

R

Use the R command to change an IMOD name.

Ways to Issue It

Specify R in the input field next to the IMOD name on the IMOD panel.

Usage Note

You will be prompted for the new IMOD name. To exit the panel without completing the rename operation, press the PF3 key or issue an END command from the command line.

SELECT

Use the SELECT command to edit an IMOD or, if the ISET has read-only status, to browse the IMOD.

Ways to Issue It

Use either of these methods to issue the SELECT command:

- Specify S in the input field next to the ISET or IMOD name on the ISET or IMOD panel.
- Go to the panel command line and issue this command:

  `Select` *`name`*

Usage Notes

- If * appears at the start of an ISET description, the ISET has read-only status.
- You can also use the EDIT and BROWSE commands to work with IMODs. EDIT lets you edit an IMOD, and BROWSE lets you look at the contents of an IMOD.

SORT

Use the SORT command to display ISETs in collating sequence.

Ways to Issue It

Issue this command from the ISET panel command line:

```
SORT
```

Usage Notes

- By default, ISETs are displayed in the order that they were specified in the parameter member.
- When ISETs are sorted in collating sequence, it is easier to use an ISPF LOCATE command to position the screen to a particular ISET.

T

Use the T command to change the status of an IMOD to test.

Ways to Issue It

Specify T in the input field next to the IMOD name on the IMOD panel.

Usage Notes

- When an IMOD has test status, it will not be executed by the ISERVE address space. However, it can still be loaded and executed in batch mode using the SRVBATCH program.
- An IMOD is automatically assigned test status after it is compiled.

TOGGLE

Use the TOGGLE command to switch between the current panel and another panel-a panel that shows an ISET data set name (when issued from the ISET Selection Panel), or an alternate form of the IMOD Selection Panel (when issued from the IMOD Selection Panel).

Ways to Issue It

Go to the ISET or IMOD panel command line and issue this command:

```
TOGGLE
```

Usage Notes

- To return to the panel that you were viewing, issue the TOGGLE command again.
- TOGGLE lets you switch between two forms of the IMOD Selection Panel-a form that shows date of last change and user ID of changer, or a form that shows the compiler version used for the IMOD and the number of source lines in the IMOD.

VERSION

Use the VERSION command to display the current editor and compiler versions.

Ways to Issue It

Go to the ISET or IMOD panel command line and issue this command:

```
VERSION
```

Usage Note

To return to the panel that you were viewing, click Enter.

X

Use the X command to execute an IMOD in an ISERVE address space and to retrieve any output that is produced.

Ways to Issue It

Use these methods to issue the X command:

- Specify X in the input field next to the IMOD name on the IMOD panel.
- Go to the panel command line and issue this command:

  X *imodname*

Usage Note

An IMOD is not loaded into the target ISERVE address space before it is executed. If you have modified the IMOD, you need to use either the L or G command first to load its most current form into the ISERVE address space.

GSS Extended Functions

The following tables provide summary information on the CA-GSS extended functions.

Stack Functions

These functions are stack-related functions.

Function	Description
PUBSTACK()	Controls public access to stack data.
PULL()	Removes a record from the top of a specified stack.
PUSH()	Adds a record to the top of a specified stack.
QUEUE()	Adds a record to the bottom of a specified stack.
QUEUES()	Adds an entire stack onto the end of another specified stack.
SHOVE()	Inserts a record into a specified stack at a specified location.
SORT()	Sorts a specified stack by field contents.
STACKINF()	Returns information on a stack and its contents.
SWAPSTAK()	Changes to a new stack.
TUG()	Returns a copy of the specified record from the specified stack.

Function	Description
WAIT()	Waits for a record to be placed in a stack.
YANK()	Retrieves and removes a record from within a specified stack.

IO Functions

These functions are input/output-related functions:

Function	Description
$ISPFSTAT()	Decodes and encodes ISPF PDS member statistics.
ALLOC()	Dynamically allocates new or existing data sets.
DB2()	Issues a dynamic SQL request to DB2.
DEALLOC()	Dynamically deallocates a specified file.
SAM()	Processes a sequential file or PDS.
VSAM()	Performs I/O operations on VSAM files.

External Communication Functions

These functions are related to external communication.

Function	Description
CALLX()	Invokes an external REXX subroutine at another GoalNet node.
CP()	Issues a VM CP command and returns the CP responses.
DOM()	Deletes a message from the operator console.
LU2()	Enables support for VTAM LU 2 devices.
MLWTO()	Issues a multi-line WTO.
OPSVALUE()	Inspects and sets values of CA OPS/MVS Event Management and Automation variables.
OSCMD()	Issues an operator command.
REDIRECT()	Redirects TRACE or SAY output to a stack.
SAYWHAT()	Sets the destination for SAY() output.
SETADDR()	Specifies the destination for ADDRESS commands.

Function	Description
SRVCALL()	Calls an IMOD in an ISERVE address space from SRVBATCH and TSO/E REXX.
SRVOPS()	Calls an IMOD in an ISERVE address space from CA OPS/MVS Event Management and Automation.
SUBMIT()	Submits a job to JES2.
WTO()	Issues a one-line WTO.
WTOR()	Issues a one-line WTOR (WAIT is implied).

Number, String, and Variable Functions

These functions are related to numbers, strings, and variables.

Function	Description
$3270()	Creates 3270 data stream from English descriptions.
$ALIAS()	References and maintains an alias table.
$GLOBAL()	Allocates global variables of more than 80 bytes in length.
$GREG2JUL()	Converts Gregorian-format date to Julian format.
$JUL2GREG()	Converts Gregorian-format date to Julian format.
$TIME()	Manipulates CPU timer values (double-word binary).
ALIGN()	Returns a numerical string with an aligned decimal point.
BOOLWORD()	Performs a Boolean operation on two values and returns the numeric result.
CALC()	Evaluates complex numerical expression and returns result.
CASE()	Translates text to uppercase or lowercase.
D2P()	Converts decimal number to IBM packed decimal.
DS3270()	Creates and parses 3270 data stream.
KEYVAL()	Scans an input string for a key.
LOCSTEM()	Retrieves index values of stem variables.
P2D()	Converts a packed-decimal number to decimal.

Function	Description
SCANSTR()	Scans a string with complex pattern matching.
UNIQUE()	Returns a character not appearing in a string.
VVALUE()	Accesses variables in the calling program chain.

System Information Functions

These functions are related to system information:

Function	Description
DASD()	Returns information about DASD volumes.
DEVTYPE()	Issues the IBM macro DEVTYPE SVC.
DSNENQ()	Returns information about the enqueue status of a data set in predefined stem variables.
LOCATE()	Returns the memory address of a specified item.
MEMORY()	Inspects or alters memory.
RDJFCB()	Retrieves the job file control block (JFCB) for a specified ddname.
SECURITY()	Interfaces with the system security software.
SYSVAL()	Returns system values.

MOD Execution and Environment Functions

These functions are related to IMOD execution and environment:

Function	Description
$SERVER()	Controls server IMODs.
$TIMER()	Sets one or more asynchronous timers.
$WHEREFROM()	Determines where an IMOD was called from.
BATCHRC()	Sets STEP return code during batch execution.
CPUTIME()	Returns elapsed CPU time for IMOD task.
ENQUEUE()	Reserves resources for the use of an IMOD task.
LOADIMOD()	In batch mode, causes the specified IMODs to be loaded into storage and made available for processing.
LOCIMOD()	Determines the availability and location of an IMOD.

Function	Description
PAUSE()	Delays execution of an IMOD for a specified number of seconds.
SCRASID()	Sets the default ILOG or IMOD by origination ASID.
SETRC()	Specifies returned value of RC in external subroutine.
SPAWN()	Begins the simultaneous execution of another IMOD.
VARSIZE()	Sets the maximum length of REXX variable string.

ILOG File Functions

These functions are related to ILOG files:

Function	Description
ILOG()	Retrieves information from an ILOG subfile.
ILOGC()	Controls Open and Close of ILOG datasets.
ILOGG()	Retrieves information from an ILOG file.
ILOGR()	Records information in an ILOG file.
IROUTE()	Controls ILOG routing of the current message.

Notation Conventions

Positional Operands

The operands for the extended functions have reserved positions. If you omit an optional operand, you must indicate its position with a comma.

Literal Strings

Values shown in capital letters are literal strings. When you code a literal string in REXX, you should always enclose it in single (') or double quotation marks ("). This is because REXX will always assume an "unquoted" literal string to be a variable name and will perform substitution. Note that if no variable of that name has been used, the variable's uninitialized value is its name.

Example

The syntax for the EXAMPLE() function might be as follows:

```
EXAMPLE({OPEN},handle[,{F|FB}[,lrecl][,blksize])
        {CLOSE}
        {START}
        {STOP}
```

If you choose to omit the F | FB and *lrecl* operands, but include *blksize*, you could enter the function in the following way:

```
EXAMPLE('OPEN',myhandle,,,3120)
```

Notice that OPEN, a literal string, is enclosed in single quotes, while *myhandle,* a variable, is not. The numeric value, 3120, although technically a literal, is not a valid REXX name and there is no ambiguity in omitting the quotes.

$3270()

Use the $3270() function to convert English commands and data into a 3270-compatible data stream. Information is passed to the function both as arguments and on the current stack.

Syntax

The $3270() function has this syntax:

```
stream = $3270([rows],[cols],[{WRITE|ERASE|NONE}],[{ALARM|RESET|RESTORE}])
```

Arguments

The $3270() function takes the following arguments:

stream

3270 data stream. This data stream is prefaced by both a command byte and a Write Control Character (WCC).

rows

Number of rows on the screen. The default value is determined by the Logon Facility.

cols

Number of columns on the screen. The default value is determined by the Logon Facility.

WRITE

Indicates that the finished data stream is intended to be written without first erasing the display. The new data will overlay and selectively replace the existing data.

ERASE

Indicates that the finished data stream is intended to completely replace any existing display.

NONE

Indicates that the finished data stream does not contain either a command or WCC. The returned data may be appended to another stream that does begin with a command and WCC. This is the default.

ALARM

Sets the *Sound audible alarm* flag in the WCC. This may appear in combination with RESTORE and RESET, separated by blanks.

RESTORE

Sets the *Restore keyboard* flag in the WCC. This may appear in combination with ALARM and RESET, separated by blanks.

RESET

Sets the *Reset modified data tag* flag in the WCC. This may appear in combination with RESTORE and ALARM, separated by blanks.

Commands

Commands are passed to $3270() using the current stack. Each command and its accompanying arguments must be complete in one stack record. Multiple commands may appear in a single stack record. Except where otherwise specified, commands and arguments are separated by blanks. Commands may be uppercase or lowercase. Where row and column coordinates are specified, they are 1-based. That is, row 1 column 1 is the first position on the screen. The following commands are valid:

AT

Use to position the display point to the specified row and column coordinates. The syntax is:

AT *row col*

UP | DOWN | LEFT | RIGHT

Use to move the display point a designated number of positions in any direction. An attempt to move off the screen in any direction results in an error condition. The syntax is:

commandname n

NL

Use to move the insertion point to the beginning of the next line, unless the insertion point is already at the beginning of a line. NL may wrap the screen to position 1,1. The syntax is:

NL

DATA

Use to copy the data that follows to the screen. No validity checking is performed and the screen may wrap in any direction. The text string is delimited by any special character that is not part of the data itself. In operation, the first non-blank character following the DATA command is used as the delimiter. Data is transferred until the delimiter is encountered again. The remainder of the stack record, if any, is assumed to contain additional commands. The syntax is:

```
DATA /text/
```

DATABEGIN

The DATABEGIN command is similar to the DATA command, except that there are no delimiters. Instead, the text is assumed to begin with the first non-blank character following the blank-delimited command word and continues to the end of the stack record. The syntax is:

```
DATABEGIN text
```

DATARECORD

Use to indicate that the next stack record is data to be written to the screen. Any remaining commands following DATARECORD in the same stack record are ignored. The syntax is:

```
DATARECORD
```

REPEAT

Use to repeat the specified single character *n* times. Screen wrapping is permitted. The values for *char* may be any single character, BLANK to repeat a blank, or NULL to repeat a null (binary zero) character. The syntax is:

```
REPEAT char n
```

ATTRIBUTE

Use to insert a 3270 attribute character with the specified characteristics. Note that each attribute requires one screen position. The arguments are delimited by any special character that does not appear in the arguments themselves. In operation, the first non-blank character following the ATTRIBUTE command is used as the delimiter. Arguments are processed until the delimiter is encountered again. The remainder of the stack record, if any, is assumed to contain additional commands. The syntax is:

```
ATTRIBUTE [MDT] [HIGH] [NODISP] [PROT] [NUM] [SKIP]
```

CURSOR

Use to position the cursor to the current display position. The syntax is:

```
CURSOR
```

Return Codes

The $3270() function produces these return codes:

101 - 103

ARG *n* MISSING OR INVALID

104

WCC not ALARM RESTORE RESET

121

INVALID AT ROW *recnum data*

122

INVALID AT COL *recnum data*

123

AT ROW TOO LARGE *recnum data*

124

AT COL TOO LARGE *recnum data*

125

FN ERROR *n type xxxx recnum data*

126

INVALID UP *recnum data*

127

UP OFF SCREEN *recnum data*

129

INVALID DOWN *recnum data*

130

DOWN OFF SCREEN *recnum data*

132

INVALID LEFT *recnum data*

133

LEFT OFF SCREEN *recnum data*

135

INVALID RIGHT *recnum data*

136

RIGHT OFF SCREEN *recnum data*

140

NO DATA AFTER DATARECORD *recnum data*

141

INVALID REPEAT CHAR *recnum data*

142

INVALID REPEAT COUNT *recnum data*

144

INVALID COMMAND: *cmd recnum data*

Example

```
queue 'at 1 1 attr /prot/ data /Enter ID =>/ attr /input high /'
queue 'cursor right 8 attr /prot/'
queue 'at 2 1 attr /prot/ data /Password =>/ attr /input nondisp/'
queue 'right 8 attr /prot/'
do i = 1 to canned_msg.0
queue 'nl datarecord'
queue canned_msg.i
end i
screen = $3270(,,'erase','reset restore')
 if rc ^= 0 then signal error
```

$ALIAS()

Use the $ALIAS() function to enable the establishment and use of tables of aliases. An alias is a string that, when it matches a minimum portion of an alias table entry, will have another value substituted for it.

Syntax

The $ALIAS() function has this syntax:

```
result = $ALIAS({ADD   },table,{alias      })
                {CREATE}       {width      }
                {DELETE}       {name,object}
                {LIST  }       {name       }
                {XLATE }
```

Arguments

The $ALIAS() function takes these arguments:

result

Contains one of these values:

(*any successful list operation*): Maximum length of any instance of *name*, followed by the maximum length of any instance of *object* (separated by a blank).

(*any other successful operation*): Affirmative message.

(*any unsuccessful operation*): Error text.

ADD

Adds an alias entry.

CREATE

Creates a new alias table.

DELETE

Deletes an alias entry.

LIST

Lists the contents of an alias table. Entries are returned on the stack, one entry per record. Fields are blank-delimited and consist of:

Alias name, in both uppercase and lowercase characters, representing the minimum required string.

Minimum required length.

Object string for translation.

XLATE

Translates the given alias string into an object.

table

Name of an alias table. This is an arbitrary name. It should not begin with a dollar sign ($), since these names are reserved for use by CA.

alias

Character string to be used in a search of the alias table. This string is not case-sensitive.

width

Maximum number of characters that can appear in any object in the table being created.

name

Name to be added to the table as an alias. The first character must be an uppercase character. During comparisons, the leading uppercase portion of the name must be present in the alias being checked. The remaining characters in the alias being checked (if any) must then match the trailing, lowercase portion of the name. When specified with a DELETE operation, the case is not checked.

object

Text string that is returned upon match by an alias.

Return Codes

The $ALIAS() function produces these return codes:

101 - 103

ARG *n* MISSING OR INVALID

121

TABLE ALREADY EXISTS

122

TABLE DOES NOT EXIST

123

NO REQUIRED CHARACTERS

124

THIS ENTRY MATCHES EXISTING ENTRY

125

ALIAS NOT FOUND

Example

```
$ALIAS('CREATE',TAB1,80)
 $ALIAS('ADD',tab1,'ABCdef','xyz')
 $ALIAS('XLATE',tab1,'abc')     == 'xyz'
$ALIAS('XLATE',tab1,'ab')      == 'ALIAS NOT FOUND'
$ALIAS('XLATE',tab1,'abcdefx') == 'ALIAS NOT FOUND'
$ALIAS('XLATE',tab1,'abcde')   == 'xyz'
```

$GLOBAL()

Use the $GLOBAL() function to allocate global variables that contain strings longer than 80 characters.

Syntax

The $GLOBAL() function has this syntax.

```
result = $GLOBAL(ALLOC,&name,length)
```

Arguments

The $GLOBAL() function takes these arguments.

result

Null string if the operation was successful. Otherwise, error text is returned.

&name

Name of a global variable to be allocated. This variable must not have been previously used in any manner. If the name is coded directly in the function, it must be enclosed in single or double quotes. The name may also be placed in another variable and that variable name specified as the argument.

length

Number of bytes to be reserved for data. Future assignments of data strings may take any length up to this value.

Usage Notes

- Strings stored in global (&) variables are normally limited to 80 characters.
- Due to the compiled nature of the implementation of REXX by CA, variables are allocated storage before IMOD execution begins. When an IMOD that contains any reference to a global variable starts to execute, the variable is pre-allocated before the first REXX instruction is executed. This is true even if the reference is not executed. Therefore, you may not code an IMOD that calls $GLOBAL() to allocate a global variable and then attempts to use the variable. What you must do instead is to execute the $GLOBAL() function and then call another IMOD to use the variable.

Return Codes

The $GLOBAL() function produces these return codes:

101 - 103

ARG *n* MISSING OR INVALID

Example

```
result = $global('alloc','&bigvar',1000)
 vname = '&bigvar'
result = &global('alloc',vname,500)
```

$GREG2JUL()

Use the $GREG2JUL() function to convert a Gregorian-format date to Julian format.

Syntax

The $GREG2JUL() function has this syntax:

juldate = $GREG2JUL(*gregdate*)

Arguments

The $GREG2JUL() function takes these arguments:

juldate

Julian-format date: *yy.ddd* or *yyyy.ddd*. The number of digits assigned to the year is the same as specified in the Gregorian format, either two or four. The day of the year is left-padded with zeros to three characters.

gregdate

Gregorian-format date: *mm/dd/yy* or *mm/dd/yyyy*. The number of digits assigned to the year will be carried over to the result and must be either two or four. The length of the month and day fields is unimportant, except that the date must be valid.

Return Codes

The $GREG2JUL() function produces these return codes:

101

ARG 1 MISSING OR INVALID

121

INVALID YEAR

122

INVALID MONTH

123

INVALID DAY

Example

```
$GREG2JUL('12/31/99') ==  '99.366'
$GREG2JUL('1/1/2000') == '2000.001'
```

$ISPFSTAT()

Use $ISPFSTAT() to decode ISPF statistics stored in the user-data field of PDS member entries. It will also encode information into ISPF-compatible format.

Syntax

The $ISPFSTAT() function has this syntax:

Form 1:

```
ver crdate chdate chtime size init mod user = $ISPFSTAT(DECODE,data)
```

Form 2:

```
$ISPFSTAT(ENCODE,ver,crdate,chdate,chtime,size,init,mod,user)
```

Arguments

The $ISPFSTAT() function takes these arguments.

ver

Version and modification level, specified as *vv.mm*. *vv* and *mm* are whole numbers from 00 to 99. The period is a separator and not a decimal point. Encoding the value 1.5 and then decoding it yields 01.05 as the result.

crdate

Creation date for this member. The format is *mm/dd/yy*.

chdate

Date of last modification. The format is *mm/dd/yy*.

chtime

Time of last modification. The format is *hh:mm*, or *hh:mm:ss* where *hh* ranges from 0 to 24, *mm* ranges from 0 to 59, and *ss* ranges from 0 to 59. *hh*, *mm*, and *ss* are whole numbers.

size

Current size, in lines (logical records) of the member. A whole number from 0 through 65535.

init

Size, in lines (logical records), of the member when it was created. This is a whole number in the range of 0 through 65535.

mod

Number of times this member has been modified. A whole number from 0 to 65535.

user

User ID of the user who last updated (or created) this member. This is a 7-character field (the eighth character, if any, is truncated), right-padded with blanks.

ENCODE

Prepares a 30-byte ISPF-compatible data field, using the provided data.

DECODE

Extracts data from a 30-byte ISPF compatible data field. The resulting fields are returned as the function's result, in blank-delimited form.

data

30-byte binary string, representing ISPF-compatible member statistics. The format is as follows, where the first piece of information represents the offset, the second represents the length, and the third describes the data:

0 - 1

Version, binary

1 - 1

Modification level, binary

2 - 1

Reserved

3 - 1

For ISPF 3.3 and later, the seconds portion of the modification time. Unsigned packed decimal.

4 - 4

Creation date: 00*yyddd*F. Julian, signed packed decimal.

8 - 4

Last changed date: 00*yyddd*F. Julian, signed packed decimal.

12 - 1

The hours portion of the modification time. Unsigned packed decimal.

13 - 1

The minutes portion of the modification time. Unsigned packed decimal.

14 - 2

The current number of records (lines) in the member. Unsigned binary.

16 - 2

The number of records (lines) in the member when created. Unsigned binary.

18 - 2

The number of times the member has been modified. Unsigned binary.

20 - 7

First 7 characters of the user ID responsible for the last alteration of the member.

27 - 3

Reserved. Three character blanks (X'40').

Return Codes

The $ISPFSTAT() function produces these return codes:

101 - 109

ARG *n* MISSING OR INVALID

121

DATA LENGTH NOT 28

122

CREATE DATE NOT PACKED DECIMAL

123

CREATE DATE NOT VALID JULIAN

124

CHANGE DATE NOT PACKED DECIMAL

125

CHANGE DATE NOT VALID JULIAN

126

CHANGE TIME NOT PACKED DECIMAL

127

BAD VV.MM

128

BAD CREATE DATE

129

BAD CHANGE DATE

130

BAD CHANGE TIME

131

BAD SIZE

132

BAD INIT SIZE

133

BAD MODIFICATION LEVEL

134

BAD USERID

Example

```
/* Locate member, display ISPF attributes */

    parse upper arg dsn member

    x = alloc(dsn,'shr')
    dcb = sam('obtain',,x,'pdsin')
    if rc ^= 0 then return 'obtain returns' rc '-' dcb
    x = sam('open',dcb)
    if rc ^= 0 then return 'open returns' rc '-' x
    data = sam('bldl',dcb,member)
    if rc ^= 0 then return 'bldl returns' rc '-' data
    flags = x2b(c2x(substr(data,14,1)))
    length = x2d(b2x(substr(flags,4,5)||'0'))
    if length < 28 then return 'ISPF statistics not available'
    if substr(flags,1,1) == '1' then type = 'ALIAS'
    else type = 'MEMBER'
    data = substr(data,15,30)
    x = $ispfstat('decode',data)
    parse var x vvmm create chdate chtime size init mod user
```

$JUL2GREG()

Use the $JUL2GREG() function to convert a Julian-format date to Gregorian format.

Syntax

The $JUL2GREG() function has this syntax:

```
gregdate = $JUL2GREG(juldate)
```

Arguments

The $JUL2GREG() function takes these arguments:

gregdate

Gregorian-format date: *mm/dd/yy* or *mm/dd/yyyy*. The number of digits assigned to the year will be the same as specified in the Julian date. The month and day fields will be two digits each, and will be left-padded with a zero, if needed.

juldate

Julian-format date: *yy.ddd* or *yyyy.ddd*. The number of digits assigned to the year will be carried over to the result and must be either two or four. The number of digits occupied by the day is unimportant except that it must represent a valid date.

Return Codes

The $JUL2GREG() function produces these return codes.

101

ARG 1 MISSING OR INVALID

121

INVALID YEAR

122

INVALID DAY

Example

```
$JUL2GREG('96.366')  ==  '12/31/96'
$GREG2JUL('2000.001') == '01/01/2000'
```

$SERVER()

Use the $SERVER() function to provide linkage to server IMODs. It also provides services to the server IMODs themselves, permitting easy implementation of the server IMOD conventions.

Syntax

The $SERVER() function has this syntax:

Form 1:

```
x = $SERVER(REQUEST,service,[arg],[control])
```

Form 2:

```
$SERVER(action,service,imod,[desc])
```

Form 3:

```
$SERVER(command,service,[option])
```

Form 4:

```
$SERVER(INTERNAL,[service],subcmd,[reqimid,reqstk],[retstk],[string])
```

Arguments

The $SERVER() function takes these arguments:

x

Result string from the server IMOD.

REQUEST

Queues a request for execution by a server IMOD.

service

Name of the service to be invoked.

arg

Argument string to be passed to the server IMOD. The content of this string depends on the service being requested.

control

Control option. Specify one of the following:

WAIT

Returns control after the request is processed.

NOWAIT

Queues the request and returns control immediately to the requestor.

action

Action to be taken. Specify one of the following:

DEFINE

Defines or redefines a new server IMOD to the system.

DELETE

Deletes an existing server IMOD definition. Deleting an operating server IMOD does not affect it in any way, except that its existence becomes unknown to the system for any future requests or commands.

imod

Name of the IMOD that is to be invoked to start the named server.

desc

Description of the service. This description is made available for reports and displays.

command

Command to be issued to the server IMOD. This may be one of the predefined server commands that are listed in "Server Commands" or another command that is specifically supported by the server.

option

Modifiers that affect how a command is executed by the server IMOD. The options are listed with the server commands in the following section.

INTERNAL

Indicates that the request deals with the internal operation of a server IMOD. INTERNAL requests are only made from within a server IMOD and provide a convenient mechanism for providing standard services.

subcmd

Tailors the INTERNAL service by the specification of subcommands that are listed in "INTERNAL Subcommands".

reqimod

IMOD ID of the IMOD task that originated a request.

reqstk

Stack number, where results of a request are to be returned.

retstk

Stack number that contains stack records to return. **Default:** Current stack.

string

String of data to be returned to the requesting IMOD task.

Server Commands

In addition to REQUEST, DEFINE and DELETE, a series of commands have been defined for operating server IMODs. The following is a list of the available commands, their effects, and options:

CANCEL

Cancels the server IMOD. This command should be avoided since the current request (and all queued requests) are abandoned and the requesting IMODs are *not* notified.

CYCLE

Cancels the current server IMOD and starts a new server. Queued requests are transferred to the new server.

DEQUEUE

Transfers all queued requests to the current stack of the IMOD task issuing the command.

DISPLAY

Returns information on all currently defined servers on the stack. The first character of each stack record serves as a delimiter character. The fields returned, in order, are Service, IMOD name, IMOD task ID (if running), and Description.

HOLD

Continues to permit requests to be queued, but does not permit processing of any requests from the queue.

PAUSE

Prevents acceptance of any further requests for queuing. Processing of already queued requests is deferred.

PURGE

Fails each request in the current request stack and notifies any waiting requestors of the failure.

QUIESCE

Prevents queuing of additional requests, but permits processing of any requests that have already been queued.

RESET

Deletes all unprocessed requests. No processing is performed and the requestors are *not* notified.

RESUME

Cancels the effects of any outstanding HOLD, PAUSE, or QUIESCE.

START

Starts the named service. Any specified *options* are passed to the server IMOD in the argument string, following the service name.

STATUS

Returns the status of a particular server IMOD on the stack. The information returned is formatted for display purposes.

STOP

Performs a QUIESCE function and then a TERMINATE function when all queued requests have been completed.

TERMINATE

Terminates processing immediately. Ignores any unprocessed requests.

TEST

Returns the current status of the server as the result. Values are UNDEFINED, STOPPED, STOPPING, PAUSED, HOLDING, QUIESCED, and AVAILABLE.

INTERNAL Subcommands

The INTERNAL service is used only from within a server IMOD. It permits easy implementation of server IMOD conventions and standardized coding. The following lists each available subcommand and any modifiers that are appropriate. For a detailed discussion and examples of writing server IMODs, see the server IMOD information in the *Administration Guide*.

AUTO

Intercepts all COMMAND functions and handles them without the knowledge of the server. The interception and execution are performed whenever the WAIT subcommand is issued. The effect of AUTO can be canceled with the NOAUTO subcommand.

COMPLETE

Posts a request as complete and returns the results to the requesting IMOD. This option requires specification of *reqimod* and *reqstk*. You can use *string* to return a result string.

HOLD

Prevents the processing of any additional requests. Requests continue to be queued. Commands continue to be processed.

NOAUTO

Cancels a previously issued AUTO subcommand.

PAUSE

Prevents queuing of additional requests. Processing of any already queued requests is deferred.

PURGE

Fails each request in current request stack and notifies waiting requestors. If *string* is specified, it is returned as the result. If omitted, the result will be: FAILED: REQUEST PURGED.

QUIESCE

Prevents further requests from being queued. The WAIT subcommand returns both commands and already queued requests.

RESET

Deletes all entries from the request stack. No processing is performed and waiting requestors are *not* notified. Extreme care should be used when using this option.

RESUME

Cancels the effects of any outstanding HOLD, PAUSE, or QUIESCE.

SETUP

Performs all functions necessary to establish a server IMOD. This service should be invoked once when the server IMOD task is first started.

TERMINATE

Terminates processing immediately. Any unprocessed requests are ignored and discarded.

WAIT

Waits for a command or request to be entered. The returned value is REQUEST (indicating that a request has been received), COMMAND (if a command is to be processed), TERMINATE (if immediate termination is to occur), or FAIL: *text* (if a failure has occurred).

Usage Notes

See the server IMOD information in the *Administration Guide* for additional information and examples of coding and using server IMODs.

Results

The $SERVER() function returns a variety of result strings, depending on the request. All requests may return a result in the form: FAIL: *text*, where *text* is descriptive of the condition causing the failure.

In lieu of specific results, the string SUCCESS is returned to the caller.

$TIME()

Use the $TIME() function to obtain the current time of day (TOD) clock value, add and subtract double-word unsigned binary values, and convert CPU timer values to printable strings.

Syntax

The $TIME() function has this syntax:

Form 1:

`$TIME(ADD,`*`value1`*`,`*`value2`*`)`

Form 2:

`$TIME(FULL,`*`value1`*`)`

Form 3:

`$TIME(FULLB,`*`value1`*`)`

Form 4:

`$TIME(SEC,`*`value1`*`)`

Form 5:

`$TIME(STCK)`

Form 6:

`$TIME(SUB,`*`value1`*`,`*`value2`*`)`

Arguments

The $TIME() function takes these arguments:

ADD

Adds *value1* to *value2*. The result is an unsigned 8-byte binary value in IBM TOD clock format.

SUB

Subtracts *value2* from *value1*. The result is an unsigned 8-byte binary value in IBM TOD clock format.

value1

8-byte binary string in IBM TOD clock format.

value2

8-byte binary string in IBM TOD clock format to be added to or subtracted from *value1*.

FULL

Converts *value1* to a printable string in the format *hh:mm:ss.tho* (*hh* is hours, *mm* is minutes, *ss* is seconds, and *tho* is thousandths of a second). The string is always 12 bytes in length and all leading zeros and field separators are present.

FULLB

Same as FULL, except that leading zeros and punctuation are replaced by blanks.

SEC

Converts *value1* to seconds. Returned number is in REXX format, contains a decimal point, and is accurate to thousandths of a second.

STCK

Returns the current value of the system TOD clock as an 8-byte unsigned binary value.

Return Codes

The $TIME() function produces these return codes:

101 - 103

ARG *n* MISSING OR INVALID

Example 1

```
/*  Report total CPU usage for an address space */

parse arg ascb                  /* address of ASCB is passed          */
cpu = memory('f',ascb+64,8)     /* get elapsed jobstep time (ASCBEJST */
string = $time('full',cpu)      /* convert to printable string        */
return string
```

Example 2

```
/*  Determine elapsed CPU time for subroutine  */

start = cpu('stck')             /* obtain start CPU value         */
call test_sub                   /* invoke subroutine to time      */
stop = cpu('stck')              /* obtain ending CPU value        */
total = $time('sub',stop,start) /* determine elapsed CPU time     */
string = $time('fullb',total)   /* make printable. no lead zeros. */
say 'Subroutine CPU: ' string
```

$TIMER()

Use the $TIMER() function to set one or more asynchronous timers. As each interval expires, a record is written to a stack to notify your IMOD.

Syntax

The $TIMER() function has this syntax:

Form 1:

```
id = $TIMER(SET,sec,stknum,[rec],[imodid])
```

Form 2:

```
id = $TIMER(CLEAR,id)
```

Arguments

The $TIMER() function takes these arguments:

SET

Establishes a new timer interval without affecting existing intervals.

CLEAR

Clears an existing interval.

id

Unique ID assigned by timing facility. If you are canceling an existing interval, you must specify this ID.

sec

Number of seconds in the interval.

stknum

Number of the stack that should be posted when the interval expires.

rec

1- to 256-byte argument placed on the stack when the interval expires. The default is TIME_EXPIRED.

imodid

IMOD that owns the stack that is being posted when interval expires.

Usage Notes

When an interval expires, the text in rec (or TIME_EXPIRED) is written to the stack, followed by a blank and the interval ID (id) returned by the SET argument.

Return Codes

The $TIMER() function produces these return codes:

101 - 105

ARG *n* MISSING OR INVALID

121

TIMER IS NOT AVAILABLE

122

SYSTEM SHUTDOWN IN EFFECT

123

STORAGE EXHAUSTED

Example

```
/* Set two timers */
id1 = $timer('set',10,1,'INTERVAL_1')
 id2 = $timer('set',60,1,'INTERVAL_2')
.../* start other processes */
x = wait(0,1,2) /* wait on multiple events)
 if x = 1 then do
    rec = pull(1)
    parse var rec string id .
    select
        when string = 'INTERVAL_1 then call expire_1
        when string = 'INTERVAL_2 then call expire_2
        otherwise signal timer_error
        end
    end
```

$WHEREFROM()

Use the $WHEREFROM() function to determine the IMOD and statement number responsible for calling the current IMOD. This is useful in debugging invalid calls to common routines.

Syntax

The $WHEREFROM() function has this syntax:

```
line imodname = $WHEREFROM()
```

Arguments

The $WHEREFROM() function takes these arguments:

line

Line number in the parent IMOD responsible for invoking the currently executing IMOD.

imodname

Name of the parent IMOD.

Return Codes

The $WHEREFROM() function produces these return codes:

120

NOT CALLED FROM IMOD

Example

```
  signal on syntax                   /* set trap */
  parse arg w1 w2 w3 .               /* begin processing */
  ...
  ...
 syntax:                              /* trap code */
  say 'Error' rc 'on line' sigl      /* show location of error */
  say 'IMOD called from $wherefrom() /* show source of call */
  return
```

ALIGN()

Use the ALIGN() function to return a numeric string that is padded on the right with blanks so that decimal points line up. You can also add commas to the integer portions of the number.

Syntax

The ALIGN() function has this syntax:

```
ALIGN(value,right,[COMMAS])
```

Arguments

The ALIGN() function takes these arguments:

value

Numeric input string.

right

Number indicating the number of positions to the right of the decimal point. Trailing blanks are added and truncation can occur.

COMMAS

Inserts commas in the integer portion of the number.

Return Codes

The ALIGN() function produces these return codes:

101 - 103

ARG *n* MISSING OR INVALID

Example

```
align('100',2)           == '100. ' /* Pad with 2 blanks */
align('1000',0,'COMMAS') == '1,000.'/* Add commas   */
```

ALLOC()

Use the ALLOC() function to dynamically allocate new or existing data sets.

Syntax

The ALLOC() function has this syntax:

```
ALLOC([dsn],[disp1[disp2[disp3]],[PERM],[dd],[unit],[volser],
  [,space prim [sec] [dir],[dsorg],[recfm],[lrecl],[blksize]) = ddname
```

Arguments

The ALLOC() function takes these arguments.

ddname

DDname for the newly allocated file (or error text if the allocation fails).

Default: The system assigns a ddname that is returned by the ALLOC() function.

dsn

Name of the data set you want to allocate. To specify a PDS member, include the member in parentheses (for example, INDEX.NAME(MEMBER)). The member is allocated as a sequential file.

Default: The system assigns a temporary name.

disp1

How the file should be allocated to you. Specify one of the following:

SHR

Allocate an existing data set non-exclusively.

OLD

Allocate an existing data set for the exclusive use of the ISERVE address space. Other IMODs may also hold the same data set exclusively.

NEW

Allocate a new data set that does not already exist.

MOD

Allocate a new or existing data set for the exclusive use of the ISERVE address space. The "write pointer" is
positioned to the end of the data set so that you may extend the data set with additional records.

disp2

What should be done with the data set when it is deallocated normally. Specify one of the following:

KEEP

Keep it, but do not catalog it. This is the default if it was an existing data set that was allocated.

DELETE

Delete it. If the data set was allocated through the catalog (without specification of volume serial number),
it will also be uncataloged. This is the default for a newly allocated data set.

CATLG

Keep it and catalog it.

UNCATLG

Keep it and uncatalog it. For example, 'NEW CATLG DELETE' is equivalent to the JCL statement DISP=(NEW,CATLG,DELETE). For default values, consult your JCL reference
manual.

disp3

Data set's disposition if it is deallocated during abend processing. Specify one of the values listed in *disp2*.

PERM

Indicates that the data set should remain allocated after it is closed.

Default: FREE=CLOSE

dd

DDname for the file.

Default: The system generates a unique DDname.

unit

Unit name for the allocation. This argument is required when allocating a new or uncataloged data set.

volser

Volume to be allocated. This argument is required when allocating an uncataloged data set. It is also required if you are allocating a new data set and you are *not* using an esoteric value for the unit.

space

How much space should be allocated to a disk data set. This value is required for new data sets and ignored for existing data sets. Specify one of these values:

blocks

The size of the disk block to be allocated. This is generally the same value as the block size of the
data set. The allocation routines calculate the physical amount of disk space required, based upon
blocks of the specified size. If specified, both *prim* and *sec* are assumed to be in blocks.

CYL

The allocation is to be by cylinder. If used, both *prim* and *sec* are assumed to specify cylinder

TRK

The allocation is to be by track. If used, both *prim* and *sec* are assumed to specify tracks.

prim

Minimum amount of space for allocation (in blocks, tracks, or cylinders).

sec

Space to be allocated each time the data set is full and additional data is to be written (in blocks, tracks, or cylinders).

A data set may extend over 16 discrete areas on a single disk volume.

Default: 0

dir

Number of 256-byte directory blocks to be allocated to a partitioned data set (PDS). You must specify *dir* if you are allocating a PDS; otherwise, omit it.

dsorg

Data set organization. Specify one of the following: PS, PSU, PO, or POU.

Default: No value is assigned (for new data sets), or the *dsorg* specified in the data set label is used (for existing data sets).

recfm

Record format. Specify one of the following: F, FA, FM, FB, FBA, FBM, V, VA, VM, VB, VBA, VBM, or U.

Default: No value is assigned (for new data sets), or the record format specified in the data set label is used (for existing data sets).

lrecl

For a new data set, you may specify the logical record length. Specify a value from 0 to 32,769 (inclusive).

Default: No value is assigned.

blksize

For a new data set, you may specify the block size. Specify a value from 0 to 32,769 (inclusive).

Default: No value is assigned.

Usage Notes

Allocating a data set that requires operator intervention suspends the requesting IMOD task until the request can be satisfied.

- Unless PERM is specified, all data sets allocated by an IMOD task are freed at the termination of the IMOD task.
- Both allocating and deallocating data sets generates a message in ISRVLOG.

Return Codes

The ALLOC() function produces these return codes:

101 - 111

ARG *n* MISSING OR INVALID

102

ARG 2, SUBARG *n* INVALID

104

DDNAME ALREADY IN USE

121

error info text

error SVC 99 Error code (hexadecimal)

info SVC 99 Information code (hexadecimal)

text Explanatory text

Example

```
alloc('RBROWNE.TEST.DATA','SHR',,''JUNK') == 'JUNK'
         /* Allocate a PDS, alloc returns ddname */
         /* Note that member name is not specified until SAM() */
```

BATCHRC()

In batch mode (under SRVBATCH), the BATCHRC() function allows an IMOD to set or reset the value that is returned to the operating system as the JOBSTEP completion code. This value may be tested in the job's JCL to permit conditional execution of further steps.

Syntax

The BATCHRC() function has this syntax:

```
value = BATCHRC([number])
```

Arguments

The BATCHRC() function takes these arguments:

value

Value that is currently set as the JOBSTEP completion code.

number

Value that replaces any former value and returns to the system as the JOBSTEP completion code.

Default: The existing value is returned but is not changed.

Usage Note

BATCHRC() is valid only when the IMOD is executing in batch mode, under SRVBATCH.

Return Codes

The BATCHRC() function produces these return codes:

101

ARG 1 MISSING OR INVALID

Example

```
/* Set a return code only if RC is higher than the current
   return code */

   newret = batchrc(max(rc,batchrc()))
   if newret > 8 then exit
```

BOOLWORD()

Use the BOOLWORD() function to perform a Boolean operation on two numeric values and to return the numeric result.

Syntax

The BOOLWORD() function has this syntax:

```
result = BOOLWORD({AND},value,value)
                  {OR}
                  {XOR}
```

Arguments

The BOOLWORD() function takes these arguments:

result

Numeric result of the operation.

AND

Perform a Boolean AND function.

OR

Perform a Boolean OR function.

XOR

Perform a Boolean exclusive OR function.

value

Numeric value in character-string format.

Return Codes

The BOOLWORD() function produces these return codes:

101 - 103

ARG *n* MISSING OR INVALID

Example

```
boolword('OR','020','001') == '21' /* Leading zeros are ignored  */
boolword('OR',2,1)         == '3'
boolword('AND','2','1')    == '0'
```

CALC()

Use the CALC() function to evaluate a complex numerical expression and return the results.

Syntax

The CALC() function has this syntax:

result = CALC(expression,[decimal])

Arguments

The CALC() function takes these arguments:

result

Numeric result of the operation.

expression

Mathematical expression to be evaluated. Numbers can be up to six significant digits, containing a decimal point and leading plus sign or leading minus sign if needed. Blanks are ignored. The operators you can use are shown in the following section.

Because CALC() uses true floating-point arithmetic, values can be imprecise. Very large and very small numbers include a scaling factor (power of 10), and return values in the following form:

1.23456 * 10 ** 25

decimal

Number of digits to the right of the decimal point.

Default: All significant digits appear, but trailing zeros are omitted.

Operators for the Expression Operand

These operators can be used within the expression:

+

Plus

-

Minus

******

Raise to a power

*****

Multiply

/

Divide

(

Start priority grouping

)

End priority grouping

sin()

Evaluate sine

cos()

Evaluate cosine

atan()

Evaluate arc tangent

exp()

Exponential

abs()

Absolute value

sqrt()

Square root

log10()

Base 10 logarithm

log()

Base 10 logarithm

ln()

Base *e* (natural) logarithm

Return Codes

The CALC() function produces these return codes:

101

One of the following:

ARG 1 MISSING OR INVALID

CHARACTER IN ARG 1

102

ARG 2 INVALID

121

One of the following:

EXPECTED OPERATOR MISSING

TOO MANY NUMERIC FIELDS

OPERATOR FOLLOWS OPERATOR

INVALID OPERATOR

NUMBER FOLLOWS RIGHT PARENTHESIS

UNBALANCED PARENTHESES

OVERFLOW OR UNDERFLOW

EXPRESSION TOO COMPLEX

ERROR DURING SIN/COS EVALUATION

ERROR DURING EXP EVALUATION

ERROR DURING LOG EVALUATION

ERROR DURING SQRT EVALUATION

RESULT TOO LARGE

DIVISION BY ZERO

Example

```
calc('1+1')         == '2'
calc('(1+2)*5')     == '15'         /* Not 11   */
calc('abs(-9)')     == '9'
```

CALLX()

Use the CALLX() function to execute an IMOD as a subroutine at another GoalNet node.

Syntax

The CALLX() function has this syntax:

```
result = CALLX([node],[ssid],imod,{NONE|PASS},[args])
```

Arguments

The CALLX() function takes these arguments:

result

Value specified by the last-executed REXX RETURN or EXIT instruction in the called routine (or error text).

node

Name of the GoalNet node where the IMOD is to be executed. The current node is assumed as the default.

ssid

Subsystem ID that identifies the ISERVE that is to execute the IMOD.

If *ssid* is specified without *node*, the request is routed to the appropriate ISERVE on the same z/OS system. If *node is also specified*, the request is first routed to the requested GoalNet node and then sent to the appropriate ISERVE on that z/OS system.

imod

Name of the IMOD that is to be executed as an external routine. This argument is required.

NONE

Indicates that no stack information is to be passed to the called routine. This is the default. Any returned stack data is appended to the current stack contents. Select this option whenever records exist on the current stack and the called IMOD is not going to process them.

PASS

Passes the contents of the current stack (up to the NEWSTACK marker) to the called routine. When the information is passed, it is deleted from the caller's stack. Upon return, any records left on the stack by the called IMOD are returned to the caller. These records become the new stack contents.

args

String passed to the called routine as an argument string.

Usage Notes

- CALLX() operates like a standard REXX CALL instruction. An argument string can be passed, a result string returned, and the current stack can be passed to the called routine and the modified stack returned to the caller.
- Only a single argument can be passed, although this argument can have many fields.
- Only the passed stack can be referenced by the called IMOD. If the called IMOD creates additional stacks using the SWAPSTAK() or NEWSTACK() functions, only the current stack is returned to the caller at the conclusion of the called IMOD.

Return Codes

The CALLX() function produces these return codes:

101 - 105

ARG *n* MISSING OR INVALID

121

REQUESTED NODE NOT DEFINED

122

REQUESTED NODE NOT ACTIVE

123

REQUESTED NODE IS DOWN

124

REQUESTED NODE IS STOPPED

125

STATUS OF REQUESTED NODE IS INDETERMINATE

128

UNABLE TO ESTABLISH CONVERSATION WITH NODE

129

NODE LOST BEFORE REMOTE IMOD SCHEDULED

131

NODE LOST AFTER REMOTE IMOD SCHEDULED

133

HALT/FAILURE/ERROR/SYNTAX AT line

134

IMOD NOT FOUND

135

IMOD NOT AVAILABLE

136

IMOD NOT CALLABLE

138

NO CONNECTION AVAILABLE

150

One of these messages is returned:

GSS COMMUNICATION FAILURE return reason

IMOD was canceled during execution

IMOD was not found

IMOD was found but was deactivated

IMOD cannot be called from current GSS environment

IMOD could not be executed

ISERVE address space ABENDED during IMOD processing

IMOD Recovery has failed

Syntax error in IMOD recovery request

Invalid plist, args not in valid storage

Requested GSS service not available

Invalid GSS Plist, invalid storage subpool

JESCT has invalid format

SSCVT has invalid format

Requested GSS service is invalid

Requested ISERVE address space not valid

ISERVE SUBCOM area not found

GSS SSCVT not initialized

GSS Address Space is not active

Requested ISERVE address space not running

GSS SSCVT not found

GSS PC table not found

Insufficient storage for GSS X-MEM plist

Insufficient storage for XQAREA

PC table entry not found

PC routine not found

Requested GSS service is invalid

Insufficient storage, service dynamic area

Insufficient storage, external XQAREA

Insufficient storage, PC2 dynamic area

Invalid ASCB address, PC2

Value invalid

Input STACK format is invalid

Invalid PLIST, "GSST" is missing

GSSCVT missing from PLIST

GSSTAB1 has invalid format

PLIST not accessible

Example

```
callx('node2',,'status',,args)  /* Run "status" IMOD at node2 */
```

CASE()

Use the CASE() function to convert a string to uppercase or lowercase.

Syntax

The CASE() function has this syntax:

```
CASE({UPPER|LOWER},string)
```

Arguments

The CASE() function takes these arguments:

UPPER

Translates to uppercase.

LOWER

Translates to lowercase.

string

1- to 256-character input string.

Return Codes

The CASE() function produces these return codes:

101 - 102

ARG *n* MISSING OR INVALID

Example

```
case('lower','ABC') == 'abc'   /* Translate a string to lowercase */
case('upper','abc') == 'ABC'   /* Translate a string to uppercase */
```

CP()

Use the CP() function to issue a VM CP command and return a CP's response. z/OS must be operating under VM.

Syntax

The CP() function has this syntax:

```
result = CP(command,[stem.])
```

Arguments

The CP() function takes these arguments:

result

Error or completion status.

command

Command string to be passed to CP.

Note: VM systems are case-sensitive.

stem

Name of a stem variable in which individual lines of CP's response are returned. Include the period. The first line of the response returns in *stem*.1, the second line in *stem*.2, and so forth without breaks.

The stem variable cannot be global (that is, it cannot begin with an ampersand).

Default: Results are returned on the stack and can be accessed with PULL() or a similar function.

Return Codes

The CP() function produces these return codes:

0

COMMAND EXECUTED

101

ARG 1 MISSING OR INVALID

102

One of the following:

ARG 2 MISSING OR INVALID

ARG 2 NOT STEM VARIABLE

120

NOT UNDER VM

121

RESPONSE TOO LONG

122

PASSWORD REQUIRED

123

NO RESPONSE FROM VM

124

INSUFFICIENT STORAGE

Example

```
cp('MSG RBROWNE testing') == 0    /* RBROWNE would get a message  */
cp('Q MVSB1') == 0                /* 'MVSB1 - DSC' might be put in queue */
```

CPUTIME()

Use the CPUTIME() function to enable an IMOD task to determine how much CPU time it has consumed.

Syntax

The CPUTIME() function has this syntax:

```
{value|string|number} = CPUTIME({STCK|FULL[B]|SEC})
```

Arguments

The CPUTIME() function takes these arguments:

value

Unsigned 64-bit binary value in IBM TOD clock format.

string

12-character value, in the format *hh:mm:ss.tho*.

number

Numeric value suitable for REXX computational use.

STCK

Returns the elapsed CPU time as an unsigned 64-bit number.

FULL

Returns the elapsed CPU time as a printable string. FULL is the default. Specifying FULLB removes all leading zeros and associated separators. In each case, 12 characters are returned in the format *hh:mm:ss.tho*.

SEC

Returns the elapsed CPU time, in seconds, as a REXX numeric value. The value is accurate to three decimal places.

Return Codes

The CPUTIME() function produces these return codes:

101

ARG 1 MISSING OR INVALID

Usage Notes

The CPUTIME value includes REXX interpretation time and time used by ADDRESS environments that execute in the same address space as the IMOD. Time accrued in other address spaces is not considered.

Example

Consider this example:

```
/* Display the elapsed CPU time for an IMOD */

   time = cputime('fullb')
   time = strip(time,'l')      /* Remove leading blank characters */
   say 'CPU Time for' imod_id 'is' time
```

D2P()

Use the D2P() function to convert a REXX-format whole number into IBM packed-decimal format. This form is frequently used to store decimal numbers with two digits per byte.

Syntax

The D2P() function has this syntax:

pdec = D2P(*number*,*bytes*,[*resultform*])

Arguments

The D2P() function takes these arguments:

pdec

Packed decimal number. Each byte contains two decimal digits, except the right-most byte, which may contain a sign digit (hexadecimal F for positive, B for negative) in the right half of the byte. The presence of the sign is controlled by the third argument.

number

Whole number. You can include a leading sign.

bytes

Number of bytes to be occupied by the returned packed-decimal number. Specify a value from 1 to 8 (inclusive). Truncation and padding are performed on the left.

resultform

Result's format. Specify one of the following:

SIGN

Indicates that a hexadecimal F (positive) or B (negative) occupies the right-most four bits of the result.

NOSIGN

Indicates that the packed-decimal number occupies the entire result string and no indication of sign is present.

Default: SIGN

Return Codes

The D2P() function produces these return codes:

101 - 103

ARG *n* MISSING OR INVALID

Example

```
/*   Pack a Julian date to store in ISPF statistics format */
/*   date as year and day */
parse arg year '.' day
/* Standard form:yyddd*/
julian = right(year,2,'0')||right(day,3,'0')
 /* packed format (4 bytes): 00 yy dd dF*/
pdec = d2p(julian,4,'sign')
```

DASD()

Use the DASD() function to return information about specified DASD devices.

Syntax

The DASD() function has this syntax:

Form 1:

```
space = DASD(SPACE,{VOLUME,volser})
                   {UNIT,unitname}
```

Form 2:

```
unitname = DASD(VOL2UNIT,volser)
```

Form 3:

```
volser = DASD(UNIT2VOL,unitname)
```

Form 4:

```
ucb = DASD(UCB,{VOLUME,volser})
               {UNIT,unitname}
```

Arguments

The DASD() function takes these arguments:

space

Amount of free space on the DASD. The result string has the following format:

SPACE *fcyl ftrk ext lcyl ltrk*

fcyl Total free cylinders

ftrk Total free tracks on partial cylinders

ext Total free extents

lcyl Number of cylinders in largest contiguous group

ltrk Number of free tracks contiguous with largest free area

SPACE

Returns information on the amount of free space remaining on the volume or device.

VOLUME

Indicates that the third argument specifies a volume serial number.

volser

Volume serial number (volume label) of a DASD device.

UNIT

Indicates that the third argument specifies a unit address.

unitname

Numeric unit name assigned by the operating system. This is the decimal equivalent of the value.

VOL2UNIT

Returns the physical device address (unit address) of the device bearing the specified volume serial number (volume label).

UNIT2VOL

Returns the volume serial number (volume label) associated with the specified physical device address.

ucb

Memory address of the unit control block (UCB) assigned to the DASD device. The address is returned as a decimal number.

UCB

Returns the memory address of the unit control block assigned to the volume or device.

Return Codes

The DASD() function produces these return codes:

101 - 103

ARG *n* MISSING OR INVALID

121

SPECIFIED VOLUME NOT FOUND

122

UNIT ADDRESS NOT VALID

Example

Consider this example:

```
unit   = dasd('VOL2UNIT','mvs101')        /* Determine unit name
                                             containing MVS101 */
volser = dasd('UNIT2VOL',x2d(152))        /* Determine volser
                                             currently on unit 152 */
ucb    = dasd('UCB','UNIT',x2d(152))      /* Obtain memory address of
                                             UCB for unit 152 */
ucb    = dasd('UCB','VOLUME','mvs101')    /* Obtain memory address of
                                             UCB for volume MVS101 */
space  = dasd('SPACE','UNIT',x2d(152))    /* Determine free space on
                                             unit 152 */
space  = dasd('SPACE','VOLUME','mvs101')  /* Determine free space on
                                             volume MVS101 */
```

DB2()

Use the DB2() function to enable an IMOD to dynamically perform SQL SELECT statements and retrieve the selected rows.

Syntax

The DB2() function has this syntax:

Form 1:

result = DB2(OPEN,[*plan*])

Form 2:

result = DB2(SQL,*stmt*,[*maxrow*,[*descsep*],[OFFSET])

Form 3:

result = DB2(MORE,[*maxrow*])

Form 4:

result = DB2(CLOSE,[*option*])

Arguments

The DB2() function takes these arguments:

result

Result of an SQL request will return:

The null string, if no errors occurred.

The DB2 error code, followed by possible error text, if an error occurred that prevented execution of the request.

The text string MORE if a SELECT request successfully returned data but could not complete due to the *maxrow* limit.

OPEN

Opens a plan.

plan

Name of the DB2 plan to be used for this request. The default is GSSPLAN (which is distributed in the CA-GSS Example library and which must be linked to the DB2 driver that should service the requests).

SQL

Causes execution of the specified SELECT statement.

stmt

DB2 SELECT statement.

maxrow

Maximum table rows to be returned on the stack. Additional rows may be fetched with the MORE operation.

Default: 500

descsep

Single character to be used as a separator for the field descriptions. The first stack record returned will be the Field Description Names. The *descsep* character will be used to delimit the name.

Default: Field Description Names are not returned on the stack.

OFFSET

Generates output of a stack record consisting of pairs of 2-byte binary fields (immediately preceding the first row). The first field is the offset (0-based) to a field in each following row. The second field is the length of the data portion of the field.

MORE

Requests the return of additional rows from the previously issued SELECT statement.

CLOSE

Closes the plan and either commits or aborts changes.

option

How the close operation should process. Specify one of the following:

SYNC Normal

ABRT Abort previous processing

Neither option is applicable for SELECT processing.

Usage Notes

When a SELECT is successful, individual rows will be returned on the stack, up to the limit specified by maxrow. If descsep was specified, the first record returned is the field descriptor names. The names are separated by the descsep character. If OFFSET was specified, the next record will consist of pairs of 2-byte binary fields. Each pair describes one field in the following returned rows. The first pair is the offset in the record to the start of the field information. The second pair is the total length required for the field. You use these pairs to index into any data record to find a particular field.

Each of the remaining data records returns one selected row from the DB2 table. The first two bytes of each field is the 2-byte DB2 indicator field. The field contents immediately follow the indicator field.

Return Codes

The DB2() function produces these return codes:

101 - 105

ARG *n* MISSING OR INVALID

121

OPEN NOT FIRST OR NOT AFTER CLOSE

122

CLOSE NOT AFTER OPEN OR SQL

123

SQL AFTER CLOSE

124

MORE NOT AFTER SQL

125

DB2 error codes and messages

After OPEN or CLOSE:

rrrrrrrr ccccccc text

rrrrrrrr is the hexadecimal return code, *cccccccc* is the hexadecimal reason code, and *text* is the error text.

After SQL or MORE:

nnnn p text

nnnn is the signed numeric DB2 error code, *p* is the phase indicator (internal to CA-GSS), and *text* is the DB2-provided error text.

126

UNABLE TO PROVIDE SUBTASK

127

NO MORE ROWS AVAILABLE

Example

```
/* Select and retrieve rows. Locate the third column and print it.*/
 result = db2('open')                /* open the plan              */
if rc ^= 0 then signal error1       /* -> if plan fails to open   */
result = db2('SQL','SELECT askdfjakf',0,' ','OFFSET')
 if rc ^= 0 then signal error2       /* -> if select fails         */
parse pull . . header .             /* Extract field name         */
parse pull offset                   /* obtain offset record       */
pos = c2d(substr(offset,8,2)) + 1   /* position of third field    */
len = c2d(substr(offset,10,2)) - 2  /* length of data portion
                                        of third field            */
say header                          /* print the field header     */
do while result = 'MORE'
  limit = queued()
   do i = 1 to limit
     parse pull row
     field3 = substr(row,pos,len)
     say field3
   end i
  result = db2('MORE',50)           /* get next 50 rows           */
 end
```

DEVTYPE()

Use the DEVTYPE() function to issue the IBM macro DEVTYPE SVC (with the DEVTAB and RPS options) and return the 24-byte binary results.

Syntax

The DEVTYPE() function has this syntax:

data = DEVTYPE(*ddname*)

Arguments

The DEVTYPE() function takes these arguments:

data

24 bytes of data or error text. The format of the returned data is that of the DEVTYPE SVC with DEVTAB and RPS options specified. Further information may be found in the IBM manual, *System Data Administration* (GC26-4010).

ddname

DDname on which the DEVTYPE is to be issued.

Return Codes

The DEVTYPE() function produces these return codes.

101

ARG 1 MISSING OR INVALID

120

DDNAME NOT FOUND

Example

```
x = devtype('SYSUT1')       /* Check for allocated file        */
if rc = 120 then do         /* Perform if DDNAME not allocated */
    say 'SYSUT1 not allocated'
    end
```

DEALLOC()

Use the DEALLOC() function to dynamically deallocate a specified data set.

Syntax

The DEALLOC() function has this syntax:

```
DEALLOC(ddname)
```

Argument

The DEALLOC() function takes this argument:

ddname

DDname of the data set you want to deallocate.

Usage Notes

Dynamically allocated data sets are automatically deallocated at the end of the IMOD task that allocated them, unless ALLOC() specified the PERM option.

- Both allocating and deallocating data sets generates a message in ISRVLOG.

Return Codes

The DEALLOC() function produces these return codes:

101

ARG 1 MISSING OR INVALID

121

error info text

error SVC 99 error code (hexadecimal)

info SVC 99 information code (hexadecimal)

text explanatory text

Example

```
dealloc('SYSUT1') == '' /* Free a previously allocated data set*/
```

DOM()

Use the DOM() function to delete a previously displayed WTO or MLWTO message from the operator's console. This function is particularly useful for messages issued with non-rollable attributes.

Syntax

The DOM() function has this syntax:

result = DOM(*msgid*)

Arguments

The DOM() function takes these arguments:

result

Null string (or error text).

msgid

Message ID as returned by the WTO() or MLWTO() function.

Return Codes

The DOM() function produces these return codes:

101

ARG 1 MISSING OR INVALID

Example

```
msgid = wto('Message')       /* Place message on console     */
x     = dom(msgid)           /* Remove message from console  */
```

DS3270()

Use the DS3270() function to build and decompose IBM 3270 data streams.

Syntax

The DS3270() function has this syntax:

Form 1:

```
string = DS3270(TO,{RA,[trows],[tcols],row,col,[char]})
                   {SBA,[trows],[tcols],row,col}
                   {SF,[attributes]}
```

Form 2:

```
rowcol = DS3270(FROM,SBA,[trows],[tcols],sba)
```

Form 3:

```
aid = DS3270(FROM,PARSE,[trows],[tcols],image)
```

Form 4:

```
text = DS3270(FROM,FIELD,[trows],[tcols],row,col,image)
```

Arguments

The DS3270() function takes these arguments:

string

3270 data.

TO

Converts to a 3270 data stream.

SBA

Produces a three-byte SBA sequence.

RA

Produces a four-byte RA sequence (if *char* is not specified, only 3 bytes are generated).

trows

Total rows on the display screen.

tcols

Total columns on the display screen.

row

Specific row position, 1-based.

col

Specific column position, 1-based.

char

Single character to be repeated to the screen address specified by *row, col*. This is only valid for an RA operation. The character will be repeated up to but not into the screen position specified by *row, col*.

SF

Produces a 2-byte SF sequence.

attributes

One or more of the following attributes:

HIGH

Displays the field in high intensity.

MDT

Modified data tag. When set, the field is returned as input. Note that this value can be reset by the terminal operator for unprotected fields, but protected fields cannot be altered by the operator or be prevented from being transmitted.

NODISP

Makes the field invisible on the screen.

NUM

For an input field, only numeric data can be physically entered.

PEN

Makes the field pen-selectable.

PROT

Protects the field.

SKIP

During the course of data entry, when the cursor moves onto a SKIP attribute (protected field only), the cursor immediately skips to the next input field.

Default:The field is available for terminal operator input.

Return Codes

The DS3270() function produces these return codes:

101 - 107

ARG *n* MISSING OR INVALID

121

FIELD NOT PRESENT

Example

```
/* This Example processes a logon screen and extracts the input */

input = lu2('sendrec',rpl,screen)
 if rc ^= 0 then signal error_lu2
x = ds3270('from','parse',,,input)
 parse upper var x aid .
 if aid = 'PF3' then exit
limit = queued()
if limit ^= 6  then do
    say 'Invalid field count on logon screen'
    return
    end
parse pull row.1 col.1        /* get coordinates of user ID  */
parse upper pull userid .
 parse pull row.2 col.2        /* get coordinates of password */
parse upper pull password .
 parse pull row.3 col.3        /* get coordinates of program  */
parse upper pull program .

return
```

DSNENQ()

Use the DSNENQ() function to return information about a data set's enqueue status in a series of predefined variables.

To use the DSNENQ() function

1. Obtain a handle. This handle is required for all subsequent operations.
2. Retrieve enqueue information.
3. Release the handle, thereby freeing storage.

Syntax

The DSNENQ() function has this syntax:

```
{handle|info} = DSNENQ({OBTAIN},{[pages]},dsname)
                       {INFO}   {handle}
                       {RELEASE}{handle}
```

Arguments

The DSNENQ() function takes these arguments:

info

Contains the retrieved enqueue information.

handle

Numeric ID for an internal workspace. The handle must be passed with each subsequent call. The same handle can be reused for different data sets.

INFO

Returns data set information. The table in the following section lists the variables and values returned by a request using the INFO argument.

OBTAIN

Obtains a new value for the handle argument, which is the address of the control block defining the data set.

RELEASE

Returns a handle value to the system to free the storage used. This is done automatically when the IMOD completes.

pages

Number of 4096-byte work areas to be made available to the z/OS GQSCAN service. The work area is discarded after the function is executed. Specify a value from 1 to 5 (inclusive).

If your system regularly provides enqueues for a very large number of data sets you may need to increase this value.

Default: 1

dsname

Data set name about which information is retrieved.

Variables Returned by the INFO Argument

The following table lists the variables returned by the INFO argument. For stem variables, the stem index n begins with 1 and increases without breaks to dsn_count.

dsn_dsname

Name of the data set on which enqueue information is being retrieved.

dsn_count

Number of entries returned. This value is the maximum value of *n*.

dsn_asid.n

ASID for entry *n*.

dsn_sysid.n

GRS (Global Resource Serialization) system name for entry *n*.

dsn_status.n

Status for entry *n*, either OWN or WAIT.

dsn_mode.n

Mode for entry *n*, either SHR (shared) or EXCL (exclusive).

dsn_jobtype.n

For entry *n*, JOB, TSU, STC, or EXTERNAL (unknown type or owned by another CPU).

Return Codes

The DSNENQ() function produces these return codes:

101 - 103

ARG *n* MISSING OR INVALID

121

NO OWNERS OF DATASET

124

Return code from GQSCAN service

123

INVALID HANDLE

Example

```
handle = dsnenq('OBTAIN')                  /* Get a new handle */
dsnenq('INFO',handle,'SYS1.HASPACE') == '' /* 121 if no owner  */
dsn_count == 1                             /* One owner found  */
```

ENQUEUE()

Use the ENQUEUE() function to serialize the use of resource names among multiple IMOD tasks.

Syntax

The ENQUEUE() function has this syntax:

ENQUEUE({ENQ|DEQ},,{COND|UNCOND|TEST},{EXCL|CONT|SHR},*qname*,*rname*,[*shrcount*])

Arguments

The ENQUEUE() function takes these arguments:

ENQ

Reserves control of a resource.

DEQ

Releases control of a resource.

COND

Indicates a conditional request. Returns control to the IMOD immediately. You must check the return code to find if you have gained control or if another IMOD's ownership of the resource has prevented it.

UNCOND

Indicates an unconditional request. If the requested resource is not immediately available, your IMOD is placed in a wait state until the resource becomes available. When return is made, you have control of the resource (or an error condition was detected).

TEST

Tests the specified request. Returns control immediately, and the results indicate whether or not the request would have succeeded.

EXCL

Requests an exclusive enqueue. No other users may hold an enqueue of any type for the resource.

CONT

Requests an enqueue for control of the resource. Only one CONT requestor may hold a resource, but multiple SHR users may hold an enqueue simultaneously with the single CONT user.

SHR

Requests a shared enqueue. Any number of requestors may hold a SHR enqueue simultaneously.

qname

1- to 8-character name (case-sensitive), which is used to group requests. Qnames (except as assigned by CA) should not begin with a dollar sign ($).

rname

1- to 256-character name (case-sensitive), which is used to fully define a resource within a qname group. Rnames (except as assigned by CA) should not begin with a dollar sign ($).

shrcount

For shared enqueues, the maximum number of users that can hold the enqueue simultaneously. There is no enforcement of one user's limit on another user. Therefore, all requestors of a specific enqueue should specify the same value for *shrcount*.

Default: 0 (no limit)

Usage Notes

- The following indicates what action will be taken for each type of request, based upon the resource's current ownership status.

Request	Current "Owners"	Enqueue Assigned?
SHR	none	Yes
	SHR*	Yes
	CONTROL	Yes
	CONTROL,SHR	Yes
	EXCL	No
CONTROL	none	Yes
	SHR	Yes
	CONTROL	No
	CONTROL,SHR	No
	EXCL	No

Request	Current "Owners"	Enqueue Assigned?
EXCL	none	Yes
	SHR	No
	CONTROL	No
	CONTROL,SHR	No
	EXCL	No

* Multiple SHR owners can always be regarded as a single entity.

- Resource names are arbitrary. The only protection afforded by ENQUEUE() is management of the names. Just because you assign a name and enqueue on it does not ensure protection of a resource. Any IMOD is free to ignore an enqueue scheme. To be successful, all IMODs must accept and abide by the same naming and access conventions.
- When an owned resource is released and IMODs are waiting, ownership is assigned in the following order:
  1. The EXCL requestor who has waited longest. This assignment is made only if no other IMOD has ownership.
  2. The CONT requestor who has waited longest, along with all waiting SHR requestors. This assignment is made only if no EXCL or CONT requestors already own the resource.
  3. All SHR requestors, provided there are no existing EXCL or CONT owners.
- When an IMOD task ends, all enqueues assigned to it are automatically released.

Return Codes

The ENQUEUE() function produces these return codes.

0

If ENQ or DEQ and *option* is TEST:

null (indicates request will succeed immediately)

REQUEST FOR ALREADY OWNED RESOURCE

RESOURCE NOT AVAILABLE

RESOURCE NOT OWNED

If ENQ and *option* is NOT TEST:

null (indicates resource now owned)

If DEQ and *option* is NOT TEST:

null (indicates owned resource successfully released)

101 - 106

ARG *n* MISSING OR INVALID

122

REQUEST FOR ALREADY OWNED RESOURCE

123

RESOURCE NOT AVAILABLE

124

RESOURCE NOT OWNED

125

ENQUEUE REQUEST CANCELED

(Although you requested an unconditional enqueue, your wait was terminated without you being assigned ownership.)

Example

```
/* Protect a pool of global (&) variables against multiple user
   updates. The qname of GLOBAL and rname of IMSGROUP are
   arbitrary, but are understood by all IMODs that will access
   the variables in question. */

   x = enqueue('enq',,'uncond','excl','GLOBAL','IMSGROUP')
   if rc ^= 0 then signal error1
   &ims_count = &ims_count + 1

/* Following additional processing, the enqueue will be
   automatically released when the IMOD task ends.  */
```

ILOG()

Use the ILOG() function to return information from an ILOG subfile.

Syntax

The ILOG function has this syntax:

Form 1:

handle = ILOG(OBTAIN)

Form 2:

ILOG(POINT,*handle*,*lognum*,*subfile*,*recnum*,[*direction*])

Form 3:

record = ILOG(GET,*handle*)

Form 4:

ILOG(RELEASE,*handle*)

Arguments

The ILOG() function takes these arguments:

handle

Numeric ID for an internal workspace. A handle must be obtained and then passed to the function with each subsequent call.

OBTAIN

Obtains a new value for the handle argument. This value must be passed to the function on each subsequent call.

POINT

Positions to a specific record.

lognum

ILOG file number, 0 through 99.

subfile

Subfile number within the ILOG file, 0 through 9.

recnum

Record number in the log to locate (POINT). Specify the relative number of the record, or specify FIRST for the first record, or LAST for the last record.

record

Returned record.

direction

Direction in which the subfile is read. Specify FORWARD or BACKWARD.

Default: FORWARD

GET

Retrieves a record. The results are placed in the *ilog_* variables, as shown in the following section.

ELEASE

Releases the storage associated with *handle*. This is automatically performed at IMOD task termination.

Variables Returned by the GET Argument

When an ILOG record is read (GET argument), the retrieved information is placed in a series of predefined REXX variables. These variables and their use are listed next.

ilog_asid

The ASID of the issuer of the WTO.

ilog_date

The date the message was issued, in *yymmdd* format.

ilog_jobid

The job ID of the issuer of the WTO.

ilog_jobnm

The job name of the issuer of the WTO.

ilog_raw

The unformatted ILOG record.

ilog_seg

The number of message lines in the ILOG record.

ilog_seq

The sequence number of a record.

ilog_sysid

The GRS system name of the issuer of the WTO.

ilog_text.n

An individual line of text.

ilog_time

The time the message was issued, in *hh.mm.ss* format.

ilog_type

The record type set by ILOGR(). WTOs are recorded with type = 0.

Usage Notes

The ILOG() and ILOGG() functions handle records from ILOG subfiles in different ways, as explained in the following table.

Function	Treats ILOG subfiles as...	Useful for...
ILOG()	Individual files. You must specify both the ILOG file number and the specific subfile you want to process.	Copying a single subfile for archiving purposes

Function	Treats ILOG subfiles as...	Useful for...
ILOGG()	Parts of a single file in chronological order.	Searching for all occurrences of a particular message

To read an ILOG subfile

1. Establish linkage to an ILOG by obtaining a *handle*.
2. Specify with which record the retrieval is to begin and in what direction you want to read.
3. Read records sequentially or directly.
4. Release the handle.

Return Codes

The ILOG() function produces these return codes:

101 - 106

ARG n MISSING OR INVALID

121

END OF LOG

122

REQUESTED LOG NOT FOUND

123

INVALID FILE HANDLE

124

REQUESTED LOG NOT ACCESSIBLE

125

REQUESTED RECORD NOT IN LOG

Example

```
handle = ilog('obtain')         /* Get a new handle      */
result = ilog('point',handle,0,2,'last','backward')
                                /* Start at end          */
ilog('get',handle) == ''        /* Got the last record   */
ilog_asid          == '1D'      /* For example           */
```

ILOGC()

Use the ILOGC() function to return information from an ILOG subfile.

Syntax

The ILOG function has this syntax:

Form 1:

```
result = ILOGC(OPEN, lognum)
```

Form 2:

```
result = ILOGC(CLOSE, lognum)
```

Form 3:

```
status= ILOGC(STATUS, lognum)
```

Arguments

The ILOGC() function takes these arguments:

OPEN

Opens the ILOG and prepares it for logging.

CLOSE

Closes the ILOG.

STATUS

Request ILOG allocation status

lognum

ILOG file number, 0 through 99.

result

null, if successful. If not successful, an error text will be returned. See Return Codes for text

status

for STATUS request, the possible status that can be returned are:

OPEN

ILOG specified by lognum is open.

CLOSED

ILOG specified by lognum is closed

OPEN NOOPEN

ILOG specified by lognum is open. ILOG was defined in RUNPARMS as NOOPEN.

CLOSED NOOPEN

ILOG specified by lognum is closed. ILOG was defined in RUNPARMS as NOOPEN

NOT-DEFINED

ILOG specified by lognum is not defined to CA GSS.

Return Codes

The ILOGR() function produces these return codes:

101 – 102

ARG n MISSING OR INVALID

122

REQUESTED LOG NOT FOUND

124

REQUESTED LOG NOT ACCESSIBLE

151

REQUESTED LOG ALREADY OPEN/|CLOSED

152

REQUESTED LOG IN USE BY ANOTHER GSS

Usage Notes

By default ILOGs are opened by CA GSS during initialization and remain open until CA GSS is terminated. The initialization parameter NOOPEN in the ILOG statement can be used to delay the opening on the ILOG dataset. The dataset then can be opened by an ILOGC(OPEN,lognum) function call. The ILOGC(CLOSE,lognum) function call can be used on any open ILOG dataset, whether it was opened during initialization or by an ILOGC function call.

Example:

```
/* ILOG 55 was defined to GSS with the NOOPEN option */
result=ILOGC('OPEN'.55)   /* open ilog 55 */
if result\='' then y=wto(result)    /* if result is not null, WTO the error text */
```

ILOGG()

Use the ILOGG() function to return information from an ILOG file.

Syntax

The ILOGG() function has this syntax:

Form 1:

```
handle = ILOGG(OBTAIN)
```

Form 2:

```
ILOGG(POINT,handle,lognum,recnum,[direction])
```

Form 3:

```
record = ILOGG(GET,handle)
```

Form 4:

```
ILOGG(RELEASE,handle)
```

Arguments

The ILOGG() function takes these arguments.

handle

Numeric ID for an internal workspace. A handle must be obtained and then passed to the function with each subsequent call.

OBTAIN

Obtains a new value for the handle argument. This value must be passed to the function on each subsequent call.

POINT

Positions to a specific record.

lognum

ILOG file number, 0 through 99.

recnum

Record number in the log to locate (POINT). Specify the relative number of the record, or specify FIRST for the first record, or LAST for the last record.

record

Returned record.

direction

Direction in which the subfile is read. Specify FORWARD or BACKWARD.

Default: FORWARD

GET

Retrieves a record. The results are placed in the *ilog_* variables as shown in the following section.

RELEASE

Releases the storage associated with *handle*. This is automatically performed at IMOD task termination.

Variables Returned by the GET Argument

When an ILOG record is read (GET argument), the retrieved information is placed in a series of predefined REXX variables. The following table shows these variables and their use:

ilog_asid

The ASID of the issuer of the WTO.

ilog_date

The date the message was issued, in *yymmdd* format.

ilog_jobid

The job ID of the issuer of the WTO.

ilog_jobnm

The job name of the issuer of the WTO.

ilog_raw

The unformatted ILOG record.

ilog_seg

The number of message lines in the ILOG record.

ilog_seq

The sequence number of a record.

ilog_sysid

The GRS system name of the issuer of the WTO.

ilog_text.n

An individual line of text.

ilog_time

The time the message was issued, in *hh.mm.ss* format.

ilog_type

The record type set by ILOGR(). WTOs are recorded with type = 0.

How ILOGG() Differs from ILOG()

ILOGG() differs from the ILOG() function in that all subfiles in the ILOG file are treated as one chronologically sequenced file. See the Administration Guide for an overview of ILOG files.

The ILOG() and ILOGG() functions handle records from ILOG subfiles in different ways, as explained in the following table:

Function	Treats ILOG subfiles as...	Useful for...
ILOG()	Individual files. You must therefore specify both the ILOG file number and the specific subfile you want to process.	Copying a single subfile for archiving purposes
ILOGG()	Parts of a single file in chronological order.	Searching for all occurrences of a particular message

To read an ILOG subfile

1. Establish linkage to an ILOG by obtaining a *handle*.
2. Specify with which record the retrieval is to begin and in what direction you want to read.
3. Read records sequentially or directly.
4. Release the handle.

Return Codes

The ILOGG() function produces these return codes.

101 - 106

ARG *n* MISSING OR INVALID

121

END OF LOG

122

REQUESTED LOG NOT FOUND

123

INVALID FILE HANDLE

124

REQUESTED LOG NOT ACCESSIBLE

125

REQUESTED RECORD NOT IN LOG

Example

```
handle = ilogg('obtain')      /* Get a new handle      */
result = ilogg('point',handle,0,,'last','backward')
                              /* Start at end          */
ilogg('get',handle) == ''     /* Get the last record   */
ilog_asid == '1D'             /* For Example           */
ilog_jobnm == 'CUSTAPE'
```

ILOGR()

Use the ILOGR() function to record data in an ILOG file.

Syntax

The ILOGR() function has this syntax:

```
ILOGR(lognum,rectype,{VAR|STEM|STACK},data)
```

Arguments

The ILOGR() function takes these arguments:

lognum

ILOG number where the record is to be written.

rectype

Record-type number for the record. Record types range from 100 to 32,767. Values less than 100 are reserved for use by CA.

VAR

Indicates that a single line of text is supplied in a variable or as a literal.

STEM

Indicates that multiple lines of text are supplied in a stem variable.

STACK

Indicates that multiple lines of text are supplied from the stack.

data

Text to be recorded, as follows:

(*if VAR was specified*): Text string or variable.

(*if STEM was specified*): Name of a stem variable, including the period.

(*if STACK was specified*): Number of the stack containing the data lines.

Each data line can be no longer than 255 characters. The maximum number of data lines is 32,767, and the maximum number of characters in an entire record is 32,715 minus the number of lines returned.

Text records taken from stem variables are processed in the same order as the individual members were created.

Return Codes

The ILOGR() function produces these return codes:

101 - 104

ARG *n* MISSING OR INVALID

122

REQUESTED LOG NOT FOUND

123

ILOG IS FULL

124

REQUESTED LOG NOT ACCESSIBLE

125

STEM ITEM IS NUMERIC, NOT TEXT

126

STEM ITEM IS TOO LONG

Example

Consider this example:

```
ilogr(0,1200,'VAR','Process 7 successful')
                                /* Write text on ILOG file 0 */
ilogr(1,2321,'STEM','rec.')     /* Write multi-line record from
                                   rec. variables  */
ilogr(0,9999,'STACK',5)         /* Write multi-line record
                                   using contents of stack 5 */
```

IROUTE()

Use the IROUTE() function to control ILOG routing of the WTO text that triggered the currently running IMOD. The currently assigned routing may be inspected and a new routing assigned.

Syntax

The IROUTE() function has this syntax.

Form 1:

```
lognum = IROUTE(FETCH)
```

Form 2:

```
IROUTE(SET,newlog)
```

Arguments

The IROUTE() function takes these arguments:

lognum

ILOG file where the WTO text is to be recorded, following completion of the IMOD. A null value indicates the text will not be recorded on any file.

FETCH

Returns the number (0 to 99) of the currently assigned ILOG file.

SET

Sets a new ILOG destination.

newlog

Number (0 to 99) identifying the ILOG file where the WTO text is to be recorded. A null value prevents recording on any ILOG.

Any ILOG number that is syntactically valid is accepted by the function. No attempt is made to ensure that the specified ILOG exists and is available.

Return Codes

The IROUTE() function produces these return codes:

101 - 102

ARG *n* MISSING OR INVALID

121

ILOG FILE NOT DEFINED

Example

```
dest = iroute('FETCH')     /* Obtain the ILOG file number where
                              the message is to be recorded    */
iroute('SET',1) == ''      /* Change the destination to ILOG 1 */
```

KEYVAL()

Use the KEYVAL() function to scan an input string for a key.

Syntax

The KEYVAL() function has this syntax:

```
result = KEYVAL(string,key,[default])
```

Arguments

The KEYVAL() function takes these arguments:

result

Contains one of the following:

If an error occurs, error text is returned.

If key is found, the next blank-delimited word is returned.

Otherwise, the value for default is returned.

string

Text string to be scanned.

key

Keyword value to scan for in the input string.

If matched, the value returned is the first non-blank character after key, and all subsequent characters until the next blank.

default

Value to be returned if key is not found. The default is null string.

Return Codes

The KEYVAL() function produces these return codes:

101-103

ARG n MISSING OR INVALID

Example

```
keyval('a hush was heard','hu','NOT FOUND')   == 'was'
                                                 /* Next word */
keyval('a hush was heard',' hu ','NOT FOUND') == 'NOT FOUND'
```

LOADIMOD()

In batch mode (SRVBATCH), use the LOADIMOD() function to enable an IMOD to load other IMODs and make them available for execution as external subroutines.

Syntax

The LOADIMOD() function has this syntax:

failed_list = LOADIMOD(*iset*,*namelist*)

Arguments

The LOADIMOD() function takes these arguments:

failed_list

Returns a blank-delimited list of IMODs that could not be loaded (or other error messages).

iset

ISET name of the IMOD file to be accessed.

namelist

List of IMODs to be loaded into storage, separated by blanks. The maximum list length is 256 characters.

Usage Notes

- LOADIMOD() is valid only when the IMOD is executing in batch mode, under SRVBATCH.
- Because each use of LOADIMOD() opens and closes an IMOD file, load as many IMODs as possible each time you call the function.

Return Codes

The LOADIMOD() function produces these return codes:

101 - 102

ARG *n* MISSING OR INVALID

121

A list of the IMODs not loaded, separated by blanks.

123

OPEN FAILED

122

FUNCTION ONLY VALID IN BATCH MODE

Example

```
loadimod('GSSTECH',asid_00 not_here hello) == 'NOT_HERE'
                              /* rc==121                        */
                              /* This runs in batch only        */
                              /* IMOD 'not_here' does not exist */

                     /* Following messages appear in ISRVLOG:  */
/* 11:24:38 SRV282 Loading IMOD's from GSSTECH File Version:
                                                      02.03.02  */
/* 11:24:38 SRV168 ASID_00   loaded. 05664D38 EWF.IMOD.GSSTECH */
/* 11:24:38 SRV168 HELLO     loaded. 056649F8 EWF.IMOD.GSSTECH */
```

LOCATE()

Use the LOCATE() function to return the memory address of selected control blocks and structures.

Syntax

The LOCATE() function has this format:

```
address = LOCATE(item,[opt])
```

Arguments

The LOCATE() function takes these arguments:

address

Address of the item specified by one of the keywords. The returned value is in decimal form.

item

Control block or structure. Specify one of the following:

ASCB Returns the ASCB of the ISERVE address space.

ASID Returns the ASID of the ISERVE address space.

CSCB Returns the CSCB of the ISERVE address space. If *opt* is also specified, the appropriate CSCB address is returned.

CVT Returns the CVT.

RCT Returns the RCT.

RMCT Returns the RMCT.

TIOT Returns the TIOT of the ISERVE address space.

opt

ASID identifying the address space whose CSCB is to be returned.

Return Codes

The LOCATE() function produces these return codes:

101

ARG 1 MISSING OR INVALID

Example

```
locate('cvt')      == '16610568'/* Current CVT address in decimal     */
d2x(locate('cvt')) == 'FD7508'  /* Current CVT address in hexadecimal */
```

LOCIMOD()

Use the LOCIMOD() function to determine the availability of an IMOD subroutine before you try calling it.

Syntax

The LOCIMOD() function has this format:

```
address {GOOD|BAD} {CALL|NCALL} {SOURCE|NSOURCE} =  LOCIMOD(imodname)
```

Arguments

The LOCIMOD() function takes these arguments:

address

IMOD's memory address (in REXX decimal format).

GOOD

The IMOD was successfully loaded and is executable.

BAD

An error occurred during IMOD loading and the requested IMOD is not executable.

CALL

The IMOD is available to be called as a subroutine from another IMOD.

NCALL

The IMOD cannot be called as a subroutine from another IMOD.

SOURCE

The IMOD's source code is in memory and is available to the SOURCELINE() function and TRACE facility.

NSOURCE

The IMOD's source code was not loaded into memory and is not available to the SOURCELINE() function or TRACE facility.

imodname

Name of the IMOD to be tested. This argument is not case-sensitive.

Usage Notes

An IMOD that has not been loaded sets RC to 121. An IMOD that has been loaded but is not currently available sets RC to 0 and returns BAD as the status.

Return Codes

The LOCIMOD() function produces these return codes:

101

ARG 1 MISSING OR INVALID

121

NOT FOUND

Example

Consider this example:

```
/*  Conditionally send a message to a logging subroutine.  If the
    subroutine is not available, issue the message as a WTO */
    parse arg msg
    status = locimod('log_routine')
    if RC = 0 then do
       parse var status w1 w2 w3 .
       if w1 = 'BAD' then RC = 121
       if w2 = 'NCALL' then RC = 121
       end
    if RC = 0 then do
       call log_routine msg
       return
       end
    x = wto(msg)
    return
```

LOCSTEM()

Use the LOCSTEM() function to determine the values used for a stem variable's indexes.

Syntax

The LOCSTEM() function has this syntax:

```
index = LOCSTEM(stem.,[num])
```

Arguments

The LOCSTEM() function takes these arguments:

stem.

Stem variable's name (including the period).

num

Number of a stem item.

index

Returns one of the following:

(if *num* is specified): Value of the index for the specified stem item.

(if *num* is omitted or zero): Count of stem elements.

(if an error occurs): Error text.

Indexes are returned in the same order as they were created.

Return Codes

The LOCSTEM() function produces these return codes:

101 - 102

ARG *n* MISSING OR INVALID

121

STEM VARIABLE NOT FOUND

122

STEM HAS NO MEMBERS

Example

Consider this example:

```
limit = locstem('ABC.')             /* Obtain count of members */
do i = 1 to limit
   index = locstem('ABC.',i)        /* Obtain value for index  */
   say 'ABC.'||index 'Has value: ABC.index
   end i
```

LU2()

Use the LU2() function to enable the CA-GSS/ISERVE Logon Facility to communicate with devices defined as logical unit 2.

Syntax

The LU2() function has this syntax:

Form 1:

xx yy = LU2(BRACKET,*rpl*,[*xx*],[*yy*])

Form 2:

result = LU2(SEND,*rpl*,*outrec*)

Form 3:

input = LU2(RECEIVE,*rpl*)

Form 4:

input = LU2(SENDREC,*rpl*,*outrec*,[*option*])

Form 5:

result = LU2(CLSDST,*rpl*)

Arguments

The LU2() function takes these arguments:

xx

BB (begin bracket) or NBB (no begin bracket).

yy

EB (end bracket) or NEB (no end bracket).

BRACKET

Modifies the SNA brackets to be used with the next SEND or RECEIVE operation. If new values are omitted, the current state is returned.

rpl

RPL address, as passed to the IMOD by the Logon Facility.

result

Error text (if the SEND or CLSDST operation fails).

SEND

Transmits a record.

outrec

Complete 3270 data stream.

input

Results of a 3270 READ MODIFIED operation, a complete 3270 data stream.

RECEIVE

Issues a VTAM RECEIVE against the specific terminal session. The IMOD task is suspended until the terminal operator presses ENTER and a READ MODIFIED is completed.

SENDREC

Transmits a 3270 data stream to the terminal, waits for operator action, and receives the results.

option

One of these options:

Bracket Determines and sets the appropriate bracket state for the next transmission

NOBracket Indicates that the bracket indicators in the RPL are not changed.

Default: Bracket.

CLSDST

Issues a VTAM CLOSEDEST against the RPL. This terminates the session and returns control of the terminal to VTAM. If omitted, a CLSDST is automatically generated at the end of the IMOD task.

Usage Notes

- Many of this function's arguments are not documented since they are used only by the internal portion of the Logon Facility.
- Before attempting to use LU2(), see the information on the Logon Facility in this guide.
- 3270 data streams may be parsed and constructed through the $3270() and DS3270() functions.

Return Codes

The LU2() function produces these return codes:

101 - 104

ARG *n* MISSING OR INVALID

126

NOT RPL

127

NO RPL

128

NOT ACB

129

NO ACB

130

ACB IS OPEN

131

ACB NOT OPEN

135

PHASE: *type rpl13 rpl14 rpl15*

136

PHASE: *type* R15: *xx* REASON: *xx*

Example

```
parse arg rpl .                         /* Obtain RPL from caller   */
queue 'at 1 1 attr /prot/ data /Enter transaction =>/'
queue 'attr /input high / cursor at 1 79 attr /prot/'
screen = $3270(,,'erase','reset')     /* Construct screen         */
if rc ^= 0 then signal error_$3270
input = lu2('sendrec',rpl,screen)     /* Send screen, read screen */
input = ds3270('from','field',,,1,22) /* Extract input field      */
if rc ^= 0 then signal err_ds3270
...
 result = lu2('clsdst',rpl)            /* Terminate session        */
return
```

MEMORY()

Use the MEMORY() function to fetch or store memory contents.

Note: If you are not using the appropriate address space ID and storage protection key, the MEMORY() function fails.

Syntax

The MEMORY() function has this syntax:

Form 1:

```
result = MEMORY({Fetch|FetchN|FetchNX|FetchX},loc,bytes)
```

Form 2:

```
MEMORY(STORE,loc,bytes,value,[verify])
```

Arguments

The MEMORY() function takes these arguments:

result

Returned data (on a successful fetch operation), null string (on a successful store operation), or error text.

FETCH

Retrieves a character string from memory. No data conversion is performed. May be abbreviated to F.

FETCHX

Retrieves a character string from memory with the high-order bit always turned off. Use FETCHX to access a hexadecimal address. May be abbreviated to FX.

FETCHN

Retrieves a value from memory and converts it to a decimal value. This value will be positive or negative, depending upon the high-order bit in the source string. May be abbreviated to FN.

FETCHNX

Retrieves a value from memory, discards the high-order bit, and returns the remaining positive value as a decimal number. May be abbreviated to FNX.

STORE

Stores the specified string in memory. The stored string is not converted but is stored as is.

loc

Memory address of the storage area to be accessed.

bytes

Number of bytes to retrieve or store. The maximum number is 256 for a binary string (no conversion). FETCHX, FETCHN, and FETCHNX have a limit of 8 bytes.

value

Value to replace the current storage contents. The length must be the same as specified by the bytes argument. No type conversion is performed; value is assumed to specify data in the same format that it is to be stored.

verify

Memory location's current binary value. If the value you specify does not match the actual value, the store operation is not carried out.

Return Codes

The MEMORY() function produces these return codes:

101 - 109

ARG *n* MISSING OR INVALID

124

VERIFY FAILED

125

FETCHN INVALID FOR SPECIFIED LENGTH

126

EXCESS ARGUMENTS

Example

```
memory('f',16,4)  == '80FD7508'x    /* A data string, no
                                       conversion */
memory('fx',16,4) == '00FD7508'x    /* A data string, high bit
                                       off */
memory('fn',16,4) == '-2164094216'  /* A signed decimal
                                       conversion */
memory('fnx',16,4)== '16610568'     /* Decimal conversion, high

                                        bit off*/
```

MLWTO()

Use the MLWTO() function to issue the specified text as a multi-line WTO.

Syntax

The MLWTO() function has this syntax:

```
msgid = MLWTO([stem.],[rout],[desc],[cons],[area])
```

Arguments

The MLWTO() function takes these arguments:

msgid

System-assigned ID. Value can be passed to the DOM() function to delete message from the console. If function fails, error text is returned.

stem

Root name of a stem variable containing the text to be issued. Include the period. CA recommends that the name be enclosed within quotation marks.

The stem items are used as text lines, in the order created (not by index value). In future releases of CA-GSS, the variables will be used in collating sequence. If you create them in ascending collating sequence, you will not need to rename them later.

The variable created first is used for the control line. Up to 25 additional lines are used as data lines. The following restrictions apply:

The control line cannot exceed 34 characters.

Data lines cannot exceed 70 characters each.

You cannot use the *stem*.0 variable. It is currently reserved.

The stem variable cannot be global; that is, it cannot begin with an ampersand (&).

rout

To reuse the variables with fewer lines, use the REXX DROP command for the stem. This deletes the entire stem group and allows you to create a set with fewer variables.

If you omit the stem variable, the contents of the WTO will be the entire current contents of the currently accessed stack.

desc

One or more routing codes (like'1' or '1,2,3' or '1-5,8,10,15-128').

One or more descriptor codes (like '1' or '1,2,3' or '1-5,8,10,12-16').

For information on routing and descriptor codes, see the IBM manual *Supervisor Services and Macro Instructions* (GC28-1154-4).

cons

Console ID to receive the message.

If the IMOD is being executed as the result of an operator command, the predefined variable *imod_console* will contain the console ID of the console issuing the command.

area

Console area where the message is to be displayed. This argument has an effect only if the destination console has the requested area defined. If *area* is omitted or not defined, the message appears in the standard display area of the console.

Return Codes

The MLWTO() function produces these return codes:

101

One of the following for ARG 1:

ARG 1 MISSING OR INVALID

STEM VARIABLE NOT FOUND

STEM HAS NO MEMBERS

102

One of the following for ARG 2:

NON-NUMERIC CHARACTER

CONTIGUOUS RANGES

NUMBER TOO BIG

LIST IS INCOMPLETE

RANGE IS INVALID

ZERO VALUE

103

One of the following for ARG 3.

NON-NUMERIC CHARACTER

CONTIGUOUS RANGES

NUMBER TOO BIG

LIST IS INCOMPLETE

RANGE IS INVALID

ZERO VALUE

104

CONSOLE ID INVALID

105

AREA INVALID

Example

```
queue 'label line'              /* Get messages ready */
queue 'message line 1'
queue 'message line 2'
msgid = mlwto()                 /* Issue queued messages using
                                   system defaults*/
```

OPSVALUE()

Use the OPSVALUE() function to interrogate and set the values of CA OPS/MVS Event Management and Automation global variables. These values are maintained in the CA OPS/MVS Event Management and Automation address space.

Syntax

The OPSVALUE() function has this syntax:

```
OPSVALUE(name,[actioncode],[newvalue],[oldvalue])
```

Arguments

The OPSVALUE() function takes these arguments:

name

Name of the CA OPS/MVS Event Management and Automation global variable to be referenced. This value can be a literal string or any expression that will evaluate to the name of a global variable.

actioncode

Action code defining the operation to be performed. For a list of valid action codes and descriptions of each, see your CA OPS/MVS Event Management and Automation manual set.

newvalue

New value for the CA OPS/MVS Event Management and Automation variable.

Oldvalue

Previous value of the CA OPS/MVS Event Management and Automation variable. This value is compared with the current contents of the variable prior to assigning the value specified by newvalue. If the content of the current variable does not match that specified by *oldvalue*, the operation is suppressed and an error condition returned.

Usage Notes

- OPSVALUE() is valid only if CA OPS/MVS Event Management and Automation is installed and operational on your system.
- If you reference a CA OPS/MVS Event Management and Automation variable several times (perhaps in a loop), consider using the OPSVALUE() function to assign the value to a local variable.
- Several copies of CA OPS/MVS Event Management and Automation can execute simultaneously on your system. Each is identified by the subsystem ID that is assigned to it. By default, OPSVALUE() communicates with the CA OPS/MVS Event Management and Automation specified in the ISERVE initialization parameters. You can override this value by using the SETADDR() function.
- OPSVALUE() has the same syntax and follows the same conventions as the OPSVALUE() available to CA OPS/MVS Event Management and Automation users. For complete information on using OPSVALUE(), see the CA OPS/MVS Event Management and Automation manual set.

Return Codes

The OPSVALUE() function produces these return codes:

101 - 104

ARG *n* MISSING OR INVALID

Example

```
x = setaddr('ops','OPS5')   /* Set target CA-OPS/MVS */
x = opsvalue(...            /* Fetch or store value */
```

OSCMD()

Use the OSCMD() function to issue the specified text as a console operator command.

Syntax

The OSCMD() function has this syntax:

```
result = OSCMD(text,[console])
```

Arguments

The OSCMD() function takes these arguments:

result

Null string (or error text).

text

Text to be issued as a console operator command.

console

Console where the command is to be executed.

Default: The command is routed through ISERVE's subsystem console (if available). Otherwise, the command is issued through the master console.

Usage Note

Returned information only indicates that the command has been accepted by the system for processing (processing may consist of determining that the command is invalid). No information is returned indicating the success or failure of the passed command.

Return Codes

The OSCMD() function produces these return codes:

101 - 102

ARG *n* MISSING OR INVALID

Example

```
x = oscmd('d a,l',0)   /* The command is issued using console 0.*/
```

P2D()

Use the P2D() function to convert an IBM packed-decimal number into a REXX-format whole number.

Syntax

The P2D() function has this syntax:

number = P2D(*pdec*)

Arguments

The P2D() function takes these arguments:

number

Whole number.

pdec

Packed-decimal number. Each byte contains 2 decimal digits, except the right-most byte, which may contain a sign digit (hexadecimal F for positive, B for negative) in the right half of the byte.

Return Codes

The P2D() function produces these return codes:

101

ARG 1 MISSING OR INVALID

Example

```
/*  Unpack a julian date from ISPF statistics format */

parse arg value                          /* date: 00 yy dd dF */
julian = p2d(value)                      /* REXX number       */
final = right(insert('.',julian,length(julian)-3),6,'0')
                                         /* yy.ddd            */
```

PAUSE()

Use the PAUSE() function to stop the execution of an IMOD for the specified number of seconds.

Syntax

The PAUSE() function has this syntax:

PAUSE(*seconds*)

Arguments

The PAUSE() function takes this argument:

seconds

Number of seconds to pause, from 1 to 3600 (one hour).

Return Codes

The PAUSE() function produces these return codes:

101

ARG 1 MISSING OR INVALID

120

STIMERM FAILED

Example

```
pause(3600) == ''           /* A one-hour pause   */
```

PUBSTACK()

Use the PUBSTACK() function to declare one or more stacks public or to establish the default stack for the current IMOD or for other requesting IMODs.

Syntax

The PUBSTACK() function has this syntax:

```
result = PUBSTACK({DEFAULT},[n,[n,..[n]])
                  {PRIV}
                  {READ}
                  {OREAD}
                  {RESET}
                  {PUB}
                  {WRITE}
                  {OWRITE}
                  {QUEUE}
                  {OQUEUE}
```

Arguments

The PUBSTACK() function takes these arguments:

result

Null string, if the function is successful; otherwise, error text is returned.

DEFAULT

Designates the specified stack (argument 2) as the default stack. Specify only one value for *n*.

RESET

Resets all stacks to private status. No stack numbers are permitted.

PRIV

Sets all specified stacks as private. You can specify up to 19 stacks per function call.

PUB

Sets all specified stacks to public status with READ, WRITE, and QUEUE status. You can specify up to 19 stacks per one function call.

READ

Adds READ status to the specified stacks. External IMODs can read such stacks non-destructively using TUG().

WRITE

Adds WRITE status to the specified stacks. External IMODs can write to the stacks. If the stack also has READ status, a record can be read destructively using PULL().

QUEUE

Adds QUEUE status to the specified stacks. QUEUE is a subset of WRITE status. A stack with QUEUE status, but not WRITE status, can have records queued to it by external IMODs using QUEUE().

OREAD

Cancels the previous status of the specified stacks and sets the status to READ only.

OWRITE

Cancels the previous status of the specified stacks and sets the status to WRITE only.

OQUEUE

Cancels the previous status of the specified stacks and sets the status to QUEUE only.

n

Stack number. This argument can be repeated for a total of 19 specifications per function call, except for the RESET operation, which permits no other arguments, and the DEFAULT operation, which requires only one specification of *n*.

Return Codes

The PUBSTACK() function produces these return codes.

101 - 120

ARG *n* MISSING OR INVALID

121

NO STACKS SPECIFIED

122

EXCESS ARGUMENTS

Example

Consider this example:

```
x = pubstack('RESET')         /* Reset all stacks to original conditions */
x = pubstack('DEFAULT',2)     /* Set stack 2 to default status */
x = pubstack('QUEUE',1,2,3)   /* Allow queuing to stacks 1, 2 & 3 */
x = pubstack('OREAD',3)       /* Reset stack 3 to READ ONLY status */
```

Usage Note

By default, all stacks are private.

PULL()

Use the PULL() function to remove records from any stack belonging to any IMOD. PULL() is a destructive operation.

Syntax

The PULL() function has this syntax:

```
record = PULL([stack],[owner],[cntl])
```

Arguments

The PULL() function takes these arguments:

record

Returned data or error text.

stack

Number of the stack to be referenced.

Default: If a local stack is referenced, the current stack (as set by the SWAPSTAK() function); otherwise, the value is set by the owning IMOD task using the PUBSTACK(DEFAULT) function.

owner

IMOD ID of the stack's owner (for external reference). This value can be obtained from the variable *imod_id* in the owning IMOD task. It is also the value returned by the SPAWN() function.

Default: Current IMOD task.

cntl

Name of a variable to receive the control information stored with the record.

Default: Control information is not retrieved.

Usage Notes

- Whereas PULL translates all records to uppercase, PULL() does not. The function of PULL() is equivalent to PARSE PULL.
- The results of a PULL() instruction may not be directly executed as a command. The result must first be assigned to a variable and then issued as a command. This is due to the REXX restriction that prohibits a command string (not in quotes) from beginning with a reserved keyword, in this case, PULL.
- PULL() can specify a stack external to the current IMOD only when the target stack has been declared public for both reading and writing by the owning IMOD task using the PUBSTACK() function.

Return Codes

The PULL() function produces these return codes:

101 - 103

ARG *n* INVALID

122

RECORD NOT FOUND

125

SPECIFIED IMOD NOT FOUND

126

STACK NOT SPECIFIED AND NO DEFAULT

127

REQUESTED STACK DOES NOT EXIST

128

SPECIFIED IMOD/STACK NOT AUTHORIZED

Example

```
rec = pull()                /* A record is retrieved from the
                               current stack */
rec = pull(2)               /* A record is retrieved from local
                               stack 2 */
rec = pull(,453,'owner')    /* A record is retrieved from IMOD
                               453's default stack. The control
                               information will be placed in the
                               variable named "owner" */
```

PUSH()

Use PUSH() to add records to the top of any stack belonging to any IMOD.

Restriction

PUSH() can specify a stack external to an IMOD only when the target stack has been declared public for writing by the owning task.

Syntax

The PUSH() function has this syntax:

```
PUSH([record],[stack],[owner],,[postoption])
```

Arguments

The PUSH() function takes these arguments:

record

Record to be added to the stack. A zero-length record is added by default.

stack

Number of the stack to be referenced.

Default: If a local stack is referenced, the current stack (as set by the SWAPSTAK() function); otherwise, the value is set by the owning IMOD task using the PUBSTACK(DEFAULT) function.

owner

IMOD ID of the stack's owner (for external reference). This value can be obtained from the variable *imod_id* in the owning IMOD task. It is also the value returned by the SPAWN() function. The default is the current IMOD task.

postoption

Wait option that determines whether the owning IMOD waits for a record to appear on an external stack. Specify one of these values.

POST

Any outstanding wait is satisfied; default.

NOPOST

An outstanding wait is not satisfied.

ONLYPOST

Any outstanding wait is satisfied, but a record is not placed on the stack.

Note: If you are placing several records onto an external stack, CA recommends that you specify NOPOST for all but the final record. IMODs are time-sliced, and if the POST operation is performed before the final record is pushed on the stack, the IMOD task owning the stack could be dispatched to process before the final PUSH is completed.

Return Codes

The PUSH() function produces these return codes:

101 - 105

ARG *n* INVALID

122

STACK NOT SPECIFIED AND NO DEFAULT

123

SPECIFIED STACK DOES NOT EXIST

125

SPECIFIED IMOD NOT FOUND

126

STACK WRITE FAILED

127

REQUESTED STACK DOES NOT EXIST

128

SPECIFIED IMOD/STACK NOT AUTHORIZED

Example

```
x = push(rec)              /* A record is written to the
                              current stack */
x = push(rec,2)            /* A record is written to local
                              stack 2 */
x = push(rec,,453,,'post') /* A record is written to IMOD 453's
                              default stack. Any outstanding
                              wait will be marked complete. */
```

QUEUE()

Use QUEUE() to add records to the bottom of any stack belonging to any IMOD.

Restriction

QUEUE() can specify a stack external to an IMOD only when the target stack has been declared public for writing or for queuing.

Syntax

The QUEUE() function has this syntax:

```
QUEUE([record],[stack],[owner],,[postoption])
```

Arguments

The QUEUE() function takes these arguments:

record

Record to be added to the stack. A zero-length record is added by default.

stack

Number of the stack to be referenced.

Default: If a local stack is referenced, the current stack (as set by the SWAPSTAK() function); otherwise, the value is set by the owning IMOD task using the PUBSTACK(DEFAULT) function.

owner

IMOD ID of the stack's owner (for external reference). This value can be obtained from the variable *imod_id* in the owning IMOD task. It is also the value returned by the SPAWN() function.

Default: Current IMOD task.

postoption

Wait option, which determines whether the owning IMOD waits for a record to appear on an external stack. Specify one of these values:

POST

Any outstanding wait is satisfied; the default.

NOPOST

An outstanding wait is not satisfied.

ONLYPOST

Any outstanding wait is satisfied, but a record is not placed on the stack.

Note: If you are placing several records onto an external stack, CA recommends that you specify NOPOST for all but the final record. IMODs are time-sliced, and if the POST operation is performed before the final record is queued to the stack, the IMOD task owning the stack could be dispatched to process before the last QUEUE is completed.

Return Codes

The QUEUE() function produces these return codes:

101 - 105

ARG *n* MISSING OR INVALID

122

STACK NOT SPECIFIED AND NO DEFAULT

123

SPECIFIED STACK DOES NOT EXIST

125

SPECIFIED IMOD NOT FOUND

126

STACK WRITE FAILED

127

REQUESTED STACK DOES NOT EXIST

128

SPECIFIED IMOD/STACK NOT AUTHORIZED

Example

```
x = queue(rec)                 /* A record is queued to the
                                  current stack */
x = queue(rec,2)               /* A record is queued to local
                                  stack 2 */
x = queue(rec,,453,,'post')    /* A record is queued to IMOD 453's
                                  default stack. Any outstanding
                                  wait will be marked complete. */
```

QUEUES()

Use QUEUES() to add an entire stack to the bottom of any stack belonging to any IMOD.

Restriction

QUEUES() can specify a target stack external to an IMOD only when the target stack has been declared public for writing or for queuing.

Syntax

The QUEUES() function has this syntax:

```
QUEUES(source,[stack],[owner],,[postoption])
```

Arguments

The QUEUES() function takes these arguments:

source

Stack whose content is to be queued to the target stack.

Default: Current stack.

stack

Number of the stack to be referenced.

Default: If a local stack is referenced, the current stack (as set by the SWAPSTAK() function); otherwise, the value is set by the owning IMOD task using the PUBSTACK(DEFAULT) function.

owner

IMOD ID of the stack's owner (for external reference). This value can be obtained from the variable *imod_id* in the owning IMOD task. It is also the value returned by the SPAWN() function.

Default: Current IMOD task.

postoption

Wait option, which determines whether the owning IMOD waits for a record to appear on an external stack. Specify one of these values:

POST

Any outstanding wait is satisfied; the default.

NOPOST

An outstanding wait is not satisfied.

ONLYPOST

Any outstanding wait is satisfied, but a record is not placed on the stack.

Usage Notes

- The contents of the stack are transferred rather than copied.
- QUEUES() is preferred over QUEUE() when transferring more than one related record to an external stack. This prevents the possibility of the target IMOD processing a partial request.

Return Codes

The QUEUES() function produces these return codes:

101 - 105

ARG *n* MISSING OR INVALID

122

STACK NOT SPECIFIED AND NO DEFAULT

123

SPECIFIED STACK DOES NOT EXIST

125

SPECIFIED IMOD NOT FOUND

126

STACK WRITE FAILED

127

REQUESTED STACK DOES NOT EXIST

128

SPECIFIED IMOD/STACK NOT AUTHORIZED

129

SOURCE AND TARGET IDENTICAL

Example

```
x = queues(5)        /* Stack 5 is queued to the current stack */
x = queues(5,2)      /* Stack 5 is queued to local stack 2     */
x = queues(5,,453,,'post') /* Stack 5 is queued to IMOD 453's
                              default stack. Any outstanding
                              wait will be marked complete.    */
```

RDJFCB()

Use RDJFCB() to request the operating system's RDJFCB service and obtain a copy of the job file control block (JFCB).

Syntax

The RDJFCB() function has this syntax:

jfcb = RDJFCB(*ddname*)

Arguments

The RDJFCB() function takes these arguments:

jfcb

176-byte job file control block, if the function is successful. Otherwise, error text is returned.

ddname

DDname of the file whose JFCB you wish to read.

Return Codes

The RDJFCB() function produces these return codes:

101

ARG 1 MISSING OR INVALID

120

DDNAME not found

Example

```
jfcb = rdjfcb('SYSUT1')       /* Obtain copy of the Job File
                                      Control Block */
dsname = substr(jfcb,1,44)    /* Extract the file's dsname */
dsname = strip(dsname)        /* Remove trailing blanks */
```

REDIRECT()

Use REDIRECT() to send TRACE or SAY output to a stack instead of to the ISERVE log. The recorded information may be retrieved later using standard stack-manipulating functions and commands.

Syntax

The REDIRECT() function has this syntax:

stack = REDIRECT({TRACE|SAY},[*stack*])

Arguments

The REDIRECT() function takes these arguments:

TRACE

Redirect TRACE output.

SAY

Redirect SAY output.

stack

Stack number to receive the redirected output. To terminate redirection of the TRACE or SAY output, code this argument as OFF.

Default: The current value is returned but not changed.

Return Codes

The REDIRECT() function produces these return codes:

101 - 102

ARG *n* MISSING OR INVALID

Example

```
x = redirect('trace',5)      /* Trace output is written on
                                stack 5 */
 x = redirect('say','off')    /* Redirection of SAY output is
                                terminated.*/
```

SAM()

Use SAM() to perform I/O operations on sequential and partitioned data sets.

Syntax

The SAM() function has this syntax:

Form 1:

dcb = SAM(OBTAIN,[*scope*],*ddname*,[*use*])

Form 2:

SAM(*action1*,*dcb*)

Form 3:

SAM(RELEASE,*dcb*)

Form 4:

record = SAM(GET,*dcb*)

Form 5:

SAM(PUT,*dcb*,*record*,[*pad*])

Form 6:

SAM(OPTION,*dcb*,[*recfm1*,][*lrecl*],[*blksize*])

Form 7:

SAM(FIND,*dcb*,*member*)

Form 8:

ttr = SAM(NOTE,*dcb*)

Form 9:

entry = SAM(BLDL,*dcb*,*member*)

Form 10:

entry = SAM(STOW,*dcb*,*member*,[*ttr*],[*action2*],[ALIAS],[*newname*],[*ttrn*],[*data*])

Form 11:

state = SAM(STATUS,*dcb*)

Form 12:

```
dsorg recfm2 lrecl blksize = SAM(INFO,dcb)
```

Arguments

The SAM() function takes these arguments:

dcb

Value that identifies a file and an associated internal workspace.

OBTAIN

Obtains a new DCB.

scope

Scope of the data set. Specify one of the following:

LOCAL

Indicates that the file DCB is *local* to the IMOD; the DCB is automatically closed, and the DCB is released at the conclusion of the IMOD. LOCAL is the default.

GLOBAL

Indicates that the file is *not* automatically closed and the DCB is *not* automatically freed at IMOD termination. Instead, other IMODs can use the DCB to continue processing the same data set. Only one IMOD task can use the GLOBAL DCB at a time. GLOBAL files must be explicitly closed, and the DCBs must be explicitly freed.

Usage Notes

- Using the SAM() function requires the following steps:
  1. Obtain and initialize a DCB (data control block), which is a value that identifies the file and permits ISERVE to remember certain aspects of how you are using it.
  2. Open the file to establish a connection between the DCB and the file.
  3. Perform I/O operations on the file. These must be consistent with the type of data set you are accessing and the manner in which you initialized the DCB.
  4. Close the file. This is especially important for files used for output because physical data updates are not necessarily completed until a CLOSE is performed.
  5. Release the DCB when all processing is concluded. This action frees the storage occupied by the DCB and associated data buffers.
- Close and release operations are automatically performed at the end of the IMOD task, *except* when the OBTAIN specified *perm*.

- A particular DCB may not be used by multiple IMOD tasks simultaneously, even if GLOBAL was specified. Once any I/O operation is performed on a GLOBAL DCB, that DCB is assigned to the IMOD task for its duration.
- If multiple open DCBs specify the same ddname, the results are unpredictable.
- If the same data set is being updated from multiple IMOD tasks, the results are unpredictable.
- To replace the contents of a PDS member, perform all of the steps to create a new member. When issuing the STOW, specify REPLACE.
- Deleting or replacing a member in a PDS does *not* release the space previously occupied by the data. To recover this space, you must perform a *compress* operation, using the IEBCOPY utility program. This restriction does not apply if your installation licenses CA PDSMAN PDS Library Management space reuse feature.
- The directory of a PDS occupies the first tracks and records in the data set. If you open a PDS as a sequential data set with a DISP of OLD or SHR, in output mode, you will destroy that PDS.

Return Codes

The SAM() function produces these return codes:

101 - 109

ARG *n* MISSING OR INVALID

121

ALREADY OPEN

122

OPEN FAILED

123

FILE NOT OPEN

124

FILE OPEN FOR OUTPUT

125

ATTEMPTING TO READ PAST EOF

126

EOF

127

FILE OPEN FOR INPUT

128

INVALID FILE HANDLE

129

FILE NOT CLOSED

130

MEMBER NOT FOUND

131

DATASET NOT PARTITIONED

132

TRUNCATED

133

INVALID FILE HANDLE

134

READ FAILURE

135

DUPLICATE NAME

136

MEMBER NOT FOUND

137

NO DIRECTORY SPACE

138

PERMANENT I/O ERROR

139

DCB CLOSED OR OPEN FOR INPUT

140

INSUFFICIENT VIRTUAL STORAGE

141

STOW FAILED

142

NOTE FAILED

143

LRECL IS INVALID

144

BLKSIZE IS INVALID

145

RECFM IS UNDEFINED

146

LRECL IS UNDEFINED

147

BLKSIZE IS UNDEFINED

148

DDNAME MISSING

Adding a Member to a PDS

To add a member to the PDS

1. Allocate the PDS file with a DISP of MOD (see ALLOC()).
2. OBTAIN a DCB, specifying PDSOUT.
3. Open the file.
4. Use the PUT operand to write the desired records.
5. Use the STOW operand to create a directory entry. Do not specify a *ttr* value; the default is the first record written since the last STOW.
6. Use the TCLOSE operand to write a file mark and update all records to disk.
7. If you want to create additional members, you may continue as if the data set had just been opened.

Example

```
      /* DD name JUNK was already allocated */
dcb = sam('OBTAIN',,'JUNK','PDSIN')  /* Obtain a dcb for a PDS   */
sam('open',dcb) == ''                /* Open the PDS             */
sam('find',dcb,'QSGSS') == ''        /* Find the member QSGSS    */
sam('get',dcb) == 'The first record of the member' /* Get record */
      /* The next two calls are automatically done at IMOD end   */
sam('close',dcb)   == ''             /* Clean up                 */
sam('release',dcb) == ''             /* Clean up                 */
```

Appending Data to a PDS Member

If you consider the physical structure of a PDS, you will quickly see that it is not possible simply to add data to the end of a member. Such data would overlay the next physical member. However, the following procedure permits you to simulate appending data to a PDS member:

- Allocate the target member (OLDMEM) for input (DDNAME1).
- Using a different ddname (DDNAME2), allocate a new member (TEMPNAME) in the same data set.
- Read all records from DDNAME1 and write them to DDNAME2.

- Close DDNAME1.
- Write records to be appended to DDNAME2.
- Cause TEMPNAME to be added as member name.
- Delete OLDMEM from data set.
- Rename TEMPNAME to OLDMEM.

SAYWHAT()

Use SAYWHAT() to direct the output of TRACE and SAY operations to a data set.

Syntax

The SAYWHAT() function has this syntax:

Form 1:

dcb = SAYWHAT(OPEN,,[*lines*],[*output*],[*opt1*],[*head*],[*class*],[*opt2*],[*dest*],[*ddname*])

Form 2:

dcb = SAYWHAT(USE,*dcb*)

Form 3:

dcb = SAYWHAT(CLOSE,*dcb*)

Arguments

The SAYWHAT() function takes these arguments:

dcb

Value that identifies a file and an associated internal workspace. The DCB is returned by the OPEN operation and is required for all other functions.

OPEN

Allocates and opens a new file, and returns a *dcb*.

,,

Argument two is reserved and must be null.

USE

Indicates that the designated file is to be used for future SAY or TRACE output. You can use this argument to alternate output among multiple files.

Specifying SAYWHAT('USE',0) terminates directed SAY or TRACE output, and causes additional SAY or TRACE output to be printed in the ISERVE log file (ISRVLOG).

lines

Number of lines of data per page, not including lines used by the page headers. Null or zero indicates a page of infinite length. The default is 0.

output

Name of an OUTPUT JCL statement to be associated with this file.

opt1

Specify one of the following:

CC indicates that you will supply an ANSI carriage control character (1, ' ', 0, -, +) for each record you create with a SAY.

NOCC indicates that you will not supply an ANSI carriage control character for each record you create with a SAY. ANSI carriage control characters (single space) will be supplied by the system.

head

One or more strings to be used as page headings. Headings are separated from each other by a semicolon (;). Each heading must begin with an ANSI carriage control character. The first header must have a 1 (one) as its carriage control.

If you want one or more blank lines between the headings and data lines, you must explicitly code a blank heading line.

The following value may not exceed 256, not including semicolons:

(Total characters in headings) + (4*(number of headings)) + 2

To print the current page number, code a single occurrence of '@@@@' in your heading. This string will be replaced by the page number.

class

SYSOUT class to be assigned.

Default: The default class assigned by the operating system.

opt2

Specify one of the following:

HOLD indicates that the spun file is held.

NOHOLD indicates that the spun file will not be held; the default.

dest

User ID or *node.userid* to indicate the destination for routing the file.

ddname

DDname (supplied from JCL or the ALLOC() function) of the file to be used. Overrides the *output*, *class*, *hold*, and *dest* arguments.

CLOSE

Indicates the file is to be closed and spun.

Return Codes

The SAYWHAT() function produces these return codes:

-123

INVALID FILE DCB

101 - 110

ARG *n* MISSING OR INVALID

122

FILE FAILED TO OPEN

Example

```
ddname = alloc('RBROWNE.SAYOUT','SHR',,'DD1')  /* Output data set */
dcb = saywhat('open',,,,,,,,,ddname)      /* Tell SAYWHAT about it */
saywhat('use',dcb) == ''                  /* Switch SAY output */
say 'should go to RBROWNE.SAYOUT'
saywhat('use',0) == ''                    /* Switch back, and close file */
```

SCANSTR()

The SCANSTR() function scans a text string for complex patterns.

Syntax

The SCANSTR() function has this syntax:

{0|1} = SCANSTR(*pattern*,*string*,[*vldc*],[*fldc*])

Arguments

The SCANSTR() function takes these arguments.

0/1

If the specified pattern is present in the scanned string, a 1 is returned. If the string is not found, a 0 is returned.

pattern

Pattern character string to be searched for. The pattern is composed of any combination of characters and placeholders (VLDCs and FLDCs).

string

Text string to be scanned for the pattern.

vldc

(Variable-Length Do not-Care) A character that, when appearing in the pattern, will be taken to represent an arbitrary string of any length (including zero).

Default: & (ampersand)

fldc

(Fixed-Length Do not-Care) A character that, when appearing in the pattern, will be taken to represent exactly one arbitrary character.

Default: * (asterisk)

Usage Notes

All usage notes assume that the VLDC character is & and the FLDC character is *.

- Blanks are treated in the same manner as non-blanks, both in the pattern and in the scanned string.
- All scans are case-sensitive.
- Remember to use wildcard characters. *b* will not match *abc* but *&b&* will.
- For a pattern to match a string, all characters in the string must be accounted for in the pattern, including leading and trailing blanks.
- Since the VLDC character will expand to match any string, coding && is meaningless.
- Code *& to match a string of one or more characters, **& to match a string of two or more characters, etc.
- Note that the rules of pattern matching may yield unexpected results. A pattern of *x*&x* will match strings: *xxx*, *xyx*, *xxxxxxxxxx*, *xyxjlkjlkjlkjx*. A pattern of *&ab&c*d&* will match strings: *xxx ab c dxxx*, *xxxab cc abc dxx*, *aabbccdd*.
- If the FLDC or VLDC characters appear in the string to be scanned, they are treated as ordinary characters in the scanned string and placeholders in the pattern.

Return Codes

The SCANSTR() function produces these return codes:

101 - 104

ARG *n* MISSING OR INVALID

120

SCAN PATTERN TOO LONG OR COMPLEX

Example

```
/*  Unpack a julian date from ISPF statistics format */
1 == scanstr('&cat&','cats and dogs')
1 == scanstr('&c*t&','dogs and cats')
0 == scanstr('&Cat&','cats and dogs')
1 == scanstr('&CAT&',translate('Cats and Dogs'))
1 == scanstr('&cat&dog&','cats and dogs'
1 == scanstr('cat&d**s','cats and dogs')
1 == scanstr('&cat&dog&','complete catalog of doggerel')
```

SCRASID()

Use SCRASID() to specify that all WTOs from a particular address space be recorded on an ILOG file. You can also specify an IMOD to be executed for each WTO message.

Important! Use care when intercepting all WTOs from an address space, as this overrides other specifications that indicate specific WTO message IDs.

Syntax

The SCRASID() function has this syntax:

Form 1:

```
result = SCRASID(route,asid,action,num)
```

Form 2:

```
status = SCRASID(?,asid,action)
```

Arguments

The SCRASID() function takes these arguments:

result

Null string (or error text).

route

Specify one of the following:

ADD Adds a routing.

DELETE Deletes a routing.

asid

Address space ID (ASID) to be monitored.

action

Specify one of the following:

IMOD Triggers an IMOD.

ILOG Writes the message to an ILOG.

num

Number, from 0 through 14, that you assign. This number is used to construct the ILOG or IMOD name, as follows:

If ILOG was specified, the ILOG number is the value of the *num* argument.

If IMOD was specified, the IMOD name is ASID_*num*.

status

Value of *num*, as set by a previous use of the SCRASID() function.

?

Returns the current status of the ILOG or IMOD specification for the address space. A null return indicates that the address space is not being monitored.

Return Codes

The SCRASID() function produces these return codes:

101 - 104

ARG *n* MISSING OR INVALID

120

NO SUBCOM AREA

121

DOES NOT FUNCTION IN BATCH MODE

Example

```
scrasid('add','195','ilog',12)    /* Start logging */
scrasid('?','195','ilog') == 12   /* ILOG used for this ASID */
```

SETADDR()

Use the SETADDR() function to inspect and set the destination for ADDRESS commands and functions that may communicate with CA OPS/MVS Event Management and Automation.

Syntax

The SETADDR() function has this syntax:

value = SETADDR(OPS,[*identifier*])

Arguments

The SETADDR() function takes these arguments:

OPS

Stands for CA OPS/MVS Event Management and Automation.

value

Current setting of this identifier.

identifier

Subsystem ID assigned to the desired CA OPS/MVS Event Management and Automation address space.

Usage Notes

- Most values specified for *identifier* are case-sensitive.
- Since only one copy of the CA OPS/MVS Event Management and Automation communications module can be loaded into an ISERVE address space at one time, it is not possible to test different releases of CA OPS/MVS Event Management and Automation simultaneously from a single ISERVE address space.

Return Codes

The SETADDR() function produces these return codes:

101 - 102

ARG *n* MISSING OR INVALID

Example

```
ssid = setaddr('ops')          /* Obtain the current ssid  */
ssid = setaddr('ops','OPSX')   /* Set SSID for test system */
```

SECURITY()

Use the SECURITY() function to validate a user ID and password or to change an IMOD task's authority to that of a different user ID.

Syntax

The SECURITY() function has this syntax:

```
SECURITY({VERIFY|LOGON},userid,password,[newpass])
```

Arguments

The SECURITY() function takes these arguments:

VERIFY

Checks the user ID/password combination for validity.

LOGON

Checks the user ID/password combination for validity and, if valid, the IMOD task continues execution under the authority of the new user ID.

userid

User ID known to the system security software.

password

Password that authorizes the user ID.

newpass

New password. If both *userid* and *password* are correct, *newpass* replaces *password*.

Note: This change is permanently recorded by the system security software.

Usage Notes

- For LOGON processing to be meaningful, you must have specified the SECURITY SAF initialization parameter during ISERVE startup.
- To use this function, your installation must be using IBM's RACF security product or another product that supports the SAF RACROUTE interface.
- Following a successful LOGON attempt, the message "SRV331 *taskid* Task has logged on as user *userid*" is recorded in the ISRVLOG file.

Return Codes

The SECURITY() function produces these return codes.

101 - 104

ARG *n* MISSING OR INVALID

121

RACROUTE FAILED: *saf_rc* *racf_rc* *racf_re*

saf_rc The return code as set by SAF

racf_rc The return code as set by RACF

racf_re The reason code as set by RACF

Example

```
parse arg userid password
result = security('logon',userid,password)
 if rc ^= 0 then return 'LOGON REJECTED'
return 'LOGON SUCCESSFUL'
```

SETRC()

CA-GSS extended functions set the RC special variable upon their completion. This permits an IMOD to test for correct execution in a straightforward manner, either by checking for RC = 0 or by trapping failures with SIGNAL ON conditions. SETRC() permits an IMOD that is called as an external subroutine or function to set the RC special variable in the calling IMOD.

Syntax

The SETRC() function has this syntax:

oldval = SETRC([*newval*])

Arguments

The SETRC() function takes these arguments:

oldval

Previously set value for the RC special variable.

newval

Value to be assigned to the RC special variable upon return from the current external subroutine or function. Specify a value from 0 to 255 (inclusive).

Default: The value is not changed, and the currently set value is returned.

Usage Notes

- The value of RC in the currently executing IMOD will not be affected.
- If SETRC() is not used, RC will be set to zero upon completion of the external routine.
- SETRC() has no effect upon the value of RC upon completion of an internal routine.
- SETRC() should be issued immediately before RETURN or EXIT. If you call another IMOD after you issue a SETRC() the value will be canceled.

Example

```
#desc IMOD to check for palindromes
#callable
   parse upper value arg(1) arg1
   parse upper value arg(2) arg2
   arg2 = reverse(arg2)
   if arg1 = arg2 then return 'PALINDROME'
   x = stern(200)
   return 'NOT PALINDROME'
```

SHOVE()

Use the SHOVE() function to insert records into any stack belonging to any IMOD. Inserted records can be placed at any position within the stack.

Restriction

SHOVE() can specify a stack external to an IMOD only when the target stack has been declared public for writing.

Syntax

The SHOVE() function has this syntax:

```
SHOVE([record],[recnum],[stack],[owner],,[postoption])
```

Arguments

The SHOVE() function takes these arguments:

record

Record to be inserted into the stack.

Default: A zero-length record is added.

recnum

Record number that is to be assigned to the inserted record. Existing records of *recnum* or higher have their record numbers increased by one. The lowest record number is 1. Record 1 is the first record fetched by PULL. If the record number exceeds the total records currently on the stack, a QUEUE is performed.

stack

Number of the stack to be referenced.

Default: If a local stack is referenced, the current stack (as set by the SWAPSTAK() function); otherwise, the value is set by the owning IMOD task using the PUBSTACK(DEFAULT) function.

owner

IMOD ID of the stack's owner (for external reference). This value can be obtained from the variable imod_id in the owning IMOD task. It is also the value returned by the SPAWN() function.

Default: Current IMOD task.

postoption

Wait option, which determines whether the owning IMOD waits for a record to appear on an external stack. Specify one of these values:

POST
Any outstanding wait is satisfied; the default.

NOPOST
An outstanding wait is not satisfied.

ONLYPOST
Any outstanding wait is satisfied, but a record is not placed on the stack.

Note: If you are placing several records onto an external stack, CA recommends that you specify NOPOST for all but the final record. IMODs are time-sliced, and if the POST operation is performed before the final record is shoved on the stack, the IMOD task owning the stack could be dispatched to process before the final SHOVE is completed.

Return Codes

The SHOVE() function produces these return codes:

101 - 106

ARG *n* INVALID

122

STACK NOT SPECIFIED AND NO DEFAULT

123

SPECIFIED STACK DOES NOT EXIST

125

SPECIFIED IMOD NOT FOUND

126

STACK WRITE FAILED

127

REQUESTED STACK DOES NOT EXIST

128

SPECIFIED IMOD/STACK NOT AUTHORIZED

Example

```
x = shove(rec,5)              /* A record is inserted into the
                                 current stack */
x = shove(rec,5,2)            /* A record is inserted into local
                                 stack 2 */
x = shove(rec,5,,453,,'post') /* A record is inserted into IMOD
                                 453's default stack. Any
                                 outstanding wait is marked
                                 complete. */
```

SORT()

Use the SORT() function to sort records on any stack belonging to any IMOD. Up to six fields can be sorted on simultaneously, with the sorting precedence running from left to right.

Restriction

SORT() can specify a stack external to an IMOD only when the target stack has been declared public for writing.

Syntax

The SORT() function has this syntax:

```
SORT([stack],[owner],[order,field,leng],...,[order,field,leng])
```

Arguments

The SORT() function takes these arguments:

stack

Number of the stack to be referenced.

Default: If a local stack is referenced, the current stack (as set by the SWAPSTAK() function); otherwise, the value is set by the owning IMOD task using the PUBSTACK(DEFAULT) function.

owner

IMOD ID of the stack's owner (for external reference). This value can be obtained from the variable *imod_id* in the owning IMOD task. It is also the value returned by the SPAWN() function.

Default: Current IMOD task

order

Specify one of the following:

AP Ascending sort; field specified by column number.

DP Descending sort; field specified by column number.

field

Defines the start of a field. For column-specified fields (type = AP and DP), this is the column number of the start of the field.

leng

Length of a sort field. For column-specified fields (type = AP and DP), this is the number of characters in the field.

Return Codes

The SORT() function produces these return codes:

101 - 120

ARG *n* MISSING OR INVALID

122

STACK NOT SPECIFIED AND NO DEFAULT

123

SPECIFIED STACK DOES NOT EXIST

125

SPECIFIED IMOD NOT FOUND

126

STACK WRITE FAILED

127

REQUESTED STACK DOES NOT EXIST

128

SPECIFIED IMOD/STACK NOT AUTHORIZED

Example

```
x = sort(0,2,'AP',7,10,'AP',1,6)
                   /* Sort IMOD 2's stack 0. Sort the records
                      by the 10-character field starting in
                      column 7. Within equal records, sort on
                      the 6-character field beginning in
                      column 1. */
```

SPAWN()

Use the SPAWN() function to start the execution of another IMOD task. Spawned IMODs run concurrently with the spawning IMOD task.

Important! Do not create a situation where an IMOD task spawns itself. An unlimited number of IMOD tasks may be created, swamping the ISERVE address space.

Syntax

The SPAWN() function has this syntax:

```
imod_id = (SPAWN,imodname,[{COPYQ|NEWQ|PASSQ}],[args])
```

Arguments

The SPAWN() function takes these arguments:

imod_id

ID assigned to the new IMOD task. This IMOD ID is a unique number that can be used to reference an executing IMOD task. It is used to access the IMOD task's stacks externally and as an argument in the command CANCEL IMOD.

imodname

Name of the IMOD to be executed. This IMOD must be marked as CALLABLE by including the #CALLABLE compiler directive in the IMOD's source code.

COPYQ

Starts with a copy of the current stack. If NEWSTACK has been used, only the current stack is copied.

NEWQ

Starts with a new, empty stack. This is the default.

PASSQ

Transfers the existing stack to the new IMOD. If NEWSTACK has been used, only the current stack is transferred.

args

Text string, to be passed to the spawned task as an initial argument string.

Usage Notes

- If you want a value returned from another IMOD, use CALL instead of SPAWN().
- If an IMOD that was triggered from a WTO spawns another IMOD, the WTO's ILOG routing code, if any, is inherited. This may cause multiple copies of the WTO text to be recorded on the ILOG. If this is not desirable, use IROUTE() to suppress ILOG recording of either the original or spawned message text.

Return Codes

The SPAWN() function produces these return codes:

40

INCORRECT CALL TO ROUTINE (NOT CALLABLE)

40

INCORRECT CALL TO ROUTINE (NOT ACTIVE)

40

INCORRECT CALL TO ROUTINE (NOT ALLOWED)

101 - 103

ARG *n* MISSING OR INVALID

121

REQUESTED IMOD NOT FOUND

Example

```
queue 'Some data to work on'     /* Get data ready  */
spawn(async_work,'PASSQ') == '00000016'
                                 /* Returns IMOD internal name */
```

SRVCALL()

Use the SRVCALL() function to execute an IMOD as a subroutine in another ISERVE address space. SRVCALL() operates like a standard REXX CALL instruction. An argument string can be passed, a result string returned, and the current stack can be passed to the called routine. The modified stack is returned to the caller.

SRVCALL is TSO/E REXX-compatible and can also be used under SRVBATCH. It should not be used in an IMOD already running in an ISERVE address space.

Syntax

The SRVCALL() function has this syntax:

```
result = SRVCALL([node],[ssid],imod,[{PASS|NONE}],[args])
```

Arguments

The SRVCALL() function takes these arguments:

result

If the operation was successful, this is the value specified by the last-executed REXX RETURN or EXIT instruction in the called routine. Otherwise, error text is returned. The returned text is always padded on the left with one extra character. If no system-related errors occurred, this character is a blank. If the returned string is an error message indicating that linkage to the requested IMOD was not performed, this character is an asterisk (*).

node

Name of the GoalNet node where the IMOD is to be executed.

Default: Current node.

ssid

Subsystem ID that identifies the CA-GSS that is to execute the IMOD.

If *ssid* is specified without *node*, the request is routed to the appropriate CA-GSS on the same z/OS system.

If *ssid* is specified with *node*, the request is first routed to the CA-GSS identified by *ssid* on the same system as the requester. That CA-GSS system then routes the request to the CA-GSS at *node*.

imod

Name of the IMOD that is to be executed as an external routine. This argument is required.

PASS

Passes the contents of the current stack (up to the NEWSTACK marker) to the called routine. When the information is passed, it is deleted from the caller's stack. Upon return, any records left on the stack by the called IMOD are returned to the caller. These records become the new stack contents.

NONE

Specifies that no stack information is to be passed to the called routine. This is the default. Any returned stack data is appended to the current stack contents. Select this option whenever records exist on the current stack and the called IMOD is not going to process them.

args

String passed to the called routine as an argument string.

Usage Notes

- Only a single argument can be passed, although this argument can have many fields.
- Only the passed stack can be referenced by the called IMOD. If the called IMOD creates additional stacks using SWAPSTAK() or NEWSTACK(), only the current stack is returned to the caller at the conclusion of the called IMOD.
- Do not use this function from the ISERVE address space. It was designed for use in a single-user environment (SRVBATCH or TSO/E REXX). Other use could create severe processing bottlenecks.
- To use this function from SRVBATCH, you must provide a FUNCTION control card in the PARMLIB file.
- To use this function from TSO/E REXX, the SRVCALL load module must be accessible using the JOBLIB, STEPLIB or LINKLIST concatenation.
- Do not pass stack records to an IMOD if they are not required. Although unread records are passed back to the caller's stack, this results in considerable (and avoidable) overhead.
- SRVCALL() should not be executed from CA OPS/MVS Event Management and Automation. Instead, use the SRVOPS() function.

Return Codes

TSO/E REXX has no provision for the setting of return codes by functions. All error text generated by SRVCALL and its associated components set an asterisk in the first position. All non-error text (that returned by a successfully executed IMOD) has a single blank as the first character.

Example

```
x = srvcall('node2',,'status',,args)
                                /* Run "status" IMOD at node2 */
if substr(x,1,1) == '*' then signal error1
                                /*  Check for error condition */
```

SRVOPS()

Use the SRVOPS() function to execute an IMOD as a subroutine in an ISERVE address space. SRVOPS() operates like a standard REXX CALL instruction. An argument string can be passed, a result string returned, and the current stack can be passed to the called routine. The modified stack is returned to the caller.

Restriction

SRVOPS() should only be used in execs running under CA OPS/MVS Event Management and Automation.

Syntax

The SRVOPS() function has this syntax:

```
result = SRVOPS([node],[ssid],imod,[{PASS|NONE}],[args])
```

Arguments

The SRVOPS() function takes these arguments:

result

Value specified by the last-executed REXX RETURN or EXIT instruction in the called routine (or error text). The returned text is always padded on the left with one extra character. If no system-related errors occurred, this character is a blank. If the returned string is an error message indicating that linkage to the requested IMOD was not performed, this character is an asterisk (*).

node

Name of the GoalNet node where the IMOD is to be executed. The current node is the default.

ssid

Subsystem ID that identifies the CA-GSS that is to execute the IMOD.

If *ssid* is specified without *node*, the request is routed to the appropriate CA-GSS on the same z/OS system.

If *ssid* is specified with *node*, the request is first routed to the CA-GSS identified by *ssid* on the same system as the requester. That CA-GSS system then routes the request to the CA-GSS at *node*.

imod

Name of the IMOD to be executed as an external routine; required.

PASS

Passes the contents of the current stack (up to the NEWSTACK marker) to the called routine. When the information is passed, it is deleted from the caller's stack. Upon return, any records left on the stack by the called IMOD are returned to the caller. These records become the new stack contents.

NONE

Specifies that no stack information is to be passed to the called routine. This is the default. Any returned stack data is appended to the current stack contents. Select this option whenever records exist on the current stack and the called IMOD is not going to process them.

args

String passed to the called routine as an argument string.

Usage Notes

- Use of SRVOPS() requires CA OPS/MVS Event Management and Automation.
- When testing this function, do NOT name your test exec SRVOPS. Otherwise, your exec will recursively call itself -not the SRVOPS() function- until storage has been exhausted.
- Only a single argument can be passed, although this argument can have many fields.
- Only the passed stack can be referenced by the called IMOD. If the called IMOD creates additional stacks using SWAPSTAK() or NEWSTACK(), only the current stack is returned to the caller at the conclusion of the called IMOD.
- Do not use this function from the ISERVE address space.
- Do not pass stack records to an IMOD if they are not required. Although unread records are passed back to the caller's stack, this results in considerable-and avoidable- overhead.
- CA OPS/MVS Event Management and Automation uses a zero-length stack record to indicate end-of-file. Do not attempt to embed a zero-length stack record in the stack to be passed to CA-GSS.

Return Codes

TSO/E REXX has no provision for the setting of return codes by functions. All error text generated by SRVOPS and its associated components set an asterisk in the first position. All non-error text (that returned by a successfully executed IMOD) has a single blank as the first character.

Example

```
x = SRVOPS('node2',,'status',,args)         /* Run "status" IMOD
                                               at node2 */
if substr(x,1,1) == '*' then signal error1  /* Check for error
                                               condition */
```

STACKINF()

Use the STACKINF() function to return and modify information about the current contents and status of a stack.

Syntax

The STACKINF() function has this syntax:

Form 1:

```
value = STACKINF(return,[stack],[owner])
```

Form 2:

```
curvalue = STACKINF(MAXBYTE,[stack],[owner],[kbytes])
```

Form 3:

```
acctype = STACKINF(ACCESS,[stack],[owner])
```

Arguments

The STACKINF() function takes these arguments:

value

Requested information (or error text).

return

Specify one of these values:

QSTACK

Returns the current NEWSTACK number. This is the number of times that the NEWSTACK
command has been issued against this stack less the number of times the DELSTACK command was issued.

QBUF

Returns the current buffer number. This is the number of times that the MAKEBUF command has been issued against this stack less the number of times the DROPBUF command was issued.

QELEM

Returns the number of stack records in the current buffer, as determined by the MAKEBUF command.

QUEUED

Returns the total elements on the stack. This option functions exactly like the QUEUED() function.

STACKS

Returns a string consisting of all currently existing stack numbers. The numbers are blank-delimited, and are listed in ascending order.

stack

Number of the stack to be referenced.

Default: If *stack* is omitted and a local stack is referenced, the current stack (as set by SWAPSTAK()). If *stack* is omitted and an external stack is referenced, the value set by the owning IMOD task using the PUBSTACK(DEFAULT) function.

owner

Stack's current size limit in k-bytes. If *kbytes* is omitted, curvalue returns the current setting. If *kbytes* is specified, curvalue equals *kbytes*. To convert *curvalue* to bytes, multiply by 1024.

curvalue

Stack's current size limit in k-bytes. If *kbytes* is omitted, curvalue returns the current setting. If *kbytes* is specified, curvalue equals *kbytes*. To convert *curvalue* to bytes, multiply by 1024.

MAXBYTE

Returns or sets the maximum amount of data that may be contained in the stack. Including *kbytes* sets the amount.

kbytes

Maximum storage that may be occupied by the stack. This value is specified in k-bytes and is multiplied by 1024 to obtain the actual storage limit.

acctype

One of these values is returned:

PRIVATE

The stack is private.

PUBLIC

The stack is fully accessible for external operations.

QUEUE

The stack can have data queued to it externally. This is a subset of WRITE.

READ

The stack can be read externally.

RQUEUE

The stack can be read from and queued to by external functions.

WRITE

The stack can be written to externally. This also implies QUEUE.

ACCESS

Returns the access status, as set by PUBSTACK().

Return Codes

The STACKINF() function produces these return codes:

101 - 104

ARG *n* MISSING OR INVALID

125

SPECIFIED IMOD NOT FOUND

126

STACK NOT SPECIFIED AND NO DEFAULT

127

REQUESTED STACK DOES NOT EXIST

Example

```
recs = stackinf('queued')          /* Determine total stack
                                      records */
status = stackinf('access',,123)   /* Determine public status of
                                      default stack owned by IMOD
                                      123 */
list = stackinf('stacks')          /* Obtain list of all active
                                      stacks */
list = stackinf('stacks',,123)     /* Obtain list of all active
                                      stacks belonging to IMOD
                                      123 */
```

SUBMIT()

Use the SUBMIT() function to submit a job stream to the JES2 internal reader.

Syntax

The SUBMIT() function has this syntax:

```
SUBMIT([stem.])
```

Arguments

The SUBMIT() function takes this argument:

stem.

Root name of a stem variable containing the card images to be submitted, including the period. This should be enclosed in quotation marks.

Usage Notes

- If you omit the *stem* argument, the contents of the current stack are sent to the internal reader.
- The stem variables cannot be global; that is, they cannot begin with an ampersand (&). For information on global variables, see the *Administration Guide*.
- The card images are processed in the order the stem variables were created.
- To reuse stem variables (with fewer lines), use the REXX DROP command for the stem. This deletes the entire stem group and allows you to create a set with fewer variables.

Return Codes

The SUBMIT() function produces these return codes:

101

One of the following:

ARG 1 MISSING OR INVALID

STEM VARIABLE NOT FOUND

STEM HAS NO MEMBERS

121

SVC 99 allocation error in the following form:

RC: *returncode* ERROR: *errorcode* INFO: *informationcode*

122

FILE FAILED TO OPEN

Example

```
queue '//RBROWNEX JOB ,BR14,CLASS=A'
queue '//STEP1    EXEC PGM=IEFBR14'
submit() == ''              /* Could also use a stem variable */
```

SWAPSTAK()

Use the SWAPSTAK() function to create and switch between multiple numbered stacks. Once a stack is made current by the SWAPSTAK() function, all native REXX and TSO/E emulation stack commands operate normally.

Syntax

The SWAPSTAK() function has this syntax:

stack = SWAPSTAK([*n*|NEW|DELETE])

Arguments

The SWAPSTAK() function takes these arguments:

stack

Error text is returned if the operation is not successful. Otherwise, the current stack number is returned.

n

Desired stack number. You can specify any number between 0 and 231-1. If the specified stack does not exist, it is created. Although the total number of stacks that an IMOD can maintain is limited only by available memory, large numbers of stacks (over 25, for example) compromise stack-switching efficiency.

NEW

Beginning with stack 0, examines each stack until one is found that has a status of "never-used" (the status of a stack that has never been referenced, a stack that has been explicitly destroyed with a SWAPSTAK(DELETE) operation). This stack is then assigned a status of "empty" and it is made the current stack. This is useful for subroutines that desire a temporary workspace.

DELETE

Deletes the currently accessed stack and returns it to "never-used" status. Stack 0 becomes the current stack. If you delete stack 0, it is automatically recreated as an empty stack.

Usage Notes

- When an IMOD task is first created, SWAPSTAK(0) is in effect.
- Stacks created by SWAPSTAK() are complete stack structures and can contain both NEWSTACK and MAKEBUF markers, as placed by the NEWSTACK and MAKEBUF commands.
- The current stack setting is retained during CALL and RETURN operations.
- If the argument is omitted, the stack number is left unchanged and its current value is returned.
- All stack functions except QUEUED() may operate on any stack by specifying its number as an argument. All other methods for operating on stacks that do not provide a way of specifying the stack's number can only be used if the stack has been made the current stack by use of SWAPSTAK().
- There is a subtle but important difference between an empty stack and a never-used stack. An empty stack exists, but has no records. A never-used stack has no existence. Requesting a NEW stack with SWAPSTAK() will never return a stack that was not previously in never-used status (The SWAPSTAK(NEW) operation itself changes the stack's status to empty. If you want to destroy a stack, use the DELETE argument. This will both discard any remaining records and return the stack to never-used status.

Return Codes

The SWAPSTAK() function produces these return codes:

101

ARG 1 MISSING OR INVALID

Example

```
/* This code segment can be used to obtain a work stack */

ostack = swapstak(  )          /* Save current stack number      */
x      = swapstak('new')       /* Obtain new stack for workspace */
x      = swapstak('delete')    /* Delete workspace stack         */
x      = swapstak(ostack)      /* Restore original stack number  */
```

SYSVAL()

Use the SYSVAL() function to gain access to various system values.

Syntax

The SYSVAL() function has this syntax:

```
value = SYSVAL({SYSID|SYSNAME|SSID})
```

Arguments

The SYSVAL() function takes these arguments:

value

Returned value, if the function was successful. Otherwise, error text is returned.

SYSID

Requests the SMF system ID.

SYSNAME

Requests the GRS system name.

SSID

Requests the subsystem name being used by ISERVE.

Return Codes

The SYSVAL() function produces these return codes:

101

ARG 1 MISSING OR INVALID

Example

```
sysname = sysval('SYSNAME')      /* Obtain the GRS SYSNAME */
```

TUG()

Use the TUG() function to obtain a copy of any record on any stack belonging to any IMOD task. A TUG() operation provides you with a copy of a record. Following a TUG(), the record remains on the stack.

Syntax

The TUG() function has this syntax:

record = TUG([*recnum*],[*stack*],[*owner*],[*cntl*])

Arguments

The TUG() function takes these arguments:

record

Returned record or error text.

recnum

Record number to be copied from the stack.

stack

Number of the stack to be referenced.

Default: If a local stack is referenced, the current stack (as set by the SWAPSTAK() function); otherwise, the value is set by the owning IMOD task using the PUBSTACK(DEFAULT) function.

owner

IMOD ID of the stack's owner (for external reference). This value can be obtained from the variable *imod_id* in the owning IMOD task. It is also the value returned by the SPAWN() function.

Default: Current IMOD task.

cntl

Name of a variable to receive the control information stored with the record.

Default: Control information is not retrieved.

Return Codes

The TUG() function produces these return codes:

101 - 105

ARG *n* INVALID

122

STACK NOT SPECIFIED AND NO DEFAULT

123

SPECIFIED STACK DOES NOT EXIST

125

SPECIFIED IMOD NOT FOUND

126

STACK WRITE FAILED

127

REQUESTED STACK DOES NOT EXIST

128

SPECIFIED IMOD/STACK NOT AUTHORIZED

Example

```
rec = tug(2)              /* Record number 2 is copied from the
                             current stack */
rec = tug(2,5)            /* Record number 2 is copied from local
                             stack 5 */
rec = tug(2,,453,'data')  /* Record 2 is copied from IMOD 453's
                             default stack and control information
                             is placed in the variable data. */
```

Usage Note

TUG() can specify a stack external to an IMOD task only when the target stack has been declared public for reading by the owning task.

UNIQUE()

Use the UNIQUE() function to scan a string for a character that does not appear in the string. This character can be used as a delimiter.

Syntax

The UNIQUE() function has this syntax:

```
char = UNIQUE(string,[prefer],[EXTEND])
```

Arguments

The UNIQUE() function takes these arguments.

char

Single character that does not appear in *string*.

string

String to be examined.

prefer

List of preferred characters. If specified, the left-most unique character is returned. The default is to use an internal preference.

EXTEND

If specified and none of the characters specified in *prefer* is unique, then the preference list will be extended to all 256 possible characters.

Usage Notes

- The internally generated preference string is weighted to non-alphanumeric, printable characters. Once these are exhausted, the remainder of the 256 possible characters are used.
- Use of EXTEND simply reexecutes the function with the internally generated preference list.

Return Codes

The UNIQUE() function produces these return codes:

101 - 103

ARG n MISSING OR INVALID

121

NO UNIQUE CHARACTER

Example

```
/*  Return 3 values on the Stack */
   a = .....   /* arbitrary values assigned */
   b = .....
   c = .....
   d = unique(a||b||c)
   queue d||a||d||b||d||c||d
/*  Retrieve the values from the Stack */
   parse pull 1 d +1 a (d) b (d) c (d)
```

VARSIZE()

Use the VARSIZE() function to set and return the maximum length of a REXX variable string. Typically, REXX variable strings have a maximum allowed length of 4096 bytes. Use VARSIZE() to specify a larger value. This specification applies only to the IMOD in which you set it. The value is not propagated across the CALL/RETURN interface.

Important! Be careful when specifying a larger value. Error conditions such as unplanned loops can temporarily deplete storage.

Syntax

The VARSIZE() function has this syntax:

cursize = VARSIZE([*value*])

Arguments

The VARSIZE() function takes these arguments.

cursize

Current or new size limit (or error text).

Value

Maximum length to be permitted for a REXX variable string.

Default: The current value is returned and left unchanged.

Return Codes

The VARSIZE() function produces these return codes:

101

ARG 1 MISSING OR INVALID

Example

```
varsize() == '32767'   /* Current maximum variable length  */
```

VSAM()

Use the VSAM() function to perform I/O operations on VSAM data sets.

Syntax

The VSAM() function has this syntax:

Form 1:

```
acb = VSAM(OBTAIN,[scope],cbtype)
```

Form 2:

```
VSAM(option1,acb)
```

Form 3:

```
status = VSAM(STATUS,acb)
```

Form 4:

```
VSAM(MODCB,acb,modopt1)
```

Form 5:

```
VSAM(MODCB,rpl,modopt2)
```

Form 6:

```
dsname = VSAM(DSNAME,acb)
```

Form 7:

```
VSAM(POINT,rpl,key)
```

Form 8:

```
record = VSAM(act1,rpl,[key])
```

Form 9:

```
VSAM(act2,rpl,record)
```

Form 10:

```
VSAM(act3,rpl)
```

Arguments

The VSAM() function takes these arguments:

acb

ID for a VSAM ACB and an associated internal workspace.

OBTAIN

Obtains a new RPL or ACB.

scope

Specify one of these values:

LOCAL

Indicates that the file is local to the IMOD. The RPL or ACB is automatically released at the end of the IMOD.

GLOBAL

Indicates that you can share this file across IMODs. The RPL or ACB must be released explicitly.

Default: LOCAL

cbtype

Specify one of these values:

ACB

Obtains an ACB control block.

RPL

Obtains an RPL control block.

option1

Specify one of these values:

OPEN

Opens a file for processing.

CLOSE

Completes processing.

TCLOSE

Temporarily closes the file. This option writes all information to disk and updates all pointers. However, the file remains open and processing may continue.

RELEASE

Returns handle value to the system to free storage used.

status

Current status of data set (OPEN or CLOSED) as returned by the STATUS operation.

STATUS

Returns the status of the ACB.

MODCB

Modifies the VSAM RPL or ACB.

modopt1

Specify one of these values:

DDNAME=ddname

Modifies the ACB's DDNAME value. Only effective if the data set is currently closed.

MACRF=opt

Modifies the ACB's MACRF values. Choose one or more of the following values, separated by commas: SEQ, OUT, RST, NRS, DIR, KEY, IN. Descriptions of these options can be found in the IBM VSAM reference manuals.

rpl

ID for a VSAM RPL and an associated internal workspace.

modopt2

Specify one of these values:

ACB=acb

Modifies the RPL's ACB value. Before you can use an RPL for operations on a data set, it must be linked to an open ACB.

KEY=key

Modifies the RPL's KEY field. You may set this field directly in the RPL without using the POINT, GET, or DELETE operations.

OPTCD=opt

Modifies the RPL's OPTCD values. Choose one or more of these values, separated by commas: FWD, BWD, SEQ, DIR, KEY, FKS, GEN, UPD, NUP, NSP, KEQ, KGE.

dsname

Data set name of an open data set. This is the cluster name.

DSNAME

Returns the cluster name of an open data set.

POINT

Useful in sequential mode, a POINT operation positions the file to the record that matches *key*. Subsequent GETs then retrieve records, beginning with the desired one.

key

Key of the desired record. For KSDS processing, the full key or partial key, and the match must be exact or not less than the key value, depending on the processing mode. For RRDS and ESDS processing, key is the numeric relative record number or RBA, respectively.

record

Record text. For keyed records, this includes the key.

act1

Specify one of these values:

GET

Obtains a record. For sequential mode processing, returns next logical record. For direct mode processing, returns record whose key is in the RPL. Direct mode processing permits specification of the key with GET.

DELETE

Deletes the specified record (by key). This is an extension of VSAM functions. Normally, you must first read the record to be erased and then erase it. DELETE eliminates the need to perform the GET.

act2

Specify one of these values:

PUT

Writes a record. Depending upon the update status, a new record is inserted or the current record is replaced. For RRDS and ESDS processing, the key argument is required for a non-update write. For KSDS processing, the key is embedded in the record.

UPDATE

Replaces the specified record (by key). This is an extension of VSAM functions. Normally, you must first read the record to be updated and then replace it. UPDATE eliminates the need to perform the GET. If the record to be updated does not exist, it will be added.

act3

Specify one of these values:

ERASE

In UPDATE mode, causes the last read record to be deleted from the file.

ENDREQ

Terminates any operation in progress on an RPL. For example, if you have read a record for update and then change your mind, you must issue an ENDREQ to free the RPL. Otherwise, you must either update or erase the record to complete the operation.

Return Codes

The VSAM() function produces these return codes:

101 - 104

ARG *n* MISSING OR INVALID

122

DELETE NOT MATCHED

124

ADD AND UPDATE FOR RECORD BOTH FAILED

126

NOT RPL

127

NO RPL

128

NOT ACB

129

NO ACB

130

DATASET IS OPEN

131

DATASET NOT OPEN

135

PHASE: *phase* RETURN: *ret* REASON *rea*

This is returned for an RPL-based error. *phase* indicates the failing operations (for example, GET), *ret* is the value returned in the RPLRET field, and *rea* is the value returned in the RPL FDBK2 field. These values are explained in detail in the IBM VSAM reference manuals.

136

PHASE: *phase* R15: *ret* REASON: *rea*

This is returned for a non-RPL-based error. *phase* indicates the failing operation (for example, CLOSE), *ret* is the value returned (by VSAM) in register 15, and *rea* is the value returned (by VSAM) in register 0. These values are explained in detail in the IBM VSAM reference manuals.

137

OPEN ERROR. CODE=*code*

code is the VSAM error code, as explained in the IBM VSAM reference manuals.

Example

```
/* DDname JUNK was already allocated */
acb = vsam('OBTAIN','ACB') /* Obtain an ACB */
rpl = vsam('OBTAIN','RPL') /* Obtain an RPL */
x   = vsam('MODCB',acb,'DDNAME=JUNK','MACRF=SEQ,DIR,OUT')
                           /* Modify the ACB for the desired
                              file*/
x = vsam('MODCB',rpl,'ACB='||acb)
                           /* Point the RPL back to the ACB */
x = vsam('OPEN',acb)       /* Open the file */
x = vsam('MODCB',rpl,'OPTCD=DIR,FKS,KEQ,UPD')
                           /* Modify the RPL to perform direct
                              I/O,full key,key equal,and fetch
                              for update */
record = vsam('GET',rpl,key)
                           /* Fetch the record that matches the
                              value found in "key" */
record = key||newdata      /* construct new record, retaining the
                              key */
x = vsam('PUT',rpl,record) /* Write the updated record */
x = vsam('CLOSE',acb)      /* Close the file */
                           /* The next two calls are automatic at
                              IMOD end */
vsam('release',acb) == ''  /* Clean up  */
vsam('release',rpl) == ''  /* Clean up  */
```

VVALUE()

Use the VVALUE() function to retrieve and set values for variables that exist in the external REXX subroutines that called the current routine.

Syntax

The VVALUE() function has this syntax:

```
value = VVALUE({FETCH|STORE},varname,level,[newvalue])
```

Arguments

The VVALUE() function takes these arguments:

value

Value of the specified variable, if this is a fetch operation. Otherwise *null* or error text.

FETCH

Returns the current contents of a variable.

STORE

Replaces the variable's contents with the specified value.

varname

Name of the variable to be referenced.

level

Subroutine nesting level to be accessed. 0 is the currently executing routine, 1 is the caller of the currently executing routine, 2 is the caller of the caller of the currently executing routine, etc.

newvalue

Value to be placed in the specified variable (STORE operation only).

Usage Notes

- VVALUE() works only with external subroutines. To share variables among internal subroutines, use the EXPOSE operand of the PROCEDURE statement.
- VVALUE() will not work when the calling routine is executing at another GoalNet node.
- You can use VVALUE() to allow a called subroutine to preserve values between calls.

Return Codes

The VVALUE() function produces these return codes:

101 - 104

ARG *n* MISSING OR INVALID

103

ARG 3 EXCEEDS NESTING LEVEL

121

SYSTEM ERROR

122

STORAGE DEPLETED

123

UNKNOWN SEVERE ERROR

124

SYSTEM ERROR, INVALID RETURN

125

VARIABLE DID NOT EXIST OR IS DROPPED

126

LAST VARIABLE TRANSFERRED ("N")

127

TRUNCATION OCCURRED

128

INVALID VARIABLE NAME

129

INVALID VALUE SPECIFIED

130

INVALID FUNCTION CODE (SHVCODE)

Example

```
/*  This IMOD gets called repeatedly.  It requires extensive
    initialization to derive a value for "num".  Instead of
    performing this initialization each call, the value of "num"
    is stored in the calling IMOD.  */

num = vvalue('fetch','shar_init',1)           /* obtain stored
                                                 number         */
if rc ^= 0 | datatype(num,'w') == 0 then do   /* nothing stored */
   call init     /* invoke 10,000 line internal init subroutine */
   x = vvalue('store','shar_init,1,num)       /* save for next
                                                 time           */
   end
```

WAIT()

Use WAIT() to direct an IMOD to wait for a POST operation on a stack.

Syntax

The WAIT() function has this syntax:

stknum = WAIT(*n*,[*n*],[*n*],...,[*n*])

Arguments

The WAIT() function takes these arguments:

stknum

Number of the stack that satisfied the requirements for ending the wait (or error text).

n

Number of a stack to wait on. Up to twenty stacks can be waited on. All specified stacks must be empty or WAIT() completes immediately. If multiple stacks are specified, a POST operation issued against any stack causes WAIT() to complete.

Return Codes

The WAIT() function produces these return codes:

101 - 120

ARG *n* MISSING OR INVALID

121

NO STACKS SPECIFIED

Example

```
num = wait(1,2,3)   /* Wait for a record to appear on any of 3 stacks    */
rec = pull(num)     /* Read the record that caused the wait  to complete */
```

Usage Notes

When a WAIT() is issued against one or more stacks, each specified stack is inspected for existing records. If any records are found, no wait is performed. Following completion of the function, the number of the first non-empty stack is returned as the function's result.

When a WAIT() is issued against empty stacks, the IMOD task is suspended. Execution is resumed when another IMOD task uses a function that specifies post, such as PUSH(), QUEUE(), and SHOVE(). The function providing the post may provide accompanying stack records.

WTO()

Use the WTO() function to issue the specified text as a WTO.

Syntax

The WTO() function has this syntax:

```
msgid = WTO(text,[rout],[desc],[cons])
```

Arguments

The WTO() function takes these arguments:

msgid

Unique message ID assigned by the operating system. Can be used with the DOM() function to delete messages no longer needed. If the WTO fails, error information is returned.

text

Text to appear on the operator's console as a WTO.

rout

One or more routing codes (like '1' or '1,2,3' or '1-5,8,10,15-128'). For information on routing codes, see the IBM manual *Supervisor Services and Macro* Instruction.

desc

One or more descriptor codes (like '1' or '1,2,3' or '1-5,8,10,12-16'). For information on descriptor codes, see the IBM manual *Supervisor Services and Macro Instructions*.

cons

Number of console to receive message. If the IMOD is running as the result of an operator command, the predefined variable *imod_console* contains the console ID of the console issuing the command.

Return Codes

The WTO() function produces these return codes:

101 - 102

ARG *n* MISSING OR INVALID

Example

```
msgid = wto('operator message',2,2) == '' /* Master console and
                                             highlight         */
x = dom(msgid)                           /* Delete the message */
```

WTOR()

Use the WTOR() function to display text on the operator console. WTOR() waits for the console operator's reply and returns it.

Syntax

The WTO() function produces these return codes:

101 - 102 ARG *n* MISSING OR INVALID

Example

```
msgid = wto('operator message',2,2) == '' /* Master console and
                                             highlight          */
x = dom(msgid)                           /* Delete the message */
```

Arguments

The WTOR() function takes these arguments:

reply

Reply from the operator (or error text).

text

Text to appear on the operator's console as the WTOR.

rout

One or more routing codes (like'1' or '1,2,3' or '1-5,8,10,15-128'). For information on routing, see the IBM manual *Supervisor Services and Macro Instructions*.

desc

One or more descriptor codes (like '1' or '1,2,3' or '1-5,8,10,12-16'). For information on routing and descriptor codes, see the IBM manual *Supervisor Services and Macro Instructions*.

cons

ID of the console to display the message. If the IMOD is being executed as the result of an operator's command, the *imod_console* variable will contain the console ID where the command was issued.

Usage Notes

- WTOR() suspends execution of IMOD until an operator replies to the displayed message.
- An IMOD with an outstanding WTOR cannot be canceled until the WTOR is satisfied.

Return Codes

The WTOR() function produces these return codes:

101

ARG 1 MISSING OR INVALID

102

One of the following for ARG 2:

NON-NUMERIC CHARACTER

CONTIGUOUS RANGES

NUMBER TOO BIG

LIST IS INCOMPLETE

RANGE IS INVALID

ZERO VALUE

103

One of the following for ARG 3:

NON-NUMERIC CHARACTER

CONTIGUOUS RANGES

NUMBER TOO BIG

LIST IS INCOMPLETE

RANGE IS INVALID

ZERO VALUE

104

CONSOLE ID INVALID

Example

```
wtor('an operator question') == 'ANSWER'   /* WTOR returns the reply*/
```

YANK()

Use YANK() to remove any record from any stack belonging to any IMOD.

Syntax

The YANK() function has this syntax:

```
record = YANK(recnum,[stack],[owner],[cntl])
```

Arguments

The YANK() function takes these arguments:

record

Returned record or error text.

recnum

Record number to be removed from the stack.

stack

Number of the stack to be referenced.

Default: If a local stack is referenced, the current stack (as set by the SWAPSTAK() function); otherwise, the value is set by the owning IMOD task using the PUBSTACK(DEFAULT) function.

owner

IMOD ID of the stack's owner (for external reference). This value can be obtained from the variable *imod_id* in the owning IMOD task. It is also the value returned by the SPAWN() function. The default is the current IMOD task.

cntl

Name of a variable to receive the control information stored with the record. The default is not to retrieve control information.

Return Codes

The YANK() function produces these return codes:

101 - 105

ARG *n* INVALID

122

STACK NOT SPECIFIED AND NO DEFAULT

123

RECORD NOT FOUND

125

SPECIFIED IMOD NOT FOUND

126

STACK WRITE FAILED

127

REQUESTED STACK DOES NOT EXIST

128

SPECIFIED IMOD/STACK NOT AUTHORIZED

Example

```
rec = yank(2)              /* Record number 2 is removed from the
                              current stack  */
rec = yank(2,5)            /* Record number 2 is removed from
                              local stack 5  */
rec = yank(2,,453,'data')  /* Record 2 is removed from IMOD 453's
                              information is stored in the
                              variable data. */
```

Usage Notes

YANK() can specify a stack external to an IMOD only when the target stack has been declared public for both reading and writing.

Chapter 10: CA-GSS/ISERV Commands

This section contains the following topics:

CA-GSS/ISERVE Operator Commands

How Commands Are Issued

You can issue a command in any of these ways:

- By entering the command verb (such as ISERVE) from the operator console. (An ISERVE initialization parameter defines this command verb to CA-GSS.)
- By using the MVS MODIFY (F) command from the operator console.
- By using the ISPF-based CA-GSS/ISERVE Control Panel.
- By using the CA SYSVIEW Performance Management ISERVE interface panel.

Each method is explained in the following sections.

Effect on Initialization Parameters

If you issue a command that changes the value of an initialization parameter, the change is temporary. Modifications, updates to tables, and so on, are valid until the ISERVE address space is terminated.

When you restart CA-GSS, all values revert to those specified in the initialization parameters.

Issue Commands From an Operator Console

If you are issuing a command from an operator console, use one of these formats:

verb command [parameters]

```
F identifier,command [parameters]
```

You may specify these parameters:

verb

A single word (like ISERVE) telling the operating system that ISERVE should process this command. Define *verb* with the COMMAND initialization parameter. When running both primary and secondary ISERVEs, assign a different command verb to each to prevent multiple ISERVEs from processing the same command.

identifier

Name of the started task (or JOBNAME), or the optional identifier specified in the MVS START command that started the CA-GSS address space.

command

Command that you are issuing.

parameters

Parameters that you are passing to the command.

Issue Commands From the CA-GSS/ISERVE Control Panel

If you are issuing commands from the ISPF-based CA-GSS/ISERVE Control Panel, use this format:

command [*parameters*]

A sample ISPF Control Panel follows:

```
------------------- CA-GSS/ISERVE Control Panel --------------------------
==>                                                            => PAGE
Target node: *             ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
```

These fields appear on the ISPF Control Panel:

==>

Command entry area.

=>

SCROLL field. Use this field to indicate how the screen should be scrolled by UP and DOWN commands. Valid scroll amounts are PAGE, HALF, and CSR. See your ISPF manuals for additional information.

Target node

GoalNet node where the command is to be processed. If it contains a blank or an asterisk (*), the command is not shipped over GoalNet.

(Target) ssid

Subsystem ID where the ISERVE that will execute the command is running.

If you omit the target node, CA-GSS uses the subsystem ID for the ISPF session from which the command was issued.

Host node

GoalNet node of the ISERVE that receives the initial command request. This field is for informational purposes and cannot be modified.

(Host) ssid

Identifies the subsystem ID of the ISERVE that will process the initial command request. You can specify any ISERVE that is executing on the same CPU as your ISPF session. If a target node or target subsystem ID is specified, it is the responsibility of the host ISERVE to transmit the request, receive the results, and then return them to the Control Panel.

The rest of the panel displays the results of the executed command. If the returned data exceeds the size of your screen, use standard ISPF scrolling commands (UP, DOWN, TOP) to position the display.

Command Routing Indicators

If you route your command beyond the host ISERVE, the first one or two lines of the display area include the routing information.

Example 1:

In this example, a command is sent to the ISRV address space:

```
----------------------- CA-GSS/ISERVE Control Panel --------------------------
==> command                                                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
Command response
Command response
and so on
```

Example 2:

In this example, a command is sent from the ISRV address space on the B1MAIN node to the address space at the B2MAIN node. The subsystem IDs for the ISERVEs happen to be identical:

```
----------------------- CA-GSS/ISERVE Control Panel --------------------------
==> command                                                          => PAGE
Target node: B2MAIN          ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
...Routing using GoalNet to B2MAIN.ISRV
Command response
Command response
and so on
```

Increase the Size of the Command Input Area

By default, the Control Panel input area occupies only part of a line. This provides the maximum display area and is still large enough for most ISERVE commands.

You can use the TOGGLE command to add a full line to the input area. Issue the TOGGLE command again to restore the input area to its original size.

If you need to expand the input area after you have filled it with a command, end the partial command with a hyphen (-) and press ENTER. The input area enlarges to two lines, your partial command is retained, and the cursor is positioned to the position occupied by the hyphen. The TOGGLE command can later be used to return the input area to its smaller size.

Issue Commands From the CA SYSVIEW Performance Management ISERVE Interface Panel

To issue commands from the CA SYSVIEW Performance Management ISERVE interface panel, you use the same syntax that you would use if you were issuing them from an operator console. See the section Issuing Commands From an Operator Console for details. However, you must prefix the commands with "ISERVE" to distinguish them from native CA SYSVIEW Performance Management commands. For example:

```
ISERVE DISPLAY ACTIVE
```

For additional information, see the CA SYSVIEW Performance Management manual set.

ACTIVATE

Use the ACTIVATE command to activate an IMOD that is been deactivated.

Syntax

Use this format for the ACTIVATE command:

```
ACTivate imodname
```

Parameters

The ACTIVATE command takes this parameter:

imodname

Name of the IMOD to be activated.

Example

```
------------------------- CA-GSS/ISERVE Control Panel ---------------------
==> activate job_sched                                              => PAGE
Target node: *               ssid: *    Host node: B1MAIN         ssid ISRV
================================================================================
IMOD JOB_SCHED now activated
```

Other Responses

IMOD name not loaded

The specified IMOD cannot be activated because it has not been loaded into the ISERVE address space.

Usage Notes

Also see DEACTIVATE.

ADD

Use the ADD command to add an ISET to the ISET table or to add a command to the command table.

Syntax

Use this format for the ADD command:

```
ADD {ISET isetname dsname}
    {CMD command imod [args]}
```

Parameters

The ADD command takes these parameters.

ISET isetname

Adds the specified ISET. The ISET name cannot be longer than 16 characters.

CMD command

Adds the specified command.

Specify required characters in uppercase and optional characters in lowercase.

dsname

Name of the data set to which the ISET name should refer.

imod

Name of the IMOD that will execute the command.

args

Argument string to be passed to the command IMOD. This string is concatenated before the operator-entered arguments.

Example

```
------------------------- CA-GSS/ISERVE Control Panel --------------
==> add iset newiset isrv.ABC.xyz                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN  ssid ISRV
================================================================================
NEWISET now identifies ISRV.ABC.XYZ
```

Usage Notes

- IMODs are not automatically loaded when you add their ISET to an ISET table. To load them, you need to issue a LOAD IMOD command.
- Panels for the IMOD editor are not modified when you add ISETs to tables.

  To add an ISET to an IMOD editor panel, add an ISET statement to the PARMLIB data set member where the editor's parameters are defined. (This member is identified using the EDITOR *initialization* parameter.)

  If you issue a LINK command, the added ISET will appear.
- You can use the ADD CMD command to replace an existing user-defined command.
- An ADD CMD command has no effect on native ISERVE commands.

Other Responses

ISET name missing

You omitted the ISET name.

ISET name is invalid

The ISET name is syntactically invalid. Make sure it is not too long.

ISET already exists

The specified ISET has already been defined to ISERVE, either by command or initialization parameter. You must delete the ISET before you can redefine it.

DSNAME is invalid

The data set name is syntactically invalid.

ADD option not ISET or CMD

You did not enter either ADD ISET or ADD CMD.

Command added

The specified command has been added to the command table.

Command modified

The specified command was already defined in the command table, and its definition has been replaced.

CANCEL

Use the CANCEL command to cancel an IMOD at its next REXX clause (normal cancel) or at its next dispatch (forced cancel).

Syntax

Use this format for the CANCEL command:

```
CANCEL taskid [FORCE]
```

Parameters

The CANCEL command takes these parameters:

Taskid

Unique identifying number of the IMOD that you are canceling. The value is entered in hexadecimal.

To determine this number, use the DISPLAY ACTIVE command. The number is displayed in the IMOD ID column.

FORCE

Performs a forced cancel, which cancels the IMOD at the next dispatch. Otherwise, the IMOD is terminated at the conclusion of the current REXX clause.

You cannot perform a forced cancel unless you have tried to perform a normal cancel and the cancel failed.

Example

If you issue a CANCEL 3BC7 FORCE command (after issuing a CANCEL 3BC7 command), the following information is returned:

```
------------------------- CA-GSS/ISERVE Control Panel ------------------
==> cancel 3BC7 force                                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN      ssid ISRV
================================================================================
FORCE scheduled for IMOD 3BC7
```

Usage Notes

- The CANCEL command cannot terminate an IMOD while it is waiting for some external event, such as a WTOR. However, the IMOD is canceled as soon as the wait is completed.
- The CANCEL command without the FORCE option does not terminate an IMOD that is in a non-REXX loop, such as a loop occurring within a function or ADDRESS environment.

Other Responses

CANCEL scheduled for IMOD taskname

The specified IMOD will be canceled at the end of the next REXX clause.

IMOD taskname not found

The specified IMOD is not currently executing.

IMOD taskname required "CANCEL" before "FORCE"

You cannot perform a forced cancel unless a previous normal cancel failed.

DEACTIVATE

Use the DEACTIVATE command to temporarily disable an IMOD. The IMOD may be reactivated by use of the ACTIVATE command.

Syntax

Use this format for the DEACTIVATE command:

```
DEACTivate imodname
```

Parameters

The DEACTIVATE command takes this parameter:

imodname

Name of the IMOD to be disabled.

Example

```
------------------------- CA-GSS/ISERVE Control Panel -----------------
==> deactivate badprog                                          => PAGE
Target node: *          ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
IMOD BADPROG deactivated
```

Other Responses

IMOD imodname not loaded

The specified IMOD cannot be found in the ISERVE address space that is processing the command.

Usage Notes

Also see "ACTIVATE". (see page 464)

DELETE

Use the DELETE command to delete an ISET name from the list of available ISETs.

Syntax

Use this format for the DELETE command:

```
DELete ISET isetname
```

Parameters

The DELETE command takes this parameter:

isetname

Name of the ISET to be deleted from the list of available ISETs.

Example

```
------------------------- CA-GSS/ISERVE Control Panel ----------------------
==> delete iset newiset                                              => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
NEWISET has been deleted
```

Usage Notes

- IMODs that have already been loaded from the deleted ISET are not affected by this command - they are not disabled or removed from storage. To prevent these IMODs from being used, you must deactivate them through the DEACTIVATE command.
- Panels for the IMOD editor are not modified when you delete ISETs.

To remove an ISET from an IMOD editor panel, delete the appropriate ISET statement from the PARMLIB data set member where the parameters for the editor are defined. (This member is identified using the EDITOR initialization parameter.)

Other Responses

ISET NOT FOUND

The specified ISET has not been defined to the ISERVE address space that is processing the command.

ISET name is invalid: xxxxx

The specified name (xxxxx) is syntactically invalid.

DISPLAY ACEE

Use the DISPLAY ACEE command to display information about the ACEEs that CA-GSS has obtained for user IDs that have tried to execute IMODs.

Syntax

Use this format for the DISPLAY ACEE command:

```
Display ACEE [ID|AGE]
```

Parameters

The DISPLAY command takes these parameters:

ID

Sorts the display by user ID.

AGE

Sorts the display by age, where age is the amount of time since the user ID was last used.

Example

Here is an example.

```
---------------------- CA-GSS/ISERVE Control Panel ------    ROW 1 TO 19 OF 272
==> display acee                                                        => PAGE
Target node: *                ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
    Userid     ACEE             Users    Age              CPU
    --------   --------         -----    ---------        -----------
    $ALARED    006D60C0         0        2:38:30.640      0.145
    $ISADJJ    006CEF58         0        6:08:27.679      0.021
    AALON      006CF478         0        5:05:26.937      0.105
    AAREMBA    006D8F58         0        3:47:28.402      0.019
    AA1AF47    006D0F58         0        22:47:16.907     0.021
    ACENTOF    006BD888         0        5:17:39.834      0.016
    ACUNNIN    006D0A80         0        22:48:32.076     0.016
    AFORBES    006BD578         0        1:59:14.386      0.042
    AHB        006CBC88         0        26:01.950        0.077
    AKINSIN    006C9728         0        42:39.678        0.529
    ALEWIS     006D9D98         0        27:37:03.892     0.097
```

Fields in the Example

Userid

User ID, as recognized by the system security software.

If the ID is preceded by an asterisk, it is the CA-GSS address space user ID. If it is preceded by a minus sign, it is the default user ID.

ACEE

Memory address (in hexadecimal) of the ACEE control block.

Users

Number of IMODs currently executing under this user ID/ACEE.

Age

Elapsed time since an IMOD task last executed under this user ID/ACEE. Format: hh:mm:ss.tho.

CPU

Cumulative CPU time consumed by all IMOD tasks executed under this user ID/ACEE. Format: hh:mm:ss.tho.

DISPLAY ACTIVE

Use the DISPLAY ACTIVE command to identify all active IMODs, their current execution status, and their IDs.

Syntax

Use this format for the DISPLAY ACTIVE command:

```
Display ACTive
```

Example

```
------------------------- CA-GSS/ISERVE Control Panel -------------------------
==> display active                                                          =>
PAGE
Target node: *          ssid: *                  Host node: B1MAIN      ssid ISRV
================================================================================
ID              IMOD       L EXEC               Inst Status   Attributes
--------        ---------- ---------------      ---- ------   ----------------
0000001D        $SRVO      0 $SRVO                65
                           1 $SRVC                40
                           2 $SRVC_DISPLAY        19
                           3 $SRVC_DISP_ACTIV     5
                           4 $SRV_IMOD_ACT        96   Active
00000001        $TIMER     0 $TIMER               10            ADDRESS  Server
00000005        $GNET_LOG  0 $GNET_LOG            65            STACK    Server
```

Fields in the Example

ID

Unique hexadecimal IMOD ID for this IMOD; can be used as an operand of CANCEL.

IMOD

Name of the IMOD that was initially invoked.

L

Level of external subroutine nesting. The initial IMOD has a level of 0.

EXEC

Name of an IMOD being executed as an external subroutine.

Inst

Number of the REXX instruction currently being executed in the IMOD.

Status

Execution status of the IMOD's current subroutine. The following table lists the values:

Value	Description
Active	This IMOD has control of the CPU (displayed only for the IMOD executing the DISPLAY function).
ADDRESS	Waiting for ADDRESS command to complete.
CLOSE	Waiting for CLOSE to complete.
Dormant	IMOD has completed and is waiting for its results to be claimed.
Enqueue	Waiting on another IMOD for an internal enqueue.
Function	Waiting for an asynchronous function call to complete.
GoalNet	Waiting on an external subroutine call across GoalNet.
I/O	Waiting for an I/O function to complete.
OPEN	Waiting for OPEN to complete.
OPSVALUE	Waiting for an OPSVALUE() function to complete.
OSEnq	Waiting for control of a z/OS enqueue.
Slice	Waiting due to number of REXX clauses executed; control released to other IMODs.
STACK	Waiting on a record to be posted to a stack.
SVC	Waiting for an SVC to complete.

Value	Description
Terminal	Waiting for I/O from a terminal.
Timer	Waiting due to a PAUSE() function.
WTOR	Waiting for operator response.

Attributes

Identifies IMODs that are associated with a remote GoalNet node. Fields may be preceded by one of these arrows:

<- IMOD is returning results to the associated node.

-> A request was made of the associated node.

Fields without arrows indicate a general association that depends on the IMOD being executed. The attribute SERVER indicates that the IMOD is running as a server.

DISPLAY ADDRESS

Use DISPLAY ADDRESS to display a summary of all true subtasks, which are used by address environments and certain other asynchronous processes.

Syntax

Use this format for the DISPLAY ADDRESS command:

```
Display ADDRESS
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel ----------------------
==> display address                                                 => PAGE
Target node: *        ssid: *          Host node: B1MAIN         ssid ISRV
================================================================================
                                                          Max  CPU     Accum
Name      Type  TCB       IMOD ID   Status              CPU  Left    Task CPU
--------  ----  --------  --------  ----------------  ---  ------  ----------
IDCAMS    ADDR  007F1550  000000CA  Inact,Reus          15              1.85
EMVS      ADDR  007F17E8  000000C9  Inact,Reus          15              2.25
FILE      ADDR  007F60D0  00000002  Inact,Reus          0               0.11
SYSVIEW   ADDR  007F1260  00000101  Act                3         2.85
```

Fields in the Example

Name

Name of the address environment. Environments that are indented are internal environments, invoked to perform certain synchronous tasks in an asynchronous fashion.

Type

Indicates that the task supports an address or function.

TCB

Memory address assigned to the task control block owned by the task that processes a particular ADDRESS command.

IMOD ID

IMOD that last used this subtask for processing a command.

Status

Status of the task as follows:

Abend

Task has abended.

Act

Task is currently processing a command.

Inact

Task is available for use.

Reus

Reusable subtask.

Max CPU

Maximum CPU time permitted for the completion of a single ADDRESS command.

CPU Left

CPU time left for the subtask to complete its current operation. This field is displayed only if the subtask is currently processing work.

Accum Task CPU

Total amount of CPU time used by the subtask since its creation.

DISPLAY ADDRTAB

Use the DISPLAY ADDRTAB command to display information about the relationship between an address environment and its associated load module.

Syntax

Use this format for the DISPLAY ADDRTAB command:

```
Display ADDRTAB
```

Example

```
------------------------- CA-GSS/ISERVE Control Panel --------------------
==> display addrtab
=> PAGE
Target node: *         ssid: *         Host node: B1MAIN          ssid ISRV
================================================================================
Name     Module   Address  Length   Rent Avail AMODE  Detach  Mtask Limit Type
SVC      TSKSVC   06611898          Yes  Yes   31                      15    253
FILE     TSKFILE  06610DA0          Yes  Yes   31                      15    252
SVC99    TSKSVC   99 06611A20       Yes  Yes   31                      15    254
XPSPOOL  ESFGSSAD                              24                       5    0
DSNHLI2  DSNHLI2                               24                       5    255
DSNALI   DSNALI                                24                       5    255
INSIGHT  IDB2COMM 00001E10          Yes        24              1        5    6
IDCAMS   IDCAMS   8001E198 00019090 Yes  Yes   31                       5    1
RERUN    RUNTADDR 866FFE80 0002A180 Yes  Yes   31                       5    3
JOBTRAC  GJTRGCUU 00016428          Yes        31              5        5    3
OPSVALUE OPGLEVMG 86741000 000017E8 Yes  Yes   31                       5    255
OSF      OPGLEVMG 86741000 000017E8 Yes  Yes   31                       5    13
AOF      OPGLEVMG 86741000 000017E8 Yes  Yes   31                       5    12
OPSREQ   OPGLEVMG 86741000 000017E8 Yes  Yes   31                       5    11
OPER     OPGLEVMG 86741000 000017E8 Yes  Yes   31                       5    10
INFO     BLGYRXM                       Yes        24                       5    0
```

Fields in the Example

Name

Name of the address environment. The names IRXEXEC, SVC, SVC99, and FILE represent internal environments that are not defined through the ADDRESS parameter.

Module

Name assigned to the load module that processes the related ADDRESS commands. For internal environments, this name does not reflect the name of a load module.

Address

Memory address where the module is located (or blank for modules that are not re-entrant).

Length

Length of the load module (or blank for routines that are part of the ISERVE (SRVSYS) load module).

Rent

Indicates whether or not the module is re-entrant (Yes or blank).

Avail

Indicates whether or not the module is available (Yes or blank). Some load modules specified in the initialization parameters may not be accessible to ISERVE.

AMODE

Addressing mode of the load module (24- or 31-bit).

Detach

Indicates whether or not the subtask is deleted when the IMOD task ends. Otherwise, the subtask is re-assigned to the next IMOD task requesting the same ADDRESS environment.

Mtask

Maximum number of IMOD tasks that can use this environment concurrently (or blank for no limit).

Limit

Maximum seconds of CPU time that the address environment can consume to process a single request (or blank for no limit). If this value is exceeded, the subtask is terminated and an error indication is returned to the IMOD.

Type

Interface type for the address environment. This value identifies to CA-GSS the type of support required by the environment.

Usage Notes

You can use the ADDRESS initialization parameter to define address environments and to associate them with load modules.

DISPLAY CMD

Use DISPLAY CMD to display information on CA-GSS Operator Panel commands that were created by the CMD initialization parameter and the ADD command.

Syntax

Use this format for the DISPLAY CMD command:

```
Display CMd
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel -------   ROW 1 TO 3 of 3
==> display cmd                                                      => PAGE
Target node: *              ssid: *    Host node: S18EWF               ssid EWF
================================================================================
Command                IMOD                     Args
-------------------    --------------------      ---------------------
NEWcmd                 IMOD_CMD_PROCESS
```

Fields in the Example

Command

Command verb. Required characters are shown in uppercase, and optional characters are shown in lowercase.

IMOD

Name of the IMOD that the command invokes.

Args

Optional arguments that are passed to the IMOD.

DISPLAY COMMANDS

Use the DISPLAY COMMANDS command to display information about z/OS console operator commands created using the COMMAND initialization parameter.

Syntax

Use this format for the DISPLAY COMMANDS command:

```
Display COmmands
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel ------   ROW 1 TO 4 of 4
==> display commands                                                 =>PAGE
Target node: *              ssid: *   Host node: S18EWF               ssid EWF
================================================================================
Command                 IMOD                    User ID
-------------------     --------------------    ---------------------
EWF                     $SRVC
EWFX                    Prefix EWFX_CMDUSER
```

Fields in the Example

Command

Command name.

IMOD

Name of the IMOD that should be invoked when the command is issued.

The word Prefix indicates that the IMOD name is derived by appending the first blank-delimited word after the command name to the value of the IMOD field.

User ID

User ID under which the IMOD executes. A blank indicates the IMOD executes under the default CA-GSS user ID (if one is defined) or the CA-GSS address space's user ID.

DISPLAY CPU

Use the DISPLAY CPU command to list CPU usage by user ID and ISET.

Syntax

Use this format for the DISPLAY CPU command:

```
Display Cpu {ISET|USERid}
```

Parameters

The DISPLAY CPU command takes these parameters:

ISET

Lists CPU usage by ISET name.

USERid

Lists CPU usage by user ID.

Example

```
---------------------- CA-GSS/ISERVE Control Panel ---   ROW 1 TO 19 OF 20
==> display cpu iset                                                => PAGE
Target node: *               ssid: *    Host node: B1MAIN            ssid ISRV
================================================================================
ISET             CPU
---------------  ------------
*SYSTEM*         0.000
INTERNAL         3.511
EWF              0.000
PROD             0.041
EMROPROD         15:49.671
INSIGHT          33.297
SAR/EX           0.004
MSCHARE          0.000
GSSTECH          0.000
BALA             0.000
```

Fields in the Example

Userid

User ID associated with the IMOD that accumulated the CPU usage.

ISET

ISET that the lowest level routine (IMOD) was loaded from.

CPU

Cumulative CPU time, in hours, minutes, seconds, and thousandths.

Usage Notes

CPU usage is charged to an ISET if the lowest level routine (IMOD) was loaded from the ISET. Subroutine IMODs are charged against the ISET of the lowest level routine.

DISPLAY CSECT

Use DISPLAY CSECT to display the status of any CSECT in any load module in any library.

Syntax

Use this format for the DISPLAY CSECT command.

```
Display CSECT {STEPLIB module} [csect]
              {JOBLIB module}  [IMOD imodname]
              {dsname module}
              {dsname(module)}
```

Parameters

The DISPLAY CSECT command takes these parameters:

STEPLIB

Searches the STEPLIB data set.

JOBLIB

Searches the JOBLIB data set.

Dsname

Searches the specified data set. This data set must be cataloged and must be accessible (read access) to the user issuing the request.

Module

Name of the load module to be searched.

Csect

Name of the CSECT to be displayed. If omitted, the CSECT name is assumed to be identical to the load module name.

Imodname

Search for the CSECT that contains the specified IMOD. This parameter is valid for load modules created with the IMOD Packaging Facility.

Example

```
------------------------ CA-GSS/ISERVE Control Panel ------ ROW 1 TO 4 OF 4
==> display csect ewf.fxao.authlib(test)                          => PAGE
Target node: *              ssid: *    Host node: B1EWF             ssid EWF
================================================================================
Display for load module TEST in EWF.FXAO.AUTHLIB
CSECT TEST was assembled on 03/18/06 by version 02.01
Fix TEST021 was applied on 03/18/06
Fix NO_IDENT was applied on 03/18/06
```

DISPLAY EDITOR

Use the DISPLAY EDITOR command to display IMOD editor values for the local ISERVE address space and for the primary ISERVE.

Syntax

Use this format for the DISPLAY EDITOR command:

```
Display EDitor
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel ---------  ROW 1 TO 16 of 16
==> display editor                                                      => PAGE
Target node:  *               ssid: *    Host node: S18EWF              ssid EWF
================================================================================
IMOD Editor Values
------------------------------
Global:   VIO Unit:           VIO
          Parmlib:            EWF.FXA0.PARMLIB
          Parmlib Member:     NEWEDIT
          Panel Library:      EWF.FXA0.ISRPLIB
          Message Library:    EWF.FXA0.ISRMLIB
          Load Library:       EWF.FXA0.AUTHLIB
          CLIST Library:      EWF.FXA0.CLIST
Local:    VIO Unit:           VIO
          Parmlib:            EWF.FXA0.PARMLIB
          Parmlib Member:     NEWEDIT
          Panel Library:      EWF.FXA0.ISRPLIB
          Message Library:    EWF.FXA0.ISRMLIB
          Load Library:       EWF.FXA0.AUTHLIB
          CLIST Library:      EWF.FXA0.CLIST
```

Fields in the Example

Global

Values that are in effect for anyone who is using the IMOD editor.

Local

Values that the local address space supplies when someone who is using the IMOD editor issues a LINK command to the address space.

Usage Notes

IMOD editor values are set using the EDITOR initialization parameter and the SET EDITOR command.

DISPLAY ENQUEUES

Use the DISPLAY ENQUEUES command to list all currently held enqueues that were obtained using the ENQUEUE() function.

Syntax

Use this format for the DISPLAY ENQUEUES command:

```
Display ENQueues
```

Example

```
----------------------- CA-GSS/ISERVE Control Panel -------  ROW 1 TO 2 OF 2
==> display enqueues                                                 => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
IMODID     STAT    LEVEL   COUNT      QNAME      RNAME
E 00001022 OWN     EXCL               GVAR       I/O.COUNTER.PHASE.1
E 00001295 SHR                2       GVAR       I/O.COUNTER.PHASE.1
```

Fields in the Example

prefix

One of these values:

Enqueue.

Lock. (Locks are not available for use by IMODs.)

IMODID

Hexadecimal ID of the IMOD that owns or is waiting for the resource.

STAT

Current ownership status, as follows:

OWN

IMOD has the requested level of ownership.

blank

IMOD is waiting for the requested ownership.

LEVEL

Level of ownership that was requested:

CONT

Control ownership. Other IMODs can have shared ownership of the resource, but they cannot have exclusive or control ownership of it.

EXCL

Exclusive ownership. Other IMODs cannot have any type of ownership of the resource.

SHR

Shared ownership. Other IMODs can have shared ownership of the resource, but they cannot have exclusive or control ownership of it

COUNT

For SHR-type enqueues, the maximum permitted simultaneous SHR users.

QNAME

1- to 8-character name that serves as a high-level qualifier of the resource.

RNAME

1- to 128-character name that, when appended to qname, uniquely identifies a resource. If rname length exceeds the display width, it is wrapped onto the following lines.

Usage Notes

IMOD tasks can use the ENQUEUE() function to serialize their use of resources. The enqueues obtained through this function are similar, but not identical to, z/OS system-level enqueues.

DISPLAY FNTAB

Use the DISPLAY FNTAB command to display information about REXX functions that are loaded when CA-GSS is initialized.

Syntax

Use this format for the DISPLAY FNTAB command:

```
Display FNTab [{External} [[+|-] {Class|Name} [[+|-] {Class|Name}]]
              {Internal}
              {Builtin}
              {All}
```

Parameters

The DISPLAY FNTAB command takes these parameters.

External

Lists only external functions (ones defined in the initialization parameters). This is the default.

Internal

Lists only internal (CA-GSS extended) functions, which set the RC special variable.

Builtin

Lists only built-in functions, which are included as part of the language definition.

All

Lists all functions.

+

Sorts in ascending order. This is the default.

-

Sorts in descending order.

Class

Sorts by function class (external, internal, or built-in).

Name

Sorts by function name.

Usage Notes

- You can specify up to two sort criteria. If you specify both CLASS and NAME, information is sorted by name within class.
- If no sort criteria are specified, +NAME is used.

Example

```
---------------------- CA-GSS/ISERVE Control Panel ------  ROW 1 TO 19 OF 170
==> display fntab all -class +name                                  => PAGE
Target node: *               ssid: *    Host node: B1EWF             ssid EWF
================================================================================
Class    Name       Avail  Module   Type   Address   Length    Rent AMODE
-------- ---------- -----  -------- -----  --------  --------  ---- -----
Internal  $CALLX     Yes   ***INT   SYNC
Internal  $TIME      Yes   ***INT   SYNC
Internal  ALIGN      Yes   ***INT   SYNC
Internal  ALLOC      Yes   ***INT   SYNC
Internal  BATCHR     Yes   ***INT   SYNC
Internal  BOOLWORD   Yes   ***INT   SYNC
Internal  CALC       Yes   ***INT   SYNC
Internal  CASE       Yes   ***INT   SYNC
External  ALERT            SRVALERT SYNC   85DB9330  00001558  Yes    31
External  SRVCALL    Yes   SRVCALL  ASYNC  85DB1C38  000013C8  Yes    31
```

Fields in the Example

Class

Function's class, as follows:

Builtin

Function that is considered part of the language specification.

External

Processed by external load module specified in initialization parameters.

Internal

CA-GSS extended function.

Name

Name of the function, as coded in IMODs.

Avail

Indicates whether or not the function is currently available for use by IMODs.

Module

Name of the load module that will be used to process the function.

Type

How the function executes, as follows:

ASYNC

Executes as a subtask.

SYNC

Executes under the main ISERVE task.

Address

Address in memory (hexadecimal) where the load module resides, if applicable.

Length

Length of the load module (hexadecimal) in memory, if applicable.

Rent

Indicates whether or not the load module is re-entrant, if applicable.

AMODE

Addressing mode of the load module (24- or 31-bit), if applicable.

DISPLAY GLOBAL

Use the DISPLAY GLOBAL command to display information on the current status of REXX *global variables* (shared variables that begin with an ampersand [&]).

Syntax

Use this format for the DISPLAY GLOBAL command:

```
Display GLOBAL [&var]
```

Parameters

The DISPLAY GLOBAL command takes this parameter:

&var

Limits the display to information about a particular global variable.

Example

```
----------------------- CA-GSS/ISERVE Control Panel -----------------------
==> display global                                                 => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
Global Variable Storage: 153856 bytes
```

Usage Notes

- If you do not limit the display to a particular global variable, the total virtual storage currently allocated to all global variables is displayed.
- Global variable names are case-sensitive.

DISPLAY GSS

Use the DISPLAY GSS command to display information about all CA-GSS address spaces running on the current z/OS image.

Syntax

Use this format for the DISPLAY GSS command:

```
Display GSs
```

Example

```
-------------------- CA-GSS/ISERVE Control Panel -----------    ROW 1 TO 7 OF 7
==> display gss                                                        => PAGE
Target node: *               ssid: *      Host node: B1MAIN          ssid ISRV
================================================================================
SSID   Jobname      Job ID    Version    GoalNet Node  LU name   GMF name

++++   GSSX         STC00090
ID42   GSWV42SS     Unknown
ITIM   GSWTIMSS     Unknown
GGMF   GSSCO        STC00089  02.07.00   S18GGMF       GSS05N20  S18GGMF
*ISRV  GSS          STC01222  02.07.00   S18MAIN       GSS05N1   S18MAIN
```

Fields in the Example

SSID

Subsystem ID for the CA-GSS address space. One of these special values may be shown.

preceding

Primary CA-GSS.

++++

LX-owning address space that is not executing an ISERVE.

Jobname

Address space's job name.

Job ID

Address space's job ID.

Version

Version of CA-GSS that is running. A value of Unknown indicates a pre-2.7 release of CA-GSS.

GoalNet Node

GoalNet node name (if one is assigned).

LU name

VTAM LU name being used by GoalNet (if active).

Usage Notes

The display is updated whenever a CA-GSS address space initializes. If an address space terminates, nothing is deleted.

DISPLAY ILOG

Use the DISPLAY ILOG command to display information about one or all ILOG files that are currently accessible to ISERVE.

Syntax

Use this format for the DISPLAY ILOG command:

```
Display ILog [ilognumber]
```

Parameters

The DISPLAY ILOG command takes this parameter:

Ilognumber

Limits the display to information about one particular ILOG file.

Example

```
------------------------- CA-GSS/ISERVE Control Panel ----------------------
==> display ilog                                                      => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
Log File    Status         Record  Used    First Record       Last Record
0           0 Full-Avail       12  25.4%   06/01/25 13.27.16  06/01/28 10.17.53
            1 Full-Avail        4   0.8%   06/01/28 10.34.57  06/01/28 10.44.55
            2 Active            0   0.0%
1           0 Active            0   0.0%
```

Fields in the Example

Log

ILOG file number.

File

Subfile within the ILOG.

Status

ILOG subfile's current recording status, as follows:

Active Currently being recorded on (one per ILOG).

Dump-Req Full; will not be reused until a RESET command is issued for the subfile.

Empty Empty and available.

Full-Avail Full; however, as needed, it will be reset to empty.

Not-Avail Not available. Check the ISRVLOG and your SYSLOG for error messages related to this subfile.

Record

Number of records in the ILOG.

Used

Percent full of data.

First Record

Date and time stamp of the earliest (first) record in the subfile.

Last Record

Date and time stamp of the latest (last) record in the subfile.

Usage Notes

ILOGs are automatically reset if you do not provide a $USER_ILOG_FULL IMOD.

DISPLAY IMOD

Use the DISPLAY IMOD command to determine information about a particular IMOD, including where it was loaded from and at whose request.

Syntax

Use this format for the DISPLAY IMOD command:

```
Display IMod imodname
```

Parameters

The DISPLAY IMOD command takes this parameter:

imodname

Name of the IMOD that you want information about.

Example

```
------------------------ CA-GSS/ISERVE Control Panel -----------------------
==> display imod isrv_add                                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
IMOD             LOC       LEN  USE  CHG-DATE  CMP-VER   INT  CALL  ACT
---------------- --------  ---- ---  --------  --------  ---  ----  ---
ISRV_ADD         04D52718  04C8 0    06/01/17  02.02.00  NO   YES   YES
IMOD was loaded dynamically from INTERNAL, dsn EWF.IMOD.BASE
by TSO user EWF
```

Fields in the Example

IMOD

IMOD's name.

LOC

Memory address of the entry that defines the IMOD (useful to CA Technologies Support only).

LEN

Amount of storage occupied by the IMOD.

USE

Number of IMODs currently using this IMOD. The use-count is used to determine when an IMOD can be replaced (using a LOAD request). The previous copy of an IMOD is not deleted from memory until its use-count reaches zero.

CHG-DATE

Date when the IMOD was last changed. (If you simply recompile an IMOD, this date will not change.)

CMP-VER

Version of the compiler that last compiled the IMOD.

INT

Indicates whether or not this is an internal IMOD (rather than an IMOD that is based in an ISET). These IMODs are physically part of ISERVE's load module and were installed at the factory.

CALL

Indicates whether or not the IMOD can be called as an external subroutine by other IMODs.

ACT

Indicates whether or not the IMOD is loaded and activated, making it available for execution. (If NO is shown, you can use the ACTIVATE command to re-activate the IMOD.)

Usage Notes

The rest of the panel describes how the IMOD was loaded, from what library, and the user or jobname that issued the LOAD command.

DISPLAY ISET

Use the DISPLAY ISET command to display all ISETs that are currently accessible to ISERVE. The ISETs are sorted alphabetically by name.

Syntax

Use this format for the DISPLAY ISET command:

```
Display ISet [DESC]
```

Parameters

The DISPLAY ISET command takes this parameter:

DESC

Includes descriptions (if any) of the ISETs in the display.

Example

```
----------------------- CA-GSS/ISERVE Control Panel --------------------------
==> display iset desc                                                 =>PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
ISET            VERSION                          DSNAME
*SYSTEM*        02.07.01                         Internally link-edited IMODs
GSSTECH         BASRE01.SVR.TECHIMOD             CA-GSS Technical Support
GSSTEST         CORPTECH.GSS.CSERV.SRV.TECHIMOD CA-GSS Technical Support Test
INSIGHT         GSW.IV42INTG.IMOD                IDB2 Development ISET
INTERNAL        CORPTECH.GSS.BASEIMOD           Release 2.7 (Development) IMOD's
JOBTRAC_COMMON  APC.LV1JT.COMMON.ISET            Jobtrac Support IMODs
```

Fields in the Example

ISET

ISET's name.

VERSION

Version number of the ISERVE editor last used to update the ISET. This value is obtained only when the ISET has been accessed in order to load one or more IMODs. Because not all ISETs are accessed at initialization, many of these fields will be blank.

DSNAME

Data set name associated with the ISET and description (if any) of each ISET.

Usage Notes

An ISET must appear in this display before IMODs can be loaded from it. ISETs are defined with the initialization parameters or the ADD ISET command.

DISPLAY MVS

Use the DISPLAY MVS command to display information about the system.

Syntax

Use this format for the DISPLAY MVS command:

```
Display MVS [CATalogs|LPA]
```

Parameters

The DISPLAY MVS command takes these parameters:

CATalogs

Displays information about open catalogs.

LPA

Displays information about link pack area libraries.

Example

The result of a DISPLAY MVS command:

```
---------------------- CA-GSS/ISERVE Control Panel ------------  ROW 1 TO 7 OF 7
==> display z/OS                                                      => PAGE
Target node: *                ssid: *      Host node: S18EWF           ssid EWF
================================================================================
z/OS System Values:
--------------------
IPL Date: 07/01/06    IPL Time: 22:39:40          Nucleus:IEANUC01
IPL Unit: 0801        IPL Volume:  SXXRS8
CPU Model: 9672       CPU ID:  60216              I/O Config:  04
System: SP4.3.0       FMID: HBB4430
Master Catalog:  CATALOG.MSTRCAT.S018A
```

The result of a DISPLAY MVS CATALOGS command:

```
---------------------- CA-GSS/ISERVE Control Panel ----------   ROW 1 TO 7 OF 7
==> display z/OS catalogs                                             => PAGE
Target node: *               ssid: *       Host node: S18EWF            ssid EWF
================================================================================
Open Catalogs:
--------------------
CATALOG.USERCAT.DEMO
CATALOG.MSTRCAT.S038A
CATALOG.MSTRCAT.S034A
CATALOG.MSTRCAT.S014A
CATALOG.USERCAT.LAX.MISC
CATALOG.MSTRCAT.S018A (Master)
```

The result of a DISPLAY MVS LPA command:

```
---------------------- CA-GSS/ISERVE Control Panel ------------ ROW 1 TO 7 OF 7
==> display z/OS lpa                                                 => PAGE
Target node: *             ssid: *       Host node: S18EWF           ssid EWF
================================================================================
z/OS LPA Libraries:
--------------------
SYS1.LPALIB
z/OS18.LPALIB
z/OS18.SAS.LPALIB
ISP.V3R5M0.ISPLPA
ISP.V3R5M0.ISRLPA
SYS1.ISAMLPA
REXX.V1R2M0.SEAGLMD
SYSMCAT.EPIC.V3R2.LPALIB
SYSMCAT.EPIC.V3R1.LPALIB
```

DISPLAY PRODUCT

Use the DISPLAY PRODUCT command to display information about products for which CA-GSS support has been activated.

Syntax

Use this format for the DISPLAY PRODUCT command:

```
Display Product
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel ---------- ROW 1 TO 2 OF 2
==> display product                                                  => PAGE
Target node: *             ssid: *       Host node: S18EWF           ssid EWF
================================================================================
CA-SYSVIEW:   Supported
CA-INSIGHT for DB2:  Supported
```

Usage Notes

- CA-GSS support for external products is activated using the PRODUCT initialization parameter.

DISPLAY STACK

Use the DISPLAY STACK command to display the amount of storage currently allocated to REXX stack structures. In general, this value is useful only if you suspect storage shortages in the ISERVE address space.

Syntax

Use this format for the DISPLAY STACK command:

```
Display STAck
```

Example

```
------------------------ CA-GSS/ISERVE Control Panel ----------------------
==> display stack                                                  => PAGE
Target node: *               ssid: *    Host node: B1MAIN        ssid ISRV
================================================================================
Stack allocated: 21.4 k bytes. Free: 18.8 k bytes
```

Fields in the Example

Stack Allocated

Total virtual storage in the ISERVE address space that is currently reserved for stack use (in kilobytes). This is a high-water mark and will increase during execution.

Free

Amount of the allocated stack area that is currently unused and available (in kilobytes).

DISPLAY STORAGE

Use the DISPLAY STORAGE command to display the amount of storage that is being used inside the ISERVE address space.

Syntax

Use this format for the DISPLAY STORAGE command:

```
Display STOrage [LONG]
```

Parameters

The DISPLAY STORAGE command takes this parameter:

LONG

Breaks down the display by use. This information is useful to CA Technologies Support.

Examples

```
------------------------ CA-GSS/ISERVE Control Panel --------------------
==> display storage                                                 => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
Below 16M   Above 16M

 220k         1464k
```

DISPLAY TRACE

Use the DISPLAY TRACE command to see what type of trace information is being sent to CA-GSS' ISRVLOG data set.

Syntax

Use this format for the DISPLAY TRACE command:

```
Display Trace
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel ------    ROW 1 TO 3 OF 3
==> display trace                                             => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
FAILed to run:   ON   SPACE usage:   ON   imod LOAD:   ON
imod SCHEDuled:  ON   member ALTER:  ON   RESULT:     OFF
DEBUG:           ON   SUBCALL:       OFF
```

Fields in the Example

FAILed to run

Indicates whether or not trace output is recorded for each IMOD that is requested but not available.

SPACE usage

Indicates whether or not an IMOD's memory use is recorded when it ends. CA-GSS always traces IMODs that do not release all memory.

imod LOAD

Indicates whether or not an IMOD's location and length are recorded when the IMOD is loaded or is replaced with a new copy.

imod SCHEDuled

Indicates whether or not trace output is recorded for each IMOD that is scheduled for execution.

memory ALTER

Indicates whether or not trace output is recorded when storage is altered due to use of the MEMORY() function.

RESULT

Indicates whether or not an IMOD's return value is recorded when an IMOD ends and returns a non-null string.

DEBUG

Indicates whether or not various internal debugging data (useful to CA Technologies Support) is recorded.

SUBCALL

Indicates whether or not each subroutine call to an external IMOD is traced.

Usage Notes

You can use the TRACE initialization parameter or the TRACE command to determine what trace output is sent to the ISRVLOG data set.

DISPLAY VERSION

Use the DISPLAY VERSION command to display the current versions of the ISERVE address space and all ISETs that have been accessed since ISERVE was last initialized.

Syntax

Use this format for the DISPLAY VERSION command:

```
Display Version
```

Example

```
------------------------- CA-GSS/ISERVE Control Panel --------------------
==> display version                                                 => PAGE
Target node: *               ssid: *    Host node: B1MAIN         ssid ISRV
================================================================================
ISERVE Address Space at level:  02.07.00
INTERNAL        file at level:  02.07.00
Ez/OS                file at level:  02.07.00
INSIGHT             file at level:  02.07.00
```

DISPLAY WTOS

Use the DISPLAY WTOS command to display the criteria that CA-GSS is using to intercept WTOs and execute IMODs.

Syntax

Use this format for the DISPLAY WTOS command:

```
Display WTOs
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel ---------- ROW 1 TO 3 OF 3
==> display wtos                                                  => PAGE
Target node: *              ssid: *      Host node: S18MAIN          ssid ISRV
================================================================================
WTO ID        IMOD                    ILOG            User ID

SRV808        JOBTRAC_COMMANDS
```

Fields in the Example

WTO ID

WTO ID or fragment that can be matched.

IMOD

IMOD that is invoked when the WTO is intercepted.

ILOG

ILOG number where the WTO should be recorded.

User ID

User ID assigned to the IMOD that is executed when the WTO is intercepted. A blank indicates that the IMOD executes under the user ID of the WTO's issuer.

Usage Notes

You can use the WTO initialization parameter to provide the criteria used to intercept WTOs and execute IMODs.

DISPLAY ZAPS

Use the DISPLAY ZAPS command to display fixes (zaps) that are currently applied to a particular load module.

Syntax

Use this format for the DISPLAY ZAPS command.

```
Display Zaps {STEPLIB module}
             {JOBLIB module}
             {dsname module}
             {dsname(module)}
```

Parameters

The DISPLAY ZAPS command takes these parameters:

STEPLIB

Searches the STEPLIB data set concatenation.

JOBLIB

Searches the JOBLIB data set concatenation.

dsname

Searches the specified data set for the load module. This data set must be cataloged and it must be accessible (read access) to the user issuing the command.

module

Load module to be listed.

Example

```
---------------------- CA-GSS/ISERVE Control Panel --------   ROW 1 TO 1 OF 1
==> display zaps steplib srvsys                                      => PAGE
Target node: *               ssid: *    Host node: B1MAIN         ssid ISRV
================================================================================
No ZAPS applied
```

Usage Notes

If an error occurs in the CA-GSS code, CA Technologies Support will provide you with a fix to be applied with the IBM AMASPZAP program. When you apply the zap, a record will be made within the CA-GSS load module.

GLOBAL

Use the GLOBAL command to display the currently available global variables, along with the values and amount of storage associated with each variable.

Syntax

Use this format for the GLOBAL command:

```
GLobal [ALL [Sort {Name|Members|Storage}]
       [&name [Value] [Sort {Name|Value}]
```

Parameters

The GLOBAL command takes these parameters:

ALL

Displays all global variables. For stem variables, the stem name and a count of members is shown. Also shown is how much storage is being used for data and how much for both data and control structures.

&name

Name of a global variable (case-sensitive) or stem (with trailing period) that you are displaying.

Value

Displays the currently assigned values of each global variable.

Sort

Determines the sort order for output, as follows:

Name

Ascending order by variable name.

Members

Descending order by count of stem members. Where counts are identical, lines are listed in ascending order by variable name.

Storage

Descending order by amount of total storage. Where total storage is identical, lines are listed in ascending order by variable name.

Value

Ascending order by the value (first 20 characters). Where the first 20 characters of values are identical, lines are listed in ascending order by variable name.

Example

```
----------------------- CA-GSS/ISERVE Control Panel ------- ROW 1 TO 19 OF 42
==> global all sort storage                                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
Name or Stem                Members      Storage    Data       Total

&$gnet_xref.                38                      189        4.9k
&iset.                      35                      856        4.5k
&$gnet_node.                19                      113        2.5k
&$gnet_appl.                18                      131        2.4k
&$gnet_conn.                18                      106        2.4k
&$gnet_count1.              18                      76         2.4k
&$gnet_count2.              18                      76         2.4k
&$gnet_count3.              18                      76         2.4k
&$gnet_count4.              18                      76         2.4k
&$gnet_count5.              18                      76         2.4k
&$gnet_count6.              18                      76         2.4k
&$gnet_mode.                18                      130        2.4k
&$gnet_pswd.                18                      76         2.4k
&$gnet_ssid.                18                      76         2.4k
&$gnet_status.              18                      112        2.4k
&$gnet_sysname.             18                      115        2.4kk
```

Fields

Name or Stem

1- to 42-character name of a global variable or stem. Stems are identified by a trailing period.

Members

Number of variables sharing the stem.

Storage

Storage associated with this entry. If the value of the member is more than 1, storage values are cumulative. Values over 1024 bytes are displayed in kilobytes, with a k suffix.

Data

Storage required to store only the data assigned to the variable

Total

Storage used to store data and maintain all associated control structures.

Other Responses

"VALUE" not valid unless name specified

Some form of Value was specified, but your request was for ALL. Value is restricted to requests for a specified variable or stem.

Invalid sort type: "string"

The specified text string is not a valid sort type.

Invalid argument: "string"

The specified text string is not a valid parameter for this command.

No global variables

There are no global variables allocated.

No variables fit selection criteria

There are no global variables of the name that you specified.

GOALNET DEFINE

Use the GOALNET DEFINE command to add GoalNet nodes to your already existing operating network.

Syntax

Use this format for the GOALNET DEFINE command:

```
goalNET DEFine nodename luname [logmode]
```

Parameters

The GOALNET DEFINE command takes these parameters:

nodename

Name of the GoalNet node that you are defining.

luname

Name of the VTAM logical unit over which the node will communicate.

logmode

VTAM logmode name to be used by this node.

Example

```
---------------------- CA-GSS/ISERVE Control Panel ---------  ROW 1 TO 3 OF 3
==> goalnet define test newnode newmode                                 => PAGE
Target node: *               ssid: *    Host node: B1EWF              ssid EWF
================================================================================
Node            System   SSID    APPLID   LOGMODE  Connect  Type
TEST            z/OSB1   ISRV    GSS05N1  NEWMODE  ATINIT   GSS
```

Fields in the Example

Node

GoalNet node name.

System

The GRS system name of the operating system.

SSID

Subsystem ID assigned to the ISERVE owning the node.

APPLID

VTAM LU name over which the node communicates.

LOGMODE

VTAM logmode name used by this node.

Connect

How the node connects with the rest of the network, as follows:

ATINIT

Try to connect at initialization.

ONCMD

Try to connect as directed by a GOALNET START command.

JOIN

Try to connect as directed by a GOALNET START command or a GOALNET JOIN command.

Type

The type of node as defined.

GSS

Assigned to CA-GSS for z/OS.

Usage Notes

- The GOALNET command can be abbreviated to GN, NET, or GNET.
- The GOALNET MODIFY command can be used to supply additional parameters.

GOALNET DISPLAY CONVERSATIONS

Use the GOALNET DISPLAY CONVERSATIONS command to list all active LU 6.2 conversations.

Syntax

Use this format for the GOALNET DISPLAY CONVERSATIONS command:

```
goalNET Display Conversations
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel -------   ROW 1 TO 4 OF 4
==> goalnet display conversations                                  => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
Node        luname     IMOD id          Conv id          Type         Status

B1TRAC      GSS05N7    00000038         01000105          R           Orphan
B1ERWF      GSS05N2    0000019A         01000155          R           Active
B2MAIN      GSS07N1    0000019D         01000157          S           Active
C6TEST      GSS12N2    000001A7         01000161          S           Active
B1TRAC      GSS05N7    000001A8         01000163          R           Active
```

Fields in the Example

Node

GoalNet node name.

Luname

VTAM logical unit name of the session partner.

IMOD id

Task ID of the IMOD that initiated the conversation (hexadecimal).

Conv id

Conversation ID in hexadecimal.

Type

Type of conversation:

S

Send. The conversation was initiated from this IMOD task.

R

Receive. The conversation was started by another node.

Status

The conversation's current status:

Active

The IMOD task is currently communicating over this conversation.

Orphan

The IMOD task that established this conversation ended without first ending the conversation. Use the GOALNET ENDCON command to manually terminate the conversation.

Usage Notes

The GOALNET command can be abbreviated to GN, NET, or GNET.

GOALNET DISPLAY DEFINITIONS

Use the GOALNET DISPLAY DEFINITIONS command to list the definitions of nodes as currently understood by ISERVE.

Syntax

Use this format for the GOALNET DISPLAY DEFINITIONS command:

```
goalNET Display DEFinitions [node]
```

Parameters

The GOALNET DISPLAY DEFINITIONS command takes this parameter.

node

Limits the display to information about one particular node.

Example

```
----------------------- CA-GSS/ISERVE Control Panel -----  ROW 1 TO 19 OF 22
==> goalnet display definitions                                      => PAGE
Target node: *               ssid: *    Host node: B1EWF             ssid EWF
================================================================================
Node        Connect   Type  APPLID   LOGMODE  Sess  Win  Lose

B1MAIN      ATINIT    GSS   GSS05N1  GOALNET  50    25   25
B1TECH      ATINIT    GSS   GSS05N2  GOALNET  50    25   25
B1EWF       ATINIT    GSS   GSS05N3  GOALNET  50    25   25
B1EWF2      ATINIT    GSS   GSS05N4  GOALNET  50    25   25
B1TRAC      ATINIT    GSS   GSS05N6  GOALNET  50    25   25
B1EXPL      ONCMD     GSS   GSS05N7  GOALNET  50    25   25
B1IV41      ATINIT    GSS   GSS05N9  GOALNET  50    25   25
B2TRAC      ATINIT    GSS   GSS07N6  GOALNET  50    25   25
B2EXPL      ONCMD     GSS   GSS07N7  GOALNET  50    25   25
B2TRCT      ATINIT    GSS   GSS07N8  GOALNET  50    25   25
COAS400     ONCMD     GMF   SWICO000 GSS62M   2     1    1
CORS6000    ONCMD     GMF   SWICO100 GSS62M   SNGL  0    0


Node        Connect   Type   T-Out    Retry    Address
TCP_RS6     ATINIT    GMF    30       0        155.035.020.030.05290
TCP_HP9     ATINIT    GMF    30       0        155.035.020.086.05290
TCP_NCR     ATINIT    GMF    30       0        155.035.045.107.05290
TCP_SUN     ATINIT    GMF    30       0        155.035.020.111.05290
```

Fields in the Example

Node

GoalNet node name.

APPLID

VTAM LU name over which the node communicates.

LOGMODE

VTAM logmode name used by this node.

Connect

How the node connects with the rest of the network, as follows:

ATINIT

Try to connect at initialization.

ONCMD

Try to connect as directed by a GOALNET START command.

JOIN

Try to connect as directed by a GOALNET START command or a GOALNET JOIN command.

Type

Type of node as defined.

GSS

Assigned to CA-GSS for z/OS.

Sess

Maximum number of sessions permitted with the node. If the node is single-session capable, SNGL is displayed.

Win

Maximum number of contention-winner sessions permitted with the node.

Lose

Maximum number of contention-loser sessions permitted with the node.

T-Out

Seconds before an unresponsive node is ignored.

Retry

Maximum number of times unacknowledged data is resent.

Address

TCP/IP address for the node.

Usage Notes

The GOALNET command can be abbreviated to GN, NET, or GNET.

GOALNET DISPLAY NODE

Use the GOALNET DISPLAY NODE command to display information about specific nodes.

Syntax

Use this format for the GOALNET DISPLAY NODE command:

```
goalNET Display NODE [node]
```

Parameters

The GOALNET DISPLAY NODE command takes this parameter.

node

Name of the node that you want information about.

Example

```
------------------------ CA-GSS/ISERVE Control Panel  ----------------------
==> goalnet display node b2main                                    => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
Node: B2MAIN          Luname: GSS07N1       Full name: GOAL.GSS07N1
   Parallel_Session_Capable Security_Info_Valid
Mode: GOALNET
  Defined Session Limit: 2   Contention Winners:     Local: 1      Partner: 1
  Minimum Session Limit: 2   Contention Winners:     Local: 1      Partner: 1
  Current Session Limit: 2   Contention Winners:     Local: 1        Partner: 1
  Auto Activate limit: 10       Free Sessions: 0
```

Fields in the Example

The displayed fields are described in the IBM *VTAM Programming for LU 6.2* manual.

Usage Notes

The GOALNET command can be abbreviated to GN, NET, or GNET.

GOALNET ENDCON

Use the GOALNET ENDCON command to manually terminate a GoalNet LU 6.2 conversation.

Syntax

Use this format for the GOALNET ENDCON command:

```
goalNET ENDCON convid
```

Parameters

The GOALNET ENDCON command takes this parameter:

convid

ID of the conversation that you want to terminate.

Example

```
---------------------- CA-GSS/ISERVE Control Panel --------------------------
==> goalnet endcon 01000105                                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
Conversation ended
```

Other Responses

Conversation ID not hexadecimal number

Conversation IDs are always hexadecimal numbers. Use the GOALNET DISPLAY CONVERSATIONS command to obtain a list of current conversations.

Conversation not active

The specified conversation ID could not be linked to an active conversation. If the ID was in the GoalNet-maintained table of conversation IDs, it is deleted.

Usage Notes

The GOALNET command can be abbreviated to GN, NET, or GNET.

GOALNET JOIN

Use the GOALNET JOIN command to attempt communication with all unconnected nodes that have connect classification of either ATINIT or JOIN.

Syntax

Use this format for the GOALNET JOIN command:

```
goalNET JOIN
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel   -----------------------
==> goalnet join                                                    => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
This node now joining GoalNet
```

Usage Notes

- For information about the ATINIT and JOIN connect classifications, see information on installing GoalNet in the CA-GSS for z/OS *Installation Guide*.
- The GOALNET command can be abbreviated to GN, NET, or GNET.
- Each eligible node will be tried asynchronously by the execution of IMOD $GNET_START_NODX. Use the DISPLAY ACTIVE command to follow the progress of the join.

GOALNET MODIFY

Use the GOALNET MODIFY command to change values currently assigned to GoalNet nodes.

Syntax

Use this format for the GOALNET MODIFY command:

```
goalNet MODify nodename [LUNAME luname] [LOGMODE logmode]
  [TYPE {GS}][SESSION {sess[,win[,lose]]}]
                      {SINGLE}]
```

Parameters

The GOALNET MODIFY command takes these parameters:

nodename

Name of the node that you want to modify.

luname

The VTAM logical unit name that is the application ID that will be used to identify the ACB for communication with VTAM. Also used by other GoalNet nodes to establish communication with this node.

logmode

VTAM logmode entry that is used for communications targeted to the named node. Must be defined in the VTAM logmode table associated with the VTAM APPLID definition.

TYPE

The type of node.

GSS

Assigned to CA-GSS for z/OS.

sess

Total number of sessions that are permitted with the node.

win

Total number of contention-winner sessions permitted with the node.

lose

Total number of contention-loser sessions permitted with the node.

SINGLE

Indicates that the partner node is single-session capable.

Example

```
---------------------- CA-GSS/ISERVE Control Panel ---------    ROW 1 TO 3 OF 3
==> goalnet modify test luname newlu logmode newmode type gmf sess 1,1  ==> PAGE
Target node: *               ssid: *    Host node: B1EWF               ssid EWF
================================================================================
Changes for node TEST
LUNAME changed from "OLDNODE" to "NEWLU"
LOGMODE changed from "OLDLOG" to "NEWMODE"
TYPE changed from "GSS" to "GMF"
Session limits: 1 1
```

Usage Notes

The GOALNET command can be abbreviated to GN, NET, or GNET.

GOALNET PURGE

Use the GOALNET PURGE command to terminate all IMODs that are waiting for a response from disconnected nodes.

Syntax

Use this format for the GOALNET PURGE command:

```
goalNET PURGE nodename
```

Parameters

The GOALNET PURGE command takes this parameter:

nodename

Name of the disconnected node.

Example

Here is an example:

```
---------------------- CA-GSS/ISERVE Control Panel  --------------------------
==> goalnet purge B2MAIN                                              => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
Terminating waits for the following IMODs:
000004C2, 00000532, 0000044F
```

Description of Results

A list of IMOD task IDs is produced. Each task listed is waiting for a response from the node in question. The CALLX() function that was in control during the wait indicates the failure (through the RC special variable) to the failing IMOD.

Usage Notes

- If a GoalNet node is unexpectedly terminated (for example, because of abend, system failure, or IPL), or if the necessary communications paths are lost, IMODs that are waiting for routines at disconnected nodes cannot complete. They must be terminated with the GOALNET PURGE command.
- The GOALNET command can be abbreviated to GN, NET, or GNET.

GOALNET RESTART

Use the GOALNET RESTART command to restart communications after a VTAM failure or execution of the GOALNET TERMINATE command.

Syntax

Use this format for the GOALNET RESTART command:

```
goalNET RESTART
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel  --------------------------
==> goalnet restart                                                    => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
GoalNet is being activated
```

Other Responses

OPEN failed: text

The open for the ACB failed for the reason shown in *text*.

Bad ACB status: *text*

The GoalNet ACB is in an unusable status for the reason shown in *text*.

SETLOGON failed: *text*

The VTAM SETLOGON function failed for the GoalNet ACB, for the reason shown in *text*.

Usage Notes

The GOALNET command can be abbreviated to GN, NET, or GNET.

GOALNET START

Use the GOALNET START command to start communication with a single node.

Syntax

Use this format for the GOALNET START command:

```
goalNET START nodename
```

Parameters

The GOALNET START command takes this parameter:

nodename

Name of the node with which you want to start communication.

Example

```
---------------------- CA-GSS/ISERVE Control Panel  --------------------------
==> goalnet start b2main                                              => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
Node B2MAIN is being activated
```

Other Responses

Node nodename **is being activated.**

The specified node is being started.

Node nodename **not defined**

The specified node has not been defined to GoalNet.

Node nodename **is already active**

The specified node has already been started.

Node nodename **may not be activated.**

The connect status of the node is not ATINIT, ONCMD, or JOIN.

Usage Notes

- The GOALNET command can be abbreviated to GN, NET, or GNET.
- The eligible node will be tried asynchronously by the execution of IMOD $GNET_START_NODX. Use the DISPLAY ACTIVE command to follow the join's progress.

GOALNET STATUS

Use the GOALNET STATUS command to display a summary of your current network, as seen from the node executing the command.

Syntax

Use this format for the GOALNET STATUS command:

```
goalNET STATus
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel   -------  ROW 1 TO 19 OF 23
==> goalnet status                                                    => PAGE
Target node: *               ssid: *    Host node: B1EWF              ssid EWF
================================================================================
Status for node: B1EWF   ACB: OPEN      Restart:   AUTO
Node           Status    Type    SYSNAME  SMF      P I      JESNODE

B1MAIN         ACTIVE    GSS     z/OSB1   B1       I        z/OSB1
B1TECH         DOW       GSS     z/OSB1
B1EWF          SELF      GSS     z/OSB1   B1               z/OSB1
B1EWF2         DOWN      GSS     z/OSB1
B1TRAC         ACTIVE    GSS     z/OSB1   B1               z/OSB1
B1EXP          INACTIVE  GSS     z/OSB1
B1IV41         ACTIVE    GSS     z/OSB1   B1       I
B2MAIN         DOWN      GSS     z/OSB2
B2TECH         DOWN      GSS     z/OSB2
B2EWF          INACTIVE  GSS     z/OSB2
B2EWF2         INACTIVE  GSS     z/OSB2
B2PC           INACTIVE  GSS     z/OSB2
B2TRAC         ACTIVE    GSS     z/OSB2   B2               z/OSB2
B2EXPL         INACTIVE  GSS     z/OSB2
CORS6000       INACTIVE  GMF     RS6000
```

Fields in the Example

Node

Node name as defined by the installation.

Status

Current communications status of the node:

ACTIVE

The last (or current) attempt at communication was successful and no notification of failure has been received from VTAM. This node is available for communication.

INACTIVE

No attempt has been made to communicate with this node nor has a communication request been received from the node. An incoming request will change the status to ACTIVE.

DOWN

The last attempt at communicating with this node failed. No further attempt will be made to use this node until a START command is issued against it or until a request is received from the node. An incoming request will change the status to ACTIVE.

STOPPED

A STOP command has been issued against this node (executed from this ISERVE). Until a START command is issued from this ISERVE no further communications will be made to, or accepted from, this node.

READY

The node has been successfully prepared for communications, but none have yet been attempted. An incoming or outgoing request will change the status to ACTIVE.

RECONNECT

While in an ACTIVE state, this node disconnected due to a local VTAM failure (VTAM terminated or application ID VARYed inactive). After access to VTAM is restored, a RESTART command will attempt to restart this node. If your GoalNet is configured to do so, a RESTART command will automatically be executed immediately following VTAM initialization.

SELF

Indicates the node where the STATUS command was executed.

Type

Type of node as defined.

GSS

Assigned to CA-GSS for z/OS.

SYSNAME

GRS system name where the node is executing.

Note: Because this value is determined from initialization parameters, it may not accurately reflect the real system name of the remote node.

SMF

SMF ID of the system where the node is executing.

Note: Because this value is determined from initialization parameters, it may not accurately reflect the real SMF ID of the remote node.

P

If not blank, this indicates that this node is a CA SYSVIEW Performance Management focal point. CA SYSVIEW Performance Management consolidates its activities in a single ISERVE address space per z/OS image.

I

If not blank, this indicates that this node is a CA Insight Database Performance Monitor for DB2 for z/OS focal point. CA Insight Database Performance Monitor for DB2 for z/OS consolidates its activities in a single ISERVE address space per z/OS image.

JESNODE

Name that JES2 uses to route NJE requests.

Note: Because this value is determined from initialization parameters, it may not accurately reflect the actual JES2 NJE node name.

Usage Notes

The GOALNET command can be abbreviated to GN, NET, or GNET.

GOALNET STOP

Use the GOALNET STOP command to terminate communications with a specific node.

Syntax

Use this format for the GOALNET STOP command:

goalNET STOP *nodename*

Parameters

The GOALNET STOP command takes this parameter:

nodename

Name of the node with which you want to terminate communications.

Example

```
----------------------- CA-GSS/ISERVE Control Panel --------------------------
==> goalnet stop b2main                                              => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
Stop for node B2MAIN scheduled
```

Other Results

Node nodename not currently connected.

The specified node was never started.

Usage Notes

- You may also need to use the GOALNET PURGE command to release stranded IMODs.
- The GOALNET command can be abbreviated to GN, NET, or GNET.

GOALNET TERMINATE

Use the GOALNET TERMINATE command to terminate GoalNet communications with all nodes. All IMODs that are currently involved in GoalNet communications are terminated. The ACB is closed.

Syntax

Use this format for the GOALNET TERMINATE command:

```
goalNET TERMINATE
```

Example

```
------------------------ CA-GSS/ISERVE Control Panel --------------------------
==> goalnet terminate                                                 => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
This node terminating GoalNet participation
```

Usage Notes

The GOALNET command can be abbreviated to GN, NET, or GNET.

IVP

Use the IVP command to execute CA-GSS' Installation Verification Procedure, which verifies that the CA-GSS installation is successful.

This command will either display an abbreviated system summary or print an extended one.

Syntax

Use this format for the IVP command:

```
IVP [PRINT [TO userid [AT node]]
```

Parameters

The IVP command takes these parameters:

PRINT

Dynamically allocates a SYSOUT data set and prints an extended system summary.

Default: An abbreviated system summary is returned to the requestor

userid

User to whom the SYSOUT data set is routed.

node

NJE node to which the SYSOUT data set is routed.

Examples

```
---------------------- CA-GSS/ISERVE Control Panel ---------   ROW 1 TO 2 OF 2
==> ivp                                                                 => PAGE
Target node: *               ssid: *    Host node: B1EWF              ssid EWF
================================================================================
CA-GSS/ISERVE version 02.07.00 is active
Use the PRINT option to obtain detailed report
```

KILL

Use the KILL command to terminate a true subtask that is failed to complete or that is looping.

Syntax

Use this format for the KILL command:

```
KILL TCB address
```

Parameters

The KILL command takes this parameter:

address

Address of the subtask's TCB.

Example

```
------------------------ CA-GSS/ISERVE Control Panel ---------  ROW 1 TO 1 OF 1
==> kill tcb 6e5a10                                                    => PAGE
Target node: *                ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
DETACH issued against 006E5A10
```

Other Responses

TCB specification not hexadecimal address

The specified address is not a valid hexadecimal value.

Invalid target specified

The string TCB did not follow the KILL command.

Usage Notes

- CA-GSS processes ADDRESS commands, SVCs, and certain functions in true subtasks.
- You can use the DISPLAY ADDRESS command to display a subtask's TCB address.

LOAD

Use the LOAD command to load an IMOD from an ISET and make it available for use.

Syntax

Use this format for the LOAD command:

```
LOAD IMOD imodname FROM isetname
```

Parameters

The LOAD command takes these parameters:

imodname

IMOD's name.

isetname

Name of the ISET from which you wish to load the IMOD.

Example

```
------------------------ CA-GSS/ISERVE Control Panel --------------------------
==> load imod newimod from prod                                        => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
NEWIMOD Loaded from INTERNAL at 05658C58 length: 000003A8
```

Other Responses

IMOD name invalid

The IMOD name you entered is syntactically incorrect. Check the spelling and the length of the name.

Invalid argument: xxxx

The specified text (xxxx) is invalid. Make sure that your command includes both the IMOD and FROM parameters.

IMOD NOT FOUND

There's no such IMOD in the specified ISET.

FROM value invalid

There's no such ISET known to this ISERVE address space.

IMOD NOT IN PRODUCTION STATUS

The IMOD is currently in test status. Only IMODs with production status can be loaded.

IMOD COMPILED WITH ERRORS

The IMOD contains compile-time errors. Only IMODs that have been successfully compiled can be loaded.

IMOD NOT COMPLETE IN LIBRARY

The IMOD has not been compiled since the last time it was modified. The LOAD command could find no object code for this IMOD. Or, this message may indicate a damaged IMOD or ISET.

LOGON DEFINE

Use the LOGON DEFINE command to define an application to CA-GSS' Logon Facility.

Syntax

Use this format for the LOGON DEFINE command:

```
LOGON DEFine appl imodname [argument]
```

Parameters

The LOGON DEFINE command takes these parameters:

appl

Application name. Specify required characters in uppercase and optional characters in lowercase.

Note: At logon time, case is ignored.

imodname

Name of the application's initialization IMOD.

argument

Information being passed to the application.

Example

```
--------------------- CA-GSS/ISERVE Control Panel -----------   ROW 1 TO 9 OF 9
==> logon define MYAPpl imod_myappl parm1                                 => PAGE
Target node: *               ssid: *    Host node: S18MAIN             ssid ISRV
================================================================================
Application MYAPPL has been defined
Application             IMOD                    Arguments

OPERator                $SRVV
RView                   $RVIEW2                 menu3
RUNimod                 $SESS_IMOD
TEST                    $SESS_TEST
XPCall                  $FC_COMM
MYAPpl                  $IMOD_MYAPPL            parm1
```

Fields in the Example

For information about these fields, see "LOGON DISPLAY DEFINITION (see page 531)".

LOGON DELETE

Use the LOGON DELETE command to remove an application from the CA-GSS Logon Facility.

Syntax

Use this format for the LOGON DELETE command:

```
LOGON DELete appl
```

Parameters

The LOGON DELETE command takes this parameter:

appl

Application name. This name must match the application name set using the LOGON DEFINE command. (Optional characters can be omitted and case is ignored.)

Example

```
--------------------- CA-GSS/ISERVE Control Panel -----------   ROW 1 TO 8 OF 8
==> logon delete myappl                                                 => PAGE
Target node: *                ssid: *    Host node: S18MAIN            ssid ISRV
================================================================================
Application MYAPPL has been deleted
Application           IMOD                 Arguments
--------------------  -------------------- --------------------
OPERator              $SRVV
RView                 $RVIEW2              menu3
RUNimod               $SESS_IMOD
TEST                  $SESS_TEST
XPCall                $FC_COMM
```

Fields in the Example

For information about these fields, see LOGON DISPLAY DEFINITION. (see page 531)

LOGON DISPLAY DEFINITION

Use the LOGON DISPLAY DEFINITION command to display the current values for the CA-GSS Logon Facility.

Syntax

Use this format for the LOGON DISPLAY DEFINITION command:

```
LOGON Display DEFinition
```

Example

```
--------------------- CA-GSS/ISERVE Control Panel -----------   ROW 1 TO 8 OF 8
==> logon display definition                                              => PAGE
Target node: *               ssid: *    Host node: S18MAIN            ssid ISRV
================================================================================
LUNAME:  GSS05
Application    IMOD             Arguments
-------------  --------------  --------------------
OPERator       $SRVV
RView          $RVIEW2          menu3
RUNimod        $SESS_IMOD
TEST           $SESS_TEST
XPCall         $FC_COMM
```

Fields in the Example

LUNAME

VTAM logical unit that is assigned to the Logon Facility.

Application

Application name accepted at logon. Required characters are specified in uppercase.

IMOD

IMOD that receives control for a logon to the application.

Arguments

Optional argument passed to the IMOD when this application invokes it.

MONITOR

Use the MONITOR command to monitor one or more address spaces. All WTOs from a monitored address space can be recorded on an ILOG or processed by an IMOD.

Syntax

Use one of these formats for the MONITOR command:

```
MONitor ADD {JOB jobname} {IMOD imodname|ILOG ilognum}
            {STC stcname}
            {TSU userid}
            {DASID dasid}
            {XASID xasid}
MONitor DELete {DASID dasid|XASID xasid}
MONitor STATus
```

Parameters

The MONITOR command takes these parameters:

ADD

Monitors the address space.

DELete

Stops monitoring the address space.

STATus

Displays currently monitored address spaces.

JOB jobname

Identifies the job that you want to monitor.

STC stcname

Identifies the started task that you want to monitor. (If started tasks have duplicate names, you must use the ASID specification.)

TSU userid

Identifies the address space (decimal) that you want to monitor.

DASID dasid

Identifies the address space (decimal) that you want to monitor.

XASID xasid

Identifies the address space (hexadecimal) that you want to monitor.

IMOD imodname

Executes an IMOD when the address space issues a WTO.

In place of *imodname*, specify a two-digit suffix for the IMOD's name (from 00 to 14, inclusive). The full IMOD name is ASID_*nn*.

ILOG ilognum

Records WTOs issued from the address space in the specified

Example

```
------------------------ CA-GSS/ISERVE Control Panel -------------------------
==> monitor add tsu ewf imod 00                                        => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
ASID (hex)     Jobname         ILOG        IMOD
----------     --------        ----        -------
   93          005D            DSNMSTR     1
   95          005F            DSNDBM1     1
   99          0063            IRLMPROC    1
  100          0064            DSNDIST     1
  108          006C            EWF         ASID_00
```

Fields in the Examples

After you have issued a MONITOR ADD command, CA-GSS displays the status of all monitored address spaces, as follows:

ASID

ID of the address space (decimal and hexadecimal) that is being monitored. All address spaces that are being monitored are summarized.

Jobname

Job name, TSO user ID, or started-task name associated with the address space when monitoring was requested.

ILOG

ILOG that is being used to record the text of all WTOs issued from the address space.

IMOD

IMOD that is being executed for WTOs issued from the address space.

Other Responses

ASID, JOB, STC, or TSU not specified

You did not indicate which address space you want to monitor.

Neither IMOD nor ILOG specified

You failed to specify IMOD or ILOG followed by an appropriate value.

JOB NOT FOUND

The specified jobname, TSO user ID, or started-task name is invalid or could not be located on the system.

IMOD value is invalid

The IMOD value must be a decimal number in the range of 0 through 14, inclusive.

ILOG value is invalid

The ILOG value must be a decimal number in the range of 0 through 14, inclusive.

ASID value is invalid

The specified address space does not exist on your system, or the value for XASID or DASID is syntactically invalid.

"ASID" is no longer valid

Use "XASID" to specify the ASID in hexadecimal
Use "DASID" to specify the ASID in decimal
The value asid is no longer a valid entry. Specify either xasid or dasid.

PEND

Use the PEND command to list the first line of each WTO that is in the process of being completed.

Syntax

Use this format for the PEND command:

```
PEND
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel ---------   ROW 1 TO 1 OF 1
==> pend                                                                 => PAGE
Target node: *               ssid: *    Host node: B1MAIN            ssid ISRV
================================================================================
*** No messages pending
```

Usage Notes

Before an IMOD can be scheduled to process a WTO, the entire WTO text must be available. Since tasks can issue multiple SVCs to build a single multi-line WTO, a considerable length of time may be required to obtain the complete message.

PURGE

Use the PURGE command to eliminate a dormant IMOD task.

Important! Before you purge a dormant IMOD, be sure that it will not be reclaimed by the requestor.

Syntax

Use this format for the PURGE command:

PURGE *taskid*

Parameters

The PURGE command takes this parameter:

taskid

The hexadecimal IMOD ID, as shown on the DISPLAY ACTIVE screen.

Example

```
---------------------- CA-GSS/ISERVE Control Panel ---------   ROW 1 TO 8 OF 8
==> purge 10ab                                                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN            ssid ISRV
================================================================================
IMOD 10AB Scheduled for PURGE
```

Other Responses

IMOD taskid Not Found

The specified IMOD task could not be found. Issue a DISPLAY ACTIVE command and verify the correct ID.

IMOD taskid Not valid for PURGE

The specified IMOD task is not in a dormant state. Only dormant IMOD tasks are eligible for PURGE.

IMOD taskid Not purged: value

An unexpected result was returned by purge processing. The value returned will be useful to CA Technologies Support.

Usage

Some IMODs that are executed at the request of cross-memory users (notably CA Jobtrac Job Management) are assigned a *recoverable* attribute. This means that if a requestor terminates while an IMOD is being executed on its behalf, the IMOD cannot fully complete until the requestor is restarted and *claims* the IMOD's results.

An IMOD that has finished processing and is waiting to be reclaimed displays as DORMANT on the DISPLAY ACTIVE screen.

REPORT ILOG

Use the REPORT ILOG command to print the contents of an ILOG file. Files are printed in ascending chronological order.

Syntax

Use this format for the REPORT ILOG command:

```
REPORT ILOG ilognum [TO userid [AT node]
```

Parameters

The REPORT ILOG command takes these parameters:

ilognum

Name of the ILOG.

userid

User ID to which the printout is to be routed.

Default: Output is routed to the current started task.

node

NJE node to which the printout is to be routed.

Default: Output is routed to the current started task.

Example

```
---------------------- CA-GSS/ISERVE Control Panel  -------------------------
==> report ilog 0 to ewf at gnode                                      => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
ILOG Listing is complete
```

Other Responses

ILOG file number invalid

The ILOG file that you specified does not represent one currently defined to ISERVE.

'AT' invalid without 'TO'

You cannot specify an NJE node unless you also specify a user ID.

ILOG file is empty

There is no data to report in the specified file.

Invalid workstation

JES refused to allocate the output file to the specified NJE node or user ID.

RESET ILOG

Use the RESET ILOG command to make a full ILOG file available for use again.

Syntax

Use this format for the RESET ILOG command:

```
RESET ILOG file.subfile
```

Parameters

The RESET ILOG command takes these parameters:

file

Number of the ILOG file. Specify a value from 0 to 16 (inclusive).

subfile

Number of the subfile. Specify a value from 0 to 9 (inclusive).

Example

```
----------------------- CA-GSS/ISERVE Control Panel --------------------------
==> reset ilog 0.2                                                    => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
ILOG RESET
```

Other Responses

Invalid ILOG number: *nn*

The ILOG file number *nn* is not valid. *nn* must be between 0 and 16 inclusive.

Invalid subfile number: *nn*

The subfile number *nn* is not valid. *nn* must be between 0 and 9 inclusive.

SCHEDULE

Use the SCHEDULE command to schedule the asynchronous execution of an IMOD.

Syntax

Use this format for the SCHEDULE command:

```
SCHedule imod [args]
```

Parameters

The SCHEDULE command takes these parameters:

imod

Name of the IMOD that you want to schedule for execution.

args

Arguments that you want to pass to the IMOD.

Example

```
-------------------   CA-GSS/ISERVE Control Panel -----------   ROW 1 TO 1 OF 1
==> schedule $srvc x d act                                                 => PAGE
Target node: *               ssid: *    Host node: S18MAIN             ssid ISRV
================================================================================
=
IMOD scheduled, taskid 2AEF
```

Usage Notes

The IMOD is executed as a separate IMOD task. Control is returned to the operator as soon as the IMOD is scheduled.

SERVER

Use the SERVER command to control the Server IMOD Facility. This facility is described in detail in the *CA-GSS for z/OS IMOD Guide*.

Syntax

Use this format for the SERVER command.

```
SERVER {CANCEL name}
       {CYCLE name}
       {DEFINE name imod desc}
       {DELETE name}
       {DISPLAY}
       {HOLD name}
       {PAUSE name}
       {QUIESCE name}
       {RESET name}
       {RESUME name}
       {START name}
       {STATUS name}
       {STOP name}
       {TERMINATE name}
```

Parameters

The SERVER command takes these parameters.

name

Name of a service.

imod

Name of the IMOD that will provide the service.

desc

Description of the service (for display purposes only). Case is preserved.

CANCEL

Cancels the Server IMOD. This command should be avoided since the current request (and all queued requests) is abandoned and the requesting IMODs are *not* notified.

CYCLE

Cancels the current Server IMOD and starts a new server. Queued requests are transferred to the new server.

DEFINE

Creates a new service name and links it to the IMOD that will provide the service. An optional description will be displayed during later informational commands.

DELETE

Deletes an existing service. The currently executing service IMOD task, if any, is not terminated or disturbed in any way. However, it will no longer be recognized as being related to the service.

DISPLAY

Returns information on all currently defined servers on the stack. The first character of each stack record serves as a delimiter character. The fields returned are (in order): Service, IMOD name, IMOD task ID (if running), and Description.

HOLD

Continues to queue requests without processing requests in the queue.

PAUSE

Stops accepting additional requests and does not process requests already in the queue.

QUIESCE

Stops accepting additional requests but continues processing those that are already queued.

RESET

Deletes all unprocessed requests. No processing is performed and the requestors are *not* notified.

RESUME

Cancels the effects of any outstanding HOLD, PAUSE, or QUIESCE.

START

Starts the named service. Any specified options are passed to the Server IMOD in the argument string, following the service name.

STATUS

Returns the status of a particular server IMOD on the stack. The information returned is formatted for display purposes.

STOP

Performs a QUIESCE operation and then a TERMINATE operation when all queued requests have been completed.

TERMINATE

Terminates processing immediately. Any unprocessed requests are ignored.

Examples

```
---------------------- CA-GSS/ISERVE Control Panel ---------    ROW 1 TO 1 OF 1
==> server define stepsum stepsum CA-SYSVIEW Stepsum Server             => PAGE
Target node: *               ssid: *    Host node: B1EWF              ssid EWF
================================================================================
Success
```

SET EDITOR

Use the SET EDITOR command to change IMOD editor values that were set using the EDITOR initialization parameter.

Syntax

Use this format for the SET EDITOR command:

```
SET EDitor {CLIST name}      [Global]
           {LLIB dsname}
           {MEMBER name}
           {MLIB dsname}
           {PARMLIB dsname}
           {PLIB dsname}
           {VIO vioname}
```

Parameters

The SET EDITOR command takes these parameters:

CLIST name

Library containing CLISTs used by the IMOD editor.

LLIB dsname

Your ISPF load library.

MEMBER name

Member of the PARMLIB data set that contains the IMOD editor's parameters and its ISET menu.

MLIB dsname

Your ISPF message library.

PARMLIB dsname

Your CA-GSS parameter library.

PLIB dsname

Your ISPF panel library.

VIO vioname

Unit name that you use for VIO.

Global

Sets the specified value in the global area, which is normally set during the primary ISERVE's initialization.

Default: Affects local storage only (which is returned in response to the IMOD editor's LINK command).

Example

```
---------------------- CA-GSS/ISERVE Control Panel ---------  ROW 1 TO 16 of 16
==> set editor member temp                                                => PAGE
Target node:   *              ssid:   *   Host node: S18EWF               ssid EWF
================================================================================
IMOD Editor Values
------------------------------
Global: VIO Unit:                VIO
        Parmlib                EWF.FXAO.PARMLIB
        Parmlib Member         NEWEDIT
        Panel Library:         EWF.FXAO.ISRPLIB
        Message Library        EWF.FXAO.ISRMLIB
        Load Library           EWF.FXAO.AUTHLIB
        CLIST Library          EWF.FXAO.CLIST
Local:  VIO Unit                VIO
        Parmlib          EWF.FXAO.PARMLIB
        Parmlib Member         NEWEDIT
        Panel Library           EWF.FXAO.ISRPLIB
        Message Library:        EWF.FXAO.ISRMLIB
        Load Library            EWF.FXAO.AUTHLIB
        CLIST Library          EWF.FXAO.CLIST
```

SET GLOBAL

Use the SET GLOBAL command to assign values to REXX global variables.

Syntax

Use this format for the SET GLOBAL command:

```
SET GLOBAL &var [/newvalue/]
```

Parameters

The SET GLOBAL command takes these parameters:

&var

Name of a new or existing global variable.

Note: Global variable names are case-sensitive.

/newvalue/

Character string to be assigned to the variable. It must be enclosed in arbitrary delimiter characters (such as the / used in the syntax diagram).

Default: Null string.

Example

```
------------------------ CA-GSS/ISERVE Control Panel -----   ROW 1 TO 2 OF 2
==> set global &ewf /123456/                                        => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
Variable &ewf set to 123456
Old value: abcdef
```

SPIN

Use the SPIN command to close and spin-off the current ISRVLOG or GNETLOG data set. Logging is resumed on a new, dynamically allocated file.

Syntax

Use this format for the SPIN command:

```
SPIN {ISRVLOG|GNETLOG}
```

Example

```
----------------------- CA-GSS/ISERVE Control Panel ---------  ROW 1 TO 3 OF 3
==> spin isrvlog                                                      => PAGE
Target node: *                ssid: *    Host node: B1EWF            ssid EWF
================================================================================
LOG closed
New LOG recording on SYS00030
OUTPUT JCL REFERENCE (ISRVOUTP) MISSING OR INVALID
```

Other Responses

OUTPUT JCL REFERENCE (ISRVOUTP) MISSING OR INVALID

The //ISRVOUTP OUTPUT statement was not included in the start-up JCL. ISERVE uses information on this card to assign attributes to the newly allocated data set.

STATUS

Use the STATUS command to display the current status of an ISERVE address space.

Syntax

Use this format for the STATUS command:

```
STATUS
```

Example

```
---------------------- CA-GSS/ISERVE Control Panel  -------------------------
==> status                                                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN           ssid ISRV
================================================================================
Subsystem name: ISRV    JES NJE Node Name: SYSTEM18
SMF id:         B1      GRS System name:  z/OSB1
Jobname: ISERVET       Job ID:  STC01217
GoalNet Node: S18EWF    GoalNet LUname:  GSS05N3
GMF Name (s)   S18EQF
Elapsed time: 38:30    CPU Time: 8.50
IMODs available. 280
IMODs executed 533
EXECs executed 3,265
Instructions executed   171,169
```

Fields in the Example

Subsystem name

Subsystem for the ISERVE being reported.

JES NJE Node Name

NJE node name for this node.

SMF id

SMF ID of the z/OS system where the ISERVE is executing.

GRS System name

SYSNAME for the z/OS system where the ISERVE is executing.

Jobname

Address space's job name.

Job ID

Address space's job ID.

GoalNet Node

Address space's GoalNet node.

GoalNet LUname

LU name used by GoalNet.

Elapsed time

Length of time since this ISERVE was last initialized (in hh:mm:ss format).

CPU Time

Total computer time used by this ISERVE since initialization (in hh:mm:ss.th format).

IMODs available

Number of IMODs that have been loaded into the ISERVE address space, including those that are disabled.

IMODs executed

Total number of IMOD tasks that have been executed, including those currently executing, since ISERVE was initialized.

EXECs executed

Total number of IMODs executed since ISERVE initialization. This includes the base IMOD for each IMOD task and each instance of an IMOD being executed as an external subroutine or function.

Instructions executed

Total number of REXX instructions executed since ISERVE was initialized.

STOP

Use the STOP command to terminate ISERVE processing and, if issued against the primary ISERVE (running in the CA-GSS address space), to terminate CA-GSS. Termination occurs after all currently executing IMODs are complete. Additional IMODs can be scheduled while waiting.

Syntax

Use this format for the STOP command:

```
STOP [FORCE]
```

Parameters

The STOP command takes this parameter:

FORCE

Causes immediate termination. Use with care, as IMODs are terminated at their current point of execution.

Example

```
----------------------- CA-GSS/ISERVE Control Panel -----------------------
==> stop                                                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
ISERVE Shutdown in Progress
```

Other Responses

Invalid option: xxx

You specified an invalid option after the STOP verb. The only valid option is FORCE.

Usage Notes

When using the STOP command:

- After entering the STOP command, you can still use the DISPLAY ACTIVE command to determine what IMODs, if any, are still processing.
- You may need to use the CANCEL command to selectively cancel IMODs to obtain a complete shutdown.

SWITCH

Use the SWITCH command to close an ILOG file, mark it *dump required*, and switch recording to the next available ILOG subfile.

Syntax

Use this format for the SWITCH command:

```
SWITCH ILOG ilognum
```

Parameters

The SWITCH command takes this parameter:

ilognum

Number of the ILOG you want switched to a new subfile.

Example

```
----------------------- CA-GSS/ISERVE Control Panel -----------------------
==> switch ilog 1                                                  => PAGE
Target node: *               ssid: *    Host node: B1MAIN         ssid ISRV
================================================================================
Log File Status     Record      Used    First Record         Last Record

1 Full-Avail        120 25.4%   06/01/25 13.27.16   06/01/28  10.17.53
1 Full-Avail        4           0.8%    06/01/28 10.34.57   06/01/28  10.44.55
2 Active            0           0.0%
```

Fields in the Example

Log

ILOG file number.

File

Subfile within the ILOG.

Status

ILOG subfile's current recording status:

Active	Currently being recorded to (one per ILOG).
Full-Avail	Full of data; will be reset to empty when needed for additional recording.
Dump-Req	Full of data; will not be reused until a RESET command is issued for it.

Record

Number of records in the subfile.

Used

Amount of the subfile that contains data, expressed as a percentage of full.

First Record

Date and time stamp of the earliest (first) record in the subfile.

Last Record

Date and time stamp of the latest (last) record in the subfile.

Other Responses

Invalid option: xxxx

The command contains an invalid option, *xxxx*.

ILOG number invalid

The specified ILOG number does not represent a valid ILOG file.

TRACE

Use the TRACE command to change the current setting of the TRACE flags.

Syntax

Use this format for the TRACE command:

```
TRACE {ALTer}    {ON|OFF}
      {DEBug}
      {FAILed}
      {LOCation}
      {RESult}
      {SCHEDule}
      {SPACE}
      {SUBcall}
```

Parameters

The TRACE command takes these parameters.

ALTer

Causes a trace entry on ISRVLOG for each use of the MEMORY() function that alters storage.

DEBug

Causes a trace entry on ISRVLOG of various internal debugging data. This information is useful to CA Technologies Support.

FAILed

Causes a trace entry on ISRVLOG for each IMOD that fails during scheduling.

LOCation

Causes a trace entry on ISRVLOG each time an IMOD is loaded or replaced with a new copy. This IMOD's location and length will be traced.

OFF ON

Disables or enables tracing of the event.

RESult

Causes a trace entry on ISRVLOG at the conclusion of each IMOD that returns a non-null string. The value returned by the IMOD is recorded.

SCHEDule

Causes a trace entry on ISRVLOG for each IMOD task that is scheduled.

SPACE

Causes a trace entry on ISRVLOG at the conclusion of each IMOD, listing the IMOD's memory use. IMODs that do not release all memory are always traced.

SUBcall

Causes a trace entry on ISRVLOG of each call to an external subroutine (IMOD).

Example

```
----------------------- CA-GSS/ISERVE Control Panel -------------------------
==> trace debug sched sub on                                          => PAGE
Target node: *               ssid: *    Host node: B1MAIN          ssid ISRV
================================================================================
FAILed to run: ON       SPACE usage:    ON          imod LOAD:  ON
imod SCHEDuled:ON       member ALTER:  ON          RESULT:      OFF
DEBUG:         ON       SUBCALL:        OFF
```

Fields in the Example

FAILed to run

Indicates whether a trace entry is being made to ISRVLOG for each IMOD that is requested but not available.

SPACE usage

Indicates whether a trace entry is being made to ISRVLOG at the conclusion of each IMOD, listing the IMOD's memory utilization. IMODs that do not release all memory are always traced.

imod LOAD

Indicates whether a trace entry is being made to ISRVLOG each time an IMOD is loaded or replaced with a new copy. This IMOD's location and length will be traced.

imod SCHEDuled

Indicates whether a trace entry is being made to ISRVLOG each time an IMOD is schedule for execution.

member ALTER

Indicates whether a trace entry is being made to ISRVLOG for each use of the MEMORY() function to alter storage.

RESULT

Indicates whether a trace entry is being made to ISRVLOG at the conclusion of each IMOD that returns a non-null string. The value returned by the IMOD is recorded.

DEBUG

Indicates whether a trace entry is being made to ISRVLOG of various internal debugging data. This information is useful to CA Technologies Support.

SUBCALL

Indicates whether a trace entry is being made to ISRVLOG each time an IMOD makes a subroutine call to another IMOD.

CA-GSS Initialization Parameters

Interpret Syntax Diagrams

A syntax diagram is provided for each CA-GSS initialization parameter or service routine. You can use these diagrams to get an overview of the information provided by the parameter/routine and to get pertinent information on values for the parameter/routine.

Make sure you read the Usage Notes section for a parameter/routine. This section can contain additional syntax information that is not shown in the diagram.

How to Specify CA-GSS Initialization Parameters

Execution of CA-GSS is controlled by options specified using the PARMLIB DD statement. The available parameters cover a wide range of required and optional values.

Format

The records that are read through the PARMLIB DD statement are RECFM F, V, FB, or VB. LRECL and BLKSIZE can be any value. Individual records are free-format and adhere to the following rules:

- All columns are examined; card sequence numbers (columns 73-80) must not appear.
- Leading and trailing blanks are removed.
- Any record whose first non-blank character is an asterisk (*) is ignored.
- Blank lines are ignored.
- Keywords are not case-sensitive. Parameters can be case-sensitive, depending upon context.
- Records are scanned left to right.
- When the characters /* or // are encountered in a record, all remaining text on the record is ignored.
- Any record whose last character is a hyphen (-) will be treated as follows:
  - The hyphen will be replaced with a blank.
  - The next record will be read.
  - After leading and trailing blanks are removed, trailing comments (// or /*) are stripped, and tests for a leading * are made, the resulting text is appended to the preceding, continued record (separated by one blank).
  - Records can be continued as many times as necessary.
  - Comments (records beginning with *) are ignored and not considered as part of a continued record.
  - A continuation indicator that follows /* or // is ignored.
- PARMLIB members can be nested by using the INCLUDE parameter.

Specify the Parameter Source

All initialization parameters are provided using the PARMLIB DD statement. This DD statement can point to a sequential data set, SYSIN data set, or partitioned data set (PDS). If PARMLIB is specified as a sequential or SYSIN data set, only that data set is processed (multiple data sets can be concatenated in the JCL).

If PARMLIB specifies a PDS (and no member name is included in the DD entry), processing proceeds as follows:

- The EXEC card PARM field is examined and the first blank-delimited word is assumed to be the initial member name for PARMLIB processing.
- If a member name is not specified in the PARM field, or if the first word is *, the default member is SRVPARM.
- Whenever an INCLUDE parameter is encountered, processing of the current member is suspended and processing of the newly specified member is begun.
- At end-of-file, processing resumes with the suspended member and the record following the INCLUDE. When end-of-file is encountered in the initial member, PARMLIB processing completes.
- Although a member can be included more than once, an INCLUDE statement cannot specify a currently suspended member (recursion).

Parameter Overview

The following table provides a brief description of each CA-GSS initialization parameter. More detailed information about each parameter is provided in the following sections.

Parameter	Description
ADDRESS	Identifies an ADDRESS environment and its associated load module.
ALTNAME	Defines an alternative name for an address.
AUTOMATION	Provides parameters for CA OPS/MVS Event Management and Automation initialization IMOD.
CMD	Adds a CA-GSS operator command.
COMMAND	Defines an OS operator console command word and links it with an IMOD.
DB2PLAN	Identifies the DB2 plan used for dynamic SQL requests.
DUMP	Routes CA-GSS-generated diagnostic dumps to a specific user or to a specific SYSOUT class.
EDITOR	Defines default data sets for the IMOD editor and the CA-GSS/ISERVE Control Panel.

Parameter	Description
EVENTS	Determines the size of the internal EVENTS table.
FUNCTION	Identifies an external function and its associated load module.
GLOBVAL	Defines a REXX global variable and assigns it an initial value.
GOALNET	Defines a GoalNet node.
GOALNETLOCAL	Defines the GoalNet name for the local system.
HELPDSN	Identifies the ISPF panel library containing CA-GSS ISPF help panels.
ILOG	Defines an ILOG data set.
INCLUDE	Nests PARMLIB data set members.
INSIGHT	Provides parameters for CA Insight Database Performance Monitor for DB2 for z/OS initialization IMOD.
ISET	Defines an ISET for loading IMODs.
JESNODE	Defines the JES node name where this CA-GSS started task is executing.
JOBTRAC	Provides parameters for the CA Jobtrac Job Management initialization IMOD.
LOGLINES	Determines the maximum number of lines written to CA-GSS ISRVLOG data set.
LOGON	Defines and activates CA-GSS Logon Facility.
MAXSTACK	Limits the size of IMOD stacks.
MAXXREQ	Determines the maximum number of active, cross-memory requests.
PRIMARY	Indicates whether this CA-GSS address space is the primary one.
PRODUCT	Identifies CA products that require special CA-GSS support.
RERUN	Provides parameters for the CA 11 Workload Automation Restart and Tracking initialization IMOD.
SCHEDULE	Executes a list of IMODs immediately after the system initializes.
SECURITY	Indicates what type of security software is installed on your system.
SLICE	Determines maximum number of REXX clauses that an IMOD can execute before giving control to another IMOD.
SSID	Identifies an external product's subsystem name.
SSNAME	Defines a subsystem name for this CA-GSS address space.

Parameter	Description
STORAGE	Forces CA-GSS work areas into 24-bit addressable storage.
SYSVIEWE	Provides parameters for CA SYSVIEW Performance Management initialization IMOD.
TCP/IP	Enables GMF to use the TCP/IP protocol for communications.
TRACE	Sends trace output to CA-GSS ISRVLOG data set.
USER	Provides parameters for the installation-provided initialization IMOD.
VIEW	Provides parameters for CA View initialization IMOD.
VIOUNIT	Defines the unit name for VIO requests.
VTAMRESTART	Indicates what action CA-GSS should take if VTAM fails and restarts.
WTO	Provides criteria for intercepting WTOs and executing IMODs.

ADDRESS

Use the ADDRESS parameter to define a REXX ADDRESS environment.

Syntax

Use this format for the ADDRESS parameter:

```
ADDRESS name module [limit [DETACH] [MAXTASK nnn] [TYPE ttt]
```

Arguments

The ADDRESS parameter takes these arguments:

Argument	Description
name	ADDRESS environment name. Most environments provided by CA or used by CA products require a specific name, either a name for the ADDRESS or one provided by an ALTNAME statement.
module	Name of the load module that will process the ADDRESS environment. In general, the load modules required are provided with the product associated with the service. Load modules must be accessible to CA-GSS using STEPLIB/JOBLIB/LINKLIST search. Modules must be link-edited with correct linkage parameters.

Argument	Description
limit	For reliability, ADDRESS code is executed in subtasks. To prevent possible looping, you can specify the maximum amount of CPU time permitted for processing a single ADDRESS instruction. Specify the value in seconds. Specify 0 (zero) for unlimited time. The default is 15.
DETACH	Detaches the subtask that processed the ADDRESS command following completion of the IMOD task that used it.
MAXTASK *nnn*	Maximum number of concurrently executing IMOD tasks that can use this ADDRESS environment. Specify a value from 0 to 32,767 (inclusive). Specify 0 to prevent any limitation. The default is 10. An IMOD task attempting to obtain the use of an ADDRESS environment that is already at its MAXTASK value is suspended until another IMOD task relinquishes use of the ADDRESS environment.
TYPE *ttt*	CA-GSS supports a variety of interfaces for ADDRESS environment load modules. To identify which interface is required, you must provide the TYPE keyword. TYPE 0 environments are TSO/E REXX compatible. TYPE 255 environments cannot be directly invoked. For other types, see "Preferred Names". Default: CA-GSS tries to determine the type by examining the ADDRESS name for forms that it recognizes.

Usage Notes

- Whenever an IMOD task issues an ADDRESS command, a subtask is assigned to the IMOD task. Additional ADDRESS requests of the same name are referred to the same subtask for the life of the IMOD task. When the IMOD task ends, the subtask is released for use by another IMOD task or is detached.
- For security, each ADDRESS-processing subtask runs under the user ID of the entity that started the IMOD task.
- For charge-back purposes, CPU time accrued by an ADDRESS-processing task is passed back to the IMOD task.
- For information on writing your own ADDRESS environments, see "Providing Your Own Functions and ADDRESS Environments."

Preferred Names

The preferred names for CA products are shown in the following table. See the notes at the end of the table for information corresponding to the numbers (1) and (2) associated with the items:

Product	Address Name	Load Module	Type	MAXTASK	DETACH
DB2 (function)	DSNALI (2)	DSNALI	255	-	No
	DSNHLI2 (2)	DSNHLI2	255	---	No
IDCAMS (IBM)	IDCAMS	IDCAMS	1	-	No
INFO/MGMT (IBM)	INFO	BLGYRXM	0	-	Yes
CA OPS/MVS Event Management and Automation	OPER	OPGLEVMG	10	-	No
	OPSREQ	OPGLEVMG	11	-	No
	AOF	OPGLEVMG	12	-	
	OSF	OPGLEVMG	13	-	
	OPSVALUE (2)	OPGLEVMG	255	-	
CA SYSVIEW Performance Management	SYSVIEWE (1)	GSVXAPIE	0	-	Yes
	COMMAND	GSVXAPIE	0	-	Yes
CA Jobtrac Job Management	JOBTRAC	GJTRGCUU	3	5	No
CA MIM Resource Sharing	TPCF	MIMAPI1	7	-	No
CA 11 Workload Automation Restart and Tracking	RERUN	RUNTADDR	3	5	No
CA View	XPVIEW	SARINTF	0	-	No
CA 7 Workload Automation	CA7	CAL2X2WR	0	15	No

Table Notes

1. COMMAND should be defined as an alternative name (using the ALTNAME parameter), since many existing IMODs may refer to ADDRESS SYSVIEWE by its previous name.
2. These ADDRESS names are placeholders only. No ADDRESS environment of this name is available to REXX processing.

ALTNAME

Use the ALTNAME parameter to establish multiple names for the same ADDRESS environment. This is especially useful when different sets of existing IMODs each rely on a different name.

Syntax

Use this format for the ALTNAME parameter:

```
ALTNAME altname ADDRESS name
```

Arguments

The ALTNAME parameter takes these arguments.

altname

1-to 8-character alternative name for this ADDRESS environment.

name

ADDRESS environment's name.

AUTOMATION

Use the AUTOMATION parameter to provide values for the CA OPS/MVS Event Management and Automation initialization IMOD.

Syntax

Use this format for the AUTOMATION parameter:

```
AUTOMATION text
```

Arguments

The AUTOMATION parameter takes this argument:

text

Information that is queued to a stack and passed to the initialization IMOD supplied by CA OPS/MVS Event Management and Automation (if one is supplied). See your CA OPS/MVS Event Management and Automation documentation for parameter requirements.

CMD

Use the CMD parameter to define your own CA-GSS operator commands (in addition to the CA-GSS operator commands provided by CA). These commands are processed by IMODs and the results reported by the various CA-GSS operator command platforms.

Syntax

Use this format for the CMD parameter:

```
CMD cmd imod [argstring]
```

Arguments

The CMD parameter takes these arguments:

cmd

Word to be recognized as an operator console command. Each command consists of an uppercase prefix and an optional lowercase suffix. The length of the uppercase portion is noted and used as the minimum required length. The lowercase portion is used to extend the required match to the actual length of the operator-entered string.

Note: The actual comparison is not case sensitive.

imod

Name of the IMOD that will be triggered to process the command.

argstring

Optional argument string to be passed to the IMOD with the arguments entered by the operator. This string is passed to the IMOD concatenated in front of any operator-specified arguments.

Usage Notes

Do not confuse the CMD parameter (that creates a command recognized when issued to CA-GSS) with the COMMAND parameter (creates a new z/OS console operator command).

COMMAND

Use COMMAND to create new z/OS console operator commands.

Syntax

Use this format for the COMMAND parameter:

```
COMMAND VERB cmd {IMOD|PREFIX} name [USERID userid]
```

Arguments

The COMMAND parameter takes these arguments.

Argument	Description
cmd	1- to 16-character word to be recognized as an operator console command. Avoid non-alphabetic characters. Do not duplicate an existing operator command name.
IMOD	Indicates that *name* is a complete IMOD name.
PREFIX	Indicates that an IMOD name is to be constructed from *name*, followed by the first blank-delimited word that follows *cmd* in the operator-entered string.
name	Either the IMOD's name or a prefix used to derive it.
userid	User ID that will be assigned to the IMOD task created to process this command, if entered from a *true* console. Commands entered from other programs or TSO address spaces are always assigned the user ID of the requestor. Default: The command executes under the default user ID.

Usage Notes

- CA-GSS lets you define one or more words that will be recognized as commands when entered by a console operator. Whenever such a command is entered, CA-GSS intercepts it and executes the specified IMOD. Results can be displayed on the issuing console.
- CA-GSS operating commands can be entered in this manner if you define a command word for that purpose. Since CA-GSS also supports the OS MODIFY (F) command, specification of a command word is optional.
- Do not confuse the COMMAND parameter (which creates a new z/OS console operator command) with the CMD parameter (which creates a command that is recognized when issued to CA-GSS).
- If you are running multiple CA-GSS address spaces, make sure the command words are unique for each address space. Otherwise, more than one address space will execute the command.

DB2PLAN

Use the DB2PLAN parameter to provide the name of the DB2 plan for executing dynamic SQL using the DB2() function.

Syntax

Use this format for the DB2PLAN parameter:

```
DB2PLAN plan
```

Arguments

The DB2PLAN parameter takes this argument:

plan

1- to 8-character name of the DB2 plan. The default is GSSPLAN.

DUMP

Use the DUMP parameter to route CA-GSS-generated diagnostic dumps to a specific user at a specific node or to use a particular SYSOUT class for the dump.

Syntax

Use this format for the DUMP parameter:

```
DUMP SPIN TO user [AT node] [Class class]
```

Arguments

The DUMP parameter takes these arguments:

Argument	Description
user	User to whom you are routing the diagnostic dump.
node	Node at which the user is located.
class	SYSOUT class that CA-GSS should use for the dump.

Usage Notes

Keep this information in mind:

- By default, diagnostic dumps are routed to the default location for the started task, using the default SYSOUT class.
- Diagnostic dumps are produced by the REXX interpreter when an unexpected program check or abend occurs.

EDITOR

Use the EDITOR parameter to define the default data sets that the IMOD editor and the CA-GSS/ISERVE Control Panel should use.

Syntax

Use this format for the EDITOR parameter:

```
EDITOR [[PLIB|MLIB|LLIB|CLIB|PARMLIB] dsn]
       [MEMBER member]
       [VIO vioname]
```

Arguments

The EDITOR parameter takes these arguments.

Argument	Description
PLIB	Identifies the data set as your ISPF panel library.
MLIB	Identifies the data set as your ISPF message library.
LLIB	Identifies the data set as your ISPF load library.
CLIB	Identifies the data set as your CLIST library (SYSPROC).
PARMLIB	Identifies the data set as your parameter library.
dsn	Data set's dsname.
member	Default member of the PARMLIB data set that contains the IMOD editor's parameters and ISET menu.
vioname	UNIT name associated with VIO in your organization. (The IMOD compiler uses this name.)

Usage Notes

- If you define default data sets through the EDITOR parameter, you do not need to pre-allocate them to individual TSO users. You can also centrally control program versions and optional editor parameters.
- Only the primary CA-GSS address space provides the editor's data sets to the IMOD editor. Secondary CA-GSS address spaces store the specified values internally. These values are retrieved when the editor's LINK command is being processed.

  EDITOR values are used only if both of the following conditions are met:

  CA-GSS is initialized. (You can initialize it by executing GSSLOAD or by starting GSSMAIN. Once initialized, CA-GSS is available until the next IPL.)

  - TSO users have not pre-allocated the IMOD editor's data sets.

EVENTS

Use the EVENTS parameter to choose the size of the EVENTS table.

Syntax

Use this format for the EVENTS parameter:

```
EVENTS n
```

Arguments

The EVENTS parameter takes this argument:

n

Number of slots to be available in the EVENTS table. Specify a value from 500 to 32767 (inclusive). The default is 1000.

Usage Notes

- The CA-GSS dispatcher is constructed around the z/OS EVENTS facility. The EVENTS table must allow a slot for each pending (unprocessed) event. Depending on dispatching priorities and selected options, the EVENTS table will need room to buffer deferred work.
- An S502 abend usually indicates an EVENTS table that is too small.

FUNCTION

Use the FUNCTION parameter to identify functions that are not included in the CA-GSS load module.

Syntax

Use this format for the FUNCTION parameter:

```
FUNCTION name module [seconds] {SYNC|ASYNC|TEST}
```

Arguments

The FUNCTION parameter takes these arguments:

Argument	Description
Name	1- to 8-character function name. This name should not duplicate the name of any CA-provided functions.
Module	Name of the load module to process the function call.

Argument	Description
Seconds	Maximum time, in seconds, that a single function call can consume. This parameter has effect only when the function is executed as a subtask (ASYNC or TEST).
SYNC	Indicates that function processing contains no implied or explicit WAITs (no I/O). The load module must be re-entrant and must not exit without restoring the original environment (release storage, and so on).
ASYNC	Indicates that function processing should be performed under a true subtask. I/O and WAITs are permissible.
TEST	Same as ASYNC, except that the subtask is detached at the conclusion of the IMOD task responsible for creating it. This ensures that a fresh copy of the non-re-entrant load module is fetched for each new IMOD task that calls the function call.

GLOBVAL

Use the GLOBVAL parameter to provide initial values for global variables.

Syntax

Use this format for the GLOBVAL parameter:

```
GLOBVAL &var {d}value{d} [SIZE size]
```

Arguments

The GLOBVAL parameter takes these arguments:

Argument	Description
&*var*	Name of the variable to be initialized. It must begin with an ampersand. Global variable names are case-sensitive.
D	Arbitrary delimiter character. You must enclose the value to be assigned within delimiters of your choice. The delimiter can be any character, except blank, that is not contained in *value*.
Value	Value to be assigned to the global variable. This value is concatenated to the global variable's current value, if any. You can code the GLOBVAL parameter multiple times for the same global variable in order to assign longer strings than can be accommodated on a single parameter line.

Argument	Description
Size	Global variables are normally limited to 80 characters of storage. You can override this value by including a value for the *size* argument. Specify a value from 80 to 32000. Once allocated, a global variable's maximum size is fixed. If you are coding multiple GLOBVAL statements for the same variable (to assign a long value) only the first statement needs a size specification.

Usage Notes

Global variables begin with an ampersand (&) and are common to all IMOD tasks. They differ from REXX variables in that an uninitialized global variable has a null value (as opposed to its name).

GOALNET

Use the GOALNET parameter, specified several times, to define your GoalNet configuration to CA-GSS.

Syntax

Use this format for the GOALNET parameter.

```
GOALNET node LUNAME luname[,pswd] MODE mode connect

[SESS {total[,win[,lose]}]
      {SINGLE}]
```

Arguments

The arguments that follow the GOALNET verb are positional and must be the order specified.

The following table lists the arguments and their descriptions.

Argument	Description
Node	An arbitrary but unique name to be assigned to the node. This name will be used by IMODs to specify destinations. Node names can be from 1 - 16 alphanumeric characters in length.
Luname	The VTAM luname that is the application ID used to identify the ACB to communicate with VTAM. Also used by other GoalNet nodes to establish communication with this node.

Argument	Description
Pswd	The VTAM application password that VTAM requires before an ACB can be opened. Only code a value if VTAM requires it and this node is to be the local node (specified in the GOALNETLOCAL parameter). If no password is required, omit the field.
Mode	The VTAM logmode entry that is used for communications targeted to the node being defined. The logmode entry must be defined in the VTAM logmode table associated with the VTAM APPLID definition. The default is GOALNET.
connect	Controls how this node will connect to the network. The following options are defined: **ATINIT** The node is connected when ISERVE is initialized. **JOIN** The node is connected when the GOALNET JOIN operator command is entered. **ONCMD** The node is connected when the GOALNET START *node* operator command is issued, where *node* is the name of an individual node.
SESS	Total number of sessions that can be requested with the GoalNet partner. The following options are defined: **total,win,lose** *total* is the total number of sessions permitted with the node. By default, this value is 50 for a TYPE value of GSS and 2 for all other types. Coding SINGLE for the value marks the node as single-session capable. *win* is the total number of contention-winner sessions permitted with the node. By default, this value is half the value of *total* rounded down. *lose* is the total number of contention-loser sessions permitted with the node. By default, it is the value of *total* less the value of *win*. **SINGLE** Indicates that the partner node is single-session capable; assumes a value of 1 for *total*.

GOALNETLOCAL

Use the GOALNETLOCAL parameter to identify the GoalNet node that is assigned to this CA-GSS address space.

Syntax

Use this format for the GOALNETLOCAL parameter:

```
GOALNETLOCAL name
```

Arguments

The GOALNETLOCAL parameter takes this argument:

name

1- to 16-character GoalNet node name for this CA-GSS system. This name must be defined by a GOALNET parameter.

HELPDSN

Use the HELPDSN parameter to provide the dsname of the ISPF panel library that contains ISPF help panels for CA-GSS.

Syntax

Use this format for the HELPDSN parameter:

```
HELPDSN    dsname
```

Arguments

The HELPDSN parameter takes this argument:

dsname

Your CA-GSS PLIB data set name. This data set name should be the same as the one that is specified on the EDITOR PLIB statement.

Usage Notes

These help panels can be used by users who access CA-GSS through the CA SYSVIEW Performance Management ISERVE command or through the CA-GSS Logon Facility Control Panel.

ILOG

Use the ILOG parameter to define one or more ILOG files. These can be specified in any order, but there must be one ILOG statement per ILOG file.

Syntax

Use this format for the ILOG parameter:

```
ILOG FILE file SUBFILE subfile DSN dsname [PAGES pages]

  [[SYSNAME sysname] [SSNAME ssid] [SMFID smfid]
```

Arguments

The ILOG parameter takes these arguments.

Argument	Description
FILE	Indicates that an ILOG file definition follows. Specify one FILE occurrence for each ILOG subfile required.
File	File number to identify the collection of subfiles that comprise the ILOG. Specify a value from 0 to 99 (inclusive).
Subfile	Subfile number, unique within the ILOG. CA-GSS will rotate among all subfiles within an ILOG, filling each in its turn. Specify a value from 0 to 9 (inclusive).
Dsname	Name of a VSAM linear data set, for the exclusive use of this CA-GSS address space, that will be used to contain the ILOG subfile. This file must be large enough to contain the entire ILOG subfile. Common errors include specification of a non-linear data set and invalid SHAREOPTIONS.
Pages	Number of 4K pages to be allocated to the subfile. This value must be 5 or greater. Default: 5
Sysname	If this parameter is present, its value will be compared with the z/OS system name and, if it does not match, the ILOG parameter will be ignored.
Ssid	If this parameter is present, its value will be compared with the value already specified by the SSNAME initialization parameter and, if it does not match, the ILOG parameter will be ignored.
Smfid	If this parameter is present, its value will be compared with the z/OS SMF ID and, if it does not match, the ILOG parameter will be ignored.

INCLUDE

Use the INCLUDE parameter to process additional PARMLIB members.

Syntax

Use this format for the INCLUDE parameter:

```
INCLUDE member
```

Arguments

The INCLUDE parameter takes this argument:

member

Name of the member to be included in the initialization parameter stream.

Usage Notes

The specified member from the PARMLIB DD statement is included in the initialization parameter stream. Following end-of-file in the included member, processing resumes with the record that follows the INCLUDE.

INSIGHT

Use the INSIGHT parameter to provide values for the CA Insight Database Performance Monitor for DB2 for z/OS initialization IMOD.

Syntax

Use this format for the INSIGHT parameter:

```
INSIGHT text
```

Arguments

The INSIGHT parameter takes this argument:

text

Values that you want CA-GSS to pass to the initialization IMOD that is supplied by CA Insight Database Performance Monitor for DB2 for z/OS.

Usage Notes

The parameter fields that follow INSIGHT will be queued to a STACK and passed to the initialization IMOD that is supplied by CA Insight Database Performance Monitor for DB2 for z/OS, if the IMOD exists. See your CA Insight Database Performance Monitor for DB2 for z/OS documentation for parameter requirements.

ISET

Use the ISET parameter to define an ISET for loading or reloading IMODs.

Syntax

Use this format for the ISET parameter:

```
ISET name [DSN dsname] [loadoption|accessoption] [DESC /desc/]
[SSID ssid]
```

Arguments

The ISET parameter takes these arguments.

Argument	Description
Name	1- to 16-character name for the ISET. If the ISET is to be available for reloading (from TSO), this ISET name must match the ISET name as defined to the TSO user. Throughout an installation, each ISET name should point to the same data set, although each data set can be referred to by multiple ISET names.
Dsname	Name of a VSAM KSDS that contains the ISET. When a reload request for an IMOD is processed by CA-GSS, both the ISET name and dsname, as defined to both CA-GSS and the requestor, must be an exact match.
loadoption	Indicates whether an IMOD should be loaded. Specify one of these options: **FORCE** Loads an IMOD and, if another IMOD is currently loaded, replaces it with the new IMOD. **LOAD** Loads an IMOD; however, does not replace any currently loaded IMOD. **NOFORCE** Negates a FORCE specification. **NOLOAD** Negates a LOAD specification.
accessoption	Default access level for the ISET when accessed from the IMOD editor. Specify READ (read-only) or WRITE (read/write).
Desc	1- to 52-character description of the ISET's contents. The IMOD editor displays this description. Because a description can contain blanks, the first non-blank character that follows the DESC parameter is assumed to be the delimiting character.
Ssid	Default subsystem ID that the IMOD editor should use for dynamic IMOD loading and execution.

Usage Notes

- Only valid IMODs can be loaded into CA-GSS. A valid IMOD (for an ISERVE address space) is one that compiled without errors and is in production or force status. In batch mode, a valid IMOD is one that compiled without errors and is in test, production, or force status.
- If loading of the IMOD would cause a previously loaded IMOD of the same name to be replaced, the operation is suppressed for an IMOD in production status (or test status, if batch).
- To force the load of an IMOD from the internal (system) ISET, you must either code FORCE and force-load the entire ISET, or change the status of one or more IMODs to force.
- If neither LOAD nor FORCE is coded, the ISET is merely defined and can be used later for dynamic IMOD loading.
- Each ISET that is to be accessible to CA-GSS, for the purpose of loading or reloading IMODs, must be defined by an ISET statement. These statements are processed in the same order that they are encountered and can merely define an ISET or cause the contained IMODs to be loaded into CA-GSS.
- The contents of the INTERNAL ISET, as shipped by CA, have been link-edited into the CA-GSS load module. Hence, this ISET should be defined only if there is a need to modify or reload an IMOD.
- In the case of duplicate IMODs, the first occurrence is retained unless the FORCE attribute is specified for the IMOD.
- During the course of loading an IMOD, if the compiler version does not match the current interpreter version, the IMOD will automatically be re-compiled. This process affects only the execution copy, and the updated version is not written to the ISET. To reduce start-up overhead, you should recompile all IMODs in an ISET under the current compiler version. The SRVMAINT utility's UPGRADE option provides this capability.
- If duplicate ISET statements are encountered (same ISET name), the arguments are applied to the existing definition, either by adding new information or overriding an option.
- The READ, WRITE, DESC, and SSID parameters are not used by the CA-GSS address space - they're recorded and furnished to the ISPF-based IMOD editor during the execution of the editor's LINK command.

JESNODE

Use the JESNODE parameter to specify the name of the JES NJE node where this copy of CA-GSS is executing.

Syntax

Use this format for the JESNODE parameter:

```
JESNODE node
```

Arguments

The JESNODE parameter takes this argument:

node

Name of the JES NJE node.

Usage Notes

This parameter is required because JES provides no reliable method of obtaining the information directly. This information can then be made available to IMODs at other GoalNet nodes.

This parameter is required if you are running CA Jobtrac Job Management and you want to support a multi-CPU environment.

JOBTRAC

Use the JOBTRAC parameter to provide values for the CA Jobtrac Job Management initialization IMOD.

Syntax

Use this format for the JOBTRAC parameter:

```
JOBTRAC text
```

The JOBTRAC parameter takes this argument:

text

Information that is queued to a stack and passed to the initialization IMOD supplied by CA Jobtrac Job Management (if one is supplied). See your CA Jobtrac Job Management documentation for parameter requirements.

LOGLINES

Use the LOGLINES parameter to control the maximum number of lines written to the ISRVLOG or GNETLOG data sets before they are automatically closed, spun, and a new SYSOUT data set is allocated.

Syntax

Use this format for the LOGLINES parameter:

```
LOGLINES lines
```

Arguments

The LOGLINES parameter takes this argument:

lines

Maximum number of lines for the ISRVLOG or GNETLOG data set. Specify a value from 1000 to 2147483647.

Usage Notes

- When the new log file is allocated, its ddname will be system-generated and will be of the form SYS*nnnnn*.
- In your JCL, you can include an ISRVOUTP OUTPUT statement to indicate the desired SYSOUT class, routing, and so on, for the newly allocated log file.
- The log file can be manually closed and spun by issuing the SPIN ISRVLOG or SPIN GNETLOG command.

LOGON

Use the LOGON parameter to define and activate the CA-GSS Logon Facility.

Syntax

```
LOGON {LUNAME luname [password]}
      {APPLication applname imod [arg_string]}
```

Arguments

The LOGON parameter takes these arguments:

Argument	Description
LUNAME	Specifies definition of the LU name to be used by the Logon Facility. Only one LUNAME specification is permitted.

Argument	Description
APPLication	Specifies definition of a Logon Facility application. This parameter can be repeated as many times as required to complete the specification of all applications. This parameter can be abbreviated as APPL.
Luname	VTAM LU name to be used by the Logon Facility.
password	VTAM password required to use this LU name. If not needed, *password* should be omitted.
applname	Application name that will be specified by a user at logon time. This name must be all uppercase or can consist of a required uppercase string followed by an optional lowercase string. At logon, the user must provide all of the uppercase string and can include any amount of the lowercase string. For example, specifying APPLname would permit the user to enter APPL, APPLN, APPLNA, APPLNAM, or APPLNAME.
Imod	Name of the IMOD that is to support the session for the specified application. The IMOD must have been written with Logon Facility capability.
arg_string	Optional argument string to be passed to the IMOD at the start of the session.

MAXSTACK

Use the MAXSTACK parameter to set an upper limit for the amount of storage that can be used to contain a single stack.

Syntax

Use this format for the MAXSTACK parameter:

```
MAXSTACK nnnnn
```

Arguments

The MAXSTACK parameter takes this argument:

nnnnn

Maximum stack size in K-bytes (1024 bytes). If specified, each IMOD task will be limited to stacks with a maximum size of *nnnnn*. The default is 1,000K (1 megabyte).

Note: This value can be overridden by the IMOD.

MAXXREQ

Use the MAXXREQ parameter to set the maximum number of concurrent, cross-memory IMOD requests.

Syntax

Use this format for the MAXXREQ parameter:

MAXXREQ *nn*

Arguments

The MAXXREQ parameter takes this argument:

nn

Maximum number of cross-memory IMOD requests that can be in-flight at a given time. Specify a value from 5 to 10,000. The default is 500.

Cross-memory IMOD requests consist of all requests to execute IMODs that originate externally to CA-GSS but from the same z/OS image. Lowering this limit can result in the rejection of requests made by other CA products.

PRIMARY

Use the PRIMARY parameter to indicate whether this CA-GSS address space is the primary CA-GSS address space.

Syntax

Use this format for the PRIMARY parameter:

```
PRIMARY {YES|NO}
```

Arguments

The PRIMARY parameter takes these arguments:

Argument	Description
YES	Identifies this address space as the primary one for CA-GSS on this z/OS image. If another CA-GSS address space is currently the primary one, that address space's status is changed to secondary.
NO	Identifies this address space as the secondary one for CA-GSS on this z/OS image.

Usage Notes

- Each z/OS image can have exactly one primary CA-GSS address space, but it can have many secondary CA-GSS address spaces.
- Service requests that are not routed to a particular address space (by subsystem ID) are automatically routed to the primary one.
- By default, the address space that executes the GSSMAIN program is considered primary, while address spaces that execute the SRVSYS program are considered secondary.
- The GSSMAIN program reserves a system linkage index (LX) for cross-memory communication. When GSSMAIN terminates, the system marks the address space as non-reusable. Therefore, if you need to cycle CA-GSS repeatedly (for example, during initial installation and testing), you may want to execute the GSSMAIN program as a secondary CA-GSS address space. This allows you to cycle your primary CA-GSS address space without destroying address spaces.

PRODUCT

Use the PRODUCT parameter to request special ISERVE support for CA Insight Database Performance Monitor for DB2 for z/OS or CA Deliver products.

Syntax

Use this format for the PRODUCT parameter:

```
PRODUCT {DBDELIVERY|INSIGHT}
```

Arguments

The PRODUCT parameter takes these arguments:

Argument	Description
DBDELIVERY	Requests special ISERVE support for CA Deliver.
INSIGHT	Requests special ISERVE support for CA Insight Database Performance Monitor for DB2 for z/OS.

RERUN

Use the RERUN parameter to provide values for the CA 11 Workload Automation Restart and Tracking initialization IMOD.

Syntax

Use this format for the RERUN parameter:

```
RERUN text
```

Arguments

The RERUN parameter takes this argument.

text

Information that is queued to a stack and passed to the initialization IMOD supplied by CA 11 Workload Automation Restart and Tracking (if one is supplied). See your CA 11 Workload Automation Restart and Tracking documentation for parameter requirements.

SCHEDULE

Use the SCHEDULE parameter to execute an IMOD immediately after the system initializes.

Syntax

Use this format for the SCHEDULE parameter:

```
SCHEDULE IMOD name [arguments]
```

Arguments

The SCHEDULE parameter takes these arguments:

Argument	Description
Name	Name of the IMOD to be executed.
arguments	Optional string that is passed to the IMOD as an argument.

Usage Notes

- You can specify the SCHEDULE parameter as many times as needed.
- IMODs that are executed as the result of the SCHEDULE parameter are spawned as individual tasks, and they execute concurrently.

SECURITY

Use the SECURITY parameter to specify how security is implemented in your installation and to assign the CA-GSS default user ID.

Syntax

Use this format for the SECURITY parameter:

```
SECURITY {NONE|SAF} [DEFAULT userid]
```

Arguments

The SECURITY parameter takes these arguments.

Argument	Description
NONE	No attempt will be made to limit processing by user ID. All IMOD tasks and subtasks will execute under the authority of the CA-GSS address space. CA-GSS will still attempt to obtain a user ID for each requestor and that user ID will be passed to IMODs and ADDRESS environments that request the information.
SAF	This option should be coded if a SAF-compatible security system is installed. CA-GSS' CALLs to SAF and its use of ACEEs are compatible with operating-system requirements. All security software should support the needed functions.
Userid	There are times when a user ID cannot be assigned to a request. For example, some started tasks do not have user IDs. When no user ID can be determined, this default value will be used. **Note:** This default must be recognized as a valid user ID by your security system.

Usage Notes

- Whenever a request for executing an IMOD is received, an attempt is made to determine the user ID responsible for the request.

  If the user ID cannot be determined, execution is performed under the default user ID, if one has been specified and it is valid. Otherwise, processing occurs under the authority of the CA-GSS address space.
- SECURITY is syntax checked and then ignored by SRVBATCH, SRVMAINT, and the IMOD Packaging Facility. These programs always execute under the security assigned by the system at job initiation.
- If you do not specify the SECURITY parameter, no attempts will be made to access your site's security system.

SLICE

Use the SLICE parameter to specify the maximum number of consecutive REXX clauses that can be executed before control is passed to other waiting IMOD tasks.

Syntax

Use this format for the SLICE parameter:

SLICE *n*

Arguments

The SLICE parameter takes this argument.

- Maximum number of REXX clauses that an IMOD task can execute before it is forced to relinquish control to other ready IMOD tasks. Specify a value from 100 to 2147483647 (inclusive). The default is 10,000.

Usage Notes

- Since multiple IMOD tasks can execute concurrently in the CA-GSS address space, the dispatcher provides mechanisms to share control of the processor. Any request for I/O, for example, will cause control to pass from the I/O requestor to any IMOD task that is waiting and ready to resume execution.
- If an IMOD task does not relinquish control due to I/O or other asynchronous requests, the SLICE parameter provides a method for time slicing between multiple IMOD tasks.

SSID

Use the SSID parameter to indicate what subsystem ID is being used by CA OPS/MVS Event Management and Automation, DB2, or CA SYSVIEW Performance Management.

Syntax

Use this format for the SSID parameter:

```
SSID {AUTOMATION|DB2|PCOMMAND} ssname
```

Arguments

The SSID parameter takes these arguments:

Argument	Description
AUTOMATION	Indicates the subsystem ID is for CA OPS/MVS Event Management and Automation.
DB2	Indicates that the subsystem name is for DB2.
PCOMMAND	Indicates that the subsystem name is for CA SYSVIEW Performance Management.
Ssname	Product's subsystem ID.

SSNAME

Use the SSNAME parameter to specify the subsystem ID that will be used to identify this particular ISERVE address space.

Syntax

Use this format for the SSNAME parameter:

```
SSNAME ssid
```

Arguments

The SSNAME parameter takes this argument:

ssid

Subsystem ID that is unique on this operating system.

Usage Notes

- At initialization, if the specified subsystem ID has not been defined to z/OS, the definition will automatically be made.
- This parameter is required.
- This parameter must precede any ILOG parameters that specify a subsystem ID.

STORAGE

Use the STORAGE parameter to force CA-GSS work areas into 24-bit addressable storage.

Syntax

Use this format for the STORAGE parameter:

```
STORAGE {HIGH|LOW}
```

Arguments

The STORAGE parameter takes these arguments:

Argument	Description
HIGH	The default; most storage allocated to CA-GSS resides above the 16-megabyte line.
LOW	Moves certain work areas into 24-bit addressable storage; required if you are running CA-GSS under z/OS/XA.

Usage Notes

CA-GSS runs RMODE ANY and allocates all possible storage from high memory. Due to the constraints of early versions of z/OS and other software, it can be necessary to cause internal work areas to be allocated from low memory. This is especially true if you are experiencing 0C4 abends while executing REXX functions that interface with the operating system (WTO, OSCMD, and so on).

SYSVIEWE

Use the SYSVIEWE parameter to provide values for the CA SYSVIEW Performance Management initialization IMOD.

Syntax

Use this format for the SYSVIEWE parameter:

```
SYSVIEWE text
```

Argument

The SYSVIEWE parameter takes this argument:

text

Information that is queued to a stack and passed to the initialization IMOD supplied by CA SYSVIEW Performance Management (if one is supplied). See your CA SYSVIEW Performance Management documentation for parameter requirements.

TCP/IP

Use the TCP/IP parameter to enable GMF to use the TCP/IP protocol for communications.

Syntax

Use this format for the TCP/IP parameter:

```
TCP/IP [JOBNAME name]
       [PORT port]
```

Arguments

The TCP/IP parameter takes these arguments:

Argument	Description
Name	Name of the TCP/IP address space on this z/OS system.
Port	TCP/IP port that CA-GSS should use.

Usage Notes

- GMF can use either SNA (LU 6.2) or TCP/IP protocols in order to communicate with certain non-z/OS platforms.
- The port address must be made known to all other GMF partners.

TRACE

Use the TRACE parameter to control what trace output is sent to the ISERVE log, ISRVLOG.

Syntax

Use this format for the TRACE parameter:

```
TRACE {ALTER}    {ON}
      {DEBUG}    {OFF}
      {FAILED}
      {LOAD}
      {RESULT}
      {SCHED}
      {SPACE}
      {SUBCALL}
```

Arguments

The TRACE parameter takes these arguments.

Argument	Description
ALTER	Lists storage alterations performed by the MEMORY() function, regardless of an explicit NOLOG option in the function parameter list.
DEBUG	Lists miscellaneous debugging information. This option should only be activated at the request of CA Technologies Support.
FAILED	Lists all IMODs that fail during scheduling.
LOAD	Lists each IMOD as it is loaded or re-loaded into memory.
RESULT	Lists the contents of the result variable, as returned by each IMOD.
SCHED	Lists each IMOD as it is executed.
SPACE	Lists memory usage statistics for each IMOD as it is executed.
SUBCALL	Lists calls to external subroutines from within an IMOD task.
ON/OFF	Determines whether tracing is active or inactive.

Usage Notes

- The TRACE parameter was named PRINT in previous releases of CA-GSS.
- By default, tracing is inactive.

USER

Use the USER parameter to provide values for the user-provided INITIALIZATION IMOD.

Syntax

Use this format for the USER parameter:

```
USER text
```

Arguments

The USER parameter takes this argument:

text

Information that is queued to a stack and passed to the installation-supplied INITIALIZATION IMOD (if one exists).

VIEW

Use the VIEW parameter to provide values for the CA View initialization IMOD.

Syntax

Use this format for the VIEW parameter:

```
VIEW text
```

Arguments

The VIEW parameter takes this argument:

text

Information that is queued to a stack and passed to the initialization IMOD supplied by CA View (if one is supplied). See your CA View documentation for parameter requirements.

VIOUNIT

Use the VIOUNIT parameter to specify the unit name for VIO requests.

Syntax

Use this format for the VIOUNIT parameter:

```
VIOUNIT unitname
```

Arguments

The VIOUNIT parameter takes this argument:

unitname

The z/OS unit name used for VIO allocation; the default value of *unitname* is VIO.

Usage Notes

Certain work files can be required by CA-GSS from time to time. This is true especially during system initialization when back-leveled IMODs are being recompiled. If you have redefined your VIO unit name from the IBM-supplied VIO, you will need to specify the replacement value with the VIOUNIT parameter.

VTAMRESTART

Use the VTAMRESTART parameter to specify how VTAM communications should be handled if VTAM terminates and then restarts.

Syntax

Use this format for the VTAMRESTART parameter:

```
VTAMRESTART {NONE|AUTO|WTOR}
```

Arguments

The VTAMRESTART parameter takes these arguments:

Argument	Description
NONE	No restart will be performed. Once VTAM has been restarted, you must manually restart GoalNet. This is the default.
AUTO	GoalNet will automatically restart when the VTAM message, IST020, is detected.

Argument	Description
WTOR	At the time of failure, GoalNet will issue a WTOR. The operator can respond with RESTART, NONE, or AUTO.

WTO

Use the WTO parameter to screen and select WTOs, based on the message ID or fragment. CA-GSS will then log the WTO to a specified ILOG file or trigger an IMOD task. If an IMOD is to be executed, the text of the WTO is passed to the IMOD as the argument string.

Syntax

Use this format for the WTO parameter:

```
WTO ID msgid [IMOD imodname] [ILOG ilognum] [USERID userid]
```

Arguments

The WTO parameter takes these arguments:

Argument	Description
Msgid	1- to 16-character message ID or beginning fragment of message to be selected. Scan begins with left-most character of message and must match each specified character. If no mismatch has occurred when pattern is exhausted, message is selected. If the first character of the source message ID is *not* in the range of A to Z, it is skipped, and comparison starts on the next character regardless of its value.
imodname	IMOD name to be executed each time specified message is found.
ilognum	Number of the ILOG file where the message is to be recorded.
Userid	User ID assigned to IMOD task created to process the WTO text. If omitted, IMOD task executes under the authority of the task that issued the WTO, if it can be determined. If user ID cannot be determined, IMOD task executes under the default user ID, as specified with the SECURITY parameter.

Usage Notes

- All logging and IMOD-based processing is performed on a delayed basis and not as part of system WTO processing.
- ILOG recording is performed after all IMOD processing associated with the message.
- If both ILOG and IMOD are specified for the same message, the ILOG number can be changed by the IMOD.
- If an IMOD executed as the result of a WTO spawns another IMOD, a copy of the original message is passed. However, the spawned message will have no default ILOG routing associated with it. The spawned IMOD can establish its own ILOG routing (using the IROUTE() function), but cannot affect the recording of the original message text. Therefore, if you are not careful, duplicate recording of the same message is possible.
- When a message ID matches more than one WTO specification, the last-matched occurrence is the one whose IMOD is executed or whose ILOG routine code is assigned. Therefore, when coding both specific and generic WTO IDs, enter the generic IDs first, followed by the specific IDs.

CA-GSS Programs

The following table shows the programs that are supplied with CA-GSS:

Program	Description
GOALNETT	VTAM LOGMODE table.
GSSLOAD	Controls CA-GSS background tasks.
GSSMAIN	Executes the CA-GSS address space and the primary ISERVE.
GSSRC	Convert CA-GSS error codes to text.
SRVALERT	External function that allows IMODs to generate IBM Tivoli NetView alerts.
SRVBASE	Required for packaging IMODs as load modules.
SRVBATCH	Allows you to execute single IMODs in a batch environment.
SRVCALL	TSO/E REXX function to invoke an IMOD running in CA-GSS address space.
SRVCCAT	Allows dynamic concatenation of data sets.
SRVCOMP	Compiler module; an internal program that cannot be directly invoked by the user.
SRVDB2P	DB2 plan for dynamic SQL execution.

Program	Description
SRVDCAT	Allows dynamic deconcatenation of data sets.
SRVEDIT	Provides the terminal user with functions needed to create and maintain IMODs.
SRVIMOD	Lets you invoke IMODs from your assembler-language programs.
SRVMAINT	Allows you to create and maintain IMOD files (ISETs).
SRVMAPS	Provides data-area maps for use with CA SYSVIEW Performance Management.
SRVOPER	Invokes the ISERVE Operator Facility.
SRVOPS	Specialized version of SRVCALL executing under CA OPS/MVS Event Management and Automation.
SRVSYS	Executes secondary ISERVEs.
SRVTSO1	Executes an IMOD and invokes ISPF/PDF editor to display stack data.
SRVTSO2	Executes an IMOD and invokes the ISPF/PDF browse facility to display the stack data.
SRVTSO3	Executes an IMOD and returns the stack data in a data set.

GOALNETT

GOALNETT provides the VTAM LOGMODEs necessary for the operation of GoalNet. This load module must be copied into your SYS1.VTAMLIB data set and be referenced by the application IDs that will be used to support GoalNet.

This load module was assembled from the GOALNETT SAMPJCL member using the VTAM version 3, release 4 MACRO library. CA recommends that you reassemble the source code to prevent any mismatch between your version of VTAM and the one used to assemble the provided load module.

GSSLOAD

GSSLOAD controls the CA-GSS functions. Commands are controlled by a single word, specified as a parameter. A sample PROC for executing GSSLOAD is provided in the SAMPJCL member GSSP.

GSSMAIN

GSSMAIN executes the CA-GSS address space and attaches the primary ISERV (SRVSYS). Only one copy of GSSMAIN can execute on a z/OS system at a time.

To prevent the ISERVE from being attached, specify PARM='ISERVE=NO'.

A sample PROC for executing GSSMAIN is provided in the SAMPJCL member GSSA. Various CA products may require additional DD statements. See the documentation for each CA product to determine its precise needs.

GSSRC

GSSRC is dynamically invoked by other CA products to convert CA-GSS internal communication error codes into printable text. This feature is extremely useful in the event of error. For this reason, is it recommended that the GSSRC load module be accessible through a LINKLIST library.

SRVALERT

SRVALERT is a CA-distributed load module that can be used as a REXX function to generate IBM Tivoli NetView alerts.

SRVBASE

SRVBASE contains the code necessary to support IMODs as stand-alone load modules. It is link-edited with the IMOD object code to form an executable package.

SRVBATCH

SRVBATCH allows execution of IMOD tasks in a batch environment. SRVBATCH terminates automatically when the last executing IMOD task is completed.

The OS operator commands MODIFY and STOP may be issued against SRVBATCH during execution. All commands supported by CA-GSS (SRVSYS) are also supported by SRVBATCH.

The following table explains the JCL.

JCL Statement	Function
EXEC...PARM = '[*member*] *imodname*/[*arg*]'	The PARM field on the EXEC card is used to specify the PARMLIB member name, the name of the IMOD to be executed and, optionally, an argument string to be passed to the IMOD. The IMOD name must be the last field or immediately precede the first slash (/) character. To include an argument string, it follows the IMOD name separated by a slash (/). The slash character is discarded and is not passed as part of the argument.
PARMLIB DD	Identifies the source of ISERVE initialization parameters. An optional file, but if provided, all parameters are checked for syntax, including parameters not used in a batch environment, such as WTO, COMMAND, and PRODUCT. If PARMLIB is a PDS, the first word in the EXEC card PARM field should identify the PARMLIB member to be processed.
ISRVLOG DD	Specifies a SYSOUT data set that receives all logged information pertaining to initialization and execution. This file is optional and, unlike SRVSYS, it is not closed and spun.
IMOD DD	This is an optional DD statement that can be used to define an ISET. During initialization, if the IMOD requested using the PARM field is not found in the ISETs already loaded (those defined in PARMLIB), or if no ISETs were defined, then the IMOD DD statement is searched for the desired IMOD.

SRVCALL

SRVCALL is a CA-distributed load module that can be used as a TSO/E REXX function to execute IMODs in the CA-GSS address space. This function will also permit you to execute IMODs from IBM Tivoli NetView REXX procedures and other TSO/E REXX-compatible implementations.

SRVCCAT

SRVCCAT lets you dynamically concatenate two DD statements by using a TSO command. It allows you to add a data set without having to free and re-allocate an entire file concatenation.

Use this format:

SRVCCAT *ddname1 ddname2*

Replace *ddname1* with the ddname of the concatenation to which you want to add another DD statement.

Replace ddname2 with the ddname of the statement you want to add to the concatenation.

If you want to add an SRVPROC DD statement to the current SYSPROC concatenation, you typically would use the FREE SYSPROC command, which would require that you then reallocate all entries. Instead, issue the command SRVCCAT SYSPROC SRVPROC to dynamically concatenate the DD statement.

SRVCOMP

The SRVCOMP program is the compiler module. You cannot directly invoke this program.

The SRVCOMP program is called by the following programs using LINK or LOAD:

SRVSYS

SRVBATCH

SRVMAINT

SRVEDIT

SRVDB2P

Execution of dynamic SQL requires that a skeleton SQL program be compiled and bound to the DB2 address space. This program has DB2 version dependencies and a requirement that there be an exact match of the compiler timestamp between the executed load module and the bound pre-compiled source. For this reason, it is necessary for you to pre-compile and assemble the SRVDB2P source code.

The source for this load module is contained in the SAMPJCL member YS28DB2P.

SRVDCAT

The SRVDCAT program allows you to cancel the concatenation of libraries performed with the SRVCCAT program. After you execute SRVDCAT, the concatenation is broken into the component files with their original ddnames as they existed before concatenation with the SRVCCAT program.

Use this format:

SRVDCAT *ddname*

Replace *ddname* with the ddname of the concatenation created with the SRVCCAT program that you now want to de-concatenate. For example, to cancel the concatenations performed by the SRVCCAT example (as shown in "SRVCCAT"), issue the command *SRVDCAT SYSPROC*.

SRVEDIT

SRVEDIT executes under ISPF and provides the terminal user with the functions needed to create and maintain IMOD libraries (ISETs). The necessary files will be allocated dynamically based on arguments provided using the EDITOR parameter.

The following table lists the files and their functions.

File Name	Function
SRVPARM	Identifies a data set that defines the accessible ISETs.
SRVPLIB	Defines the CA-GSS panel library.
SRVMLIB	Defines the CA-GSS message library.
SRVLLIB	Defines the CA-GSS LINKLIB data set, which is used to obtain a copy of the CA-GSS compiler, SRVCOMP.

Usage Notes

If you pre-allocate these ddnames, CA-GSS will not dynamically allocate them.

SRVIMOD

The SRVIMOD load module and the SRVIMOD MACRO instruction (found in the SAMPJCL data set) may be used from within installation-supplied programs to execute IMODs in an ISERVE address space. Both an argument string and initial stack contents can be supplied. Stack contents, the result string, and binary return and reason codes (set by IMODRC()) are returned. SRVIMOD may be invoked synchronously or asynchronously.

Environment

SRVIMOD has the following characteristics:

- Re-entrant
- AMODE 31
- RMODE ANY

Format

Use this format:

```
[label] SRVIMOD action[,DSECT={NO|YES}][,IMOD=spec1][,NODE=spec1]

 [,SSID=spec1][,REC=spec1][,RECLEN=spec2][,ARG=spec1]

    [,ARGLEN=spec2][,EOF=spec1][,VCON={NO|YES}]
```

The following are valid values for *action*: PLIST, INIT, LOAD, DELETE, EXECONLY, SYNC, ASYNC, RLSE, FINISH, QUEUE, PULL, RESULT, RETCODE, ERROR.

These values and associated parameters are described next.

PLIST

Causes generation of a parameter list, either as a DSECT or as code. A double-word boundary is forced.

Associated parameters are:

DSECT=NO

Maps storage without a DSECT statement.

DSECT=YES

Generates a DSECT statement.

The first generated label is SRVSTART. The length of the parameter list is the value of the symbol SRVLEN. All other references to the parameter list require implicit addressability.

INIT

Causes initialization of the parameter list. This includes both initialization of values and setting of indirect address pointers.

Associated parameters are:

VCON=NO

Suppresses references to the SRVIMOD module. Prior to invocation, you must use the LOAD action to bring the SRVIMOD module into storage.

VCON=YES

Generates a V-CON that link-edits the SRVIMOD module into your load module. Generally not recommended, since moving to a new release of SRVIMOD requires you to relink-edit your program.

LOAD

Issues an OS LOAD request for module SRVIMOD. The address of the module is stored in the parameter list for later reference.

DELETE

Issues an OS DELETE request for module SRVIMOD. The previously loaded module is deleted from storage. This should be done only when all processing that involves SRVIMOD is concluded.

EXECONLY

Causes execution of the specified IMOD. Control is returned to you immediately following the scheduling of the IMOD. No information is returned and the scheduled IMOD may not be executed for a variety of reasons (e.g., IMOD not available).

Associated parameters are:

IMOD=spec1

Specifies the name of the IMOD to be executed. This parameter must be specified.

NODE=spec1

Specifies the GoalNet node where the IMOD is to be executed. To be valid, your primary ISERVE must be a member of GoalNet and must be connected to the specified node.

SSID=spec1

SSID indicates the subsystem ID assigned to the ISERVE where the IMOD is to be executed. If NODE is *not* specified, the ISERVE must be executing under the same z/OS image where your program is running. If NODE *is* specified, the ISERVE must be executing under the same z/OS image as the ISERVE identified by NODE.

ARG=spec1

Specifies an optional character string that will be passed to the IMOD as a REXX argument.

ARGLEN=spec2

Specifies the length of the text specified by ARG=. If ARG specifies literal text, ARGLEN may be omitted.

SYNC

Causes execution of the specified IMOD. Control is returned to you following execution of the IMOD. Additional calls may be made to retrieve the resulting information.

Notes:

See action EXECONLY for a description of related keyword parameters.

- Following retrieval of all pertinent result data, use *action* RLSE to free all internal working storage.

ASYNC

Causes execution of the specified IMOD. Control is returned to you immediately following the scheduling of the IMOD. Following completion of the IMOD, the ECB that you provided is posted. You may then make additional calls to retrieve the resulting information.

Associated parameters are:

ECB=*spec3*

Provides an ECB that will be posted following completion of the IMOD. Once specified, the address will be remembered until replaced. You are responsible for clearing the ECB to zeros prior to each ASYNC call.

Notes:

- See *action* EXECONLY for a description of related keyword parameters.
- Following retrieval of all pertinent result data, use *action* RLSE to free all internal working storage.

RLSE

Releases internal working storage associated with the last request. If you issue a new EXECONLY, SYNC, ASYNC, or FINISH action, a RLSE is automatically generated for the previously executed EXECONLY, SYNC, ASYNC, or FINISH action.

FINISH

Releases all internal working storage associated with the SRVIMOD routine.

QUEUE

Places a record on the STACK that will be passed to the next-executed IMOD. The contents of the input STACK are deleted following IMOD execution.

Associated parameters are:

REC=*spec1*

Specifies the contents of the record to be queued.

RECLEN=*spec2*

Specifies the length of the record to be queued. If REC= specifies a literal string, RECLEN= may be omitted.

PULL

Following execution of an IMOD, PULL returns the address and length of the next stack record.

Reg 0

Address of record

Reg 1

Length of record

Associated parameters are:

EOF=*spec4*

Specifies a branch address to be taken when the stack is exhausted. If not specified, register 15 must be inspected for a non-zero return code.

RESULT

Following execution of an IMOD, RESULT returns the address and length of the returned result, if any.

Reg 0

Address of string

Reg 1

Length of string

RETCODE

Following execution of an IMOD, RETCODE returns the return and reason codes as set by the IMODRC() function.

Reg 0

Reason code

Reg 1

Return code

ERROR

Following the receipt of a non-zero return code in register 15 from the EXECONLY, SYNC, ASYNC, PULL, or RLSE actions, this service returns a text string that describes the error.

Reg 0

Address of string

Reg 1

Length of string

spec1 Value

Specify one of the following for *spec1*:

string

A text string that will be assembled as a literal.

((*reg*))

Any general purpose register, in the range of 2 through 12, that contains the address of the text string.

(A,*addr*)

An RX-addressable symbol that identifies a fullword that contains the address of the required text string.

(,addr*)**

An RX-addressable symbol that identifies a fullword that contains the address of another fullword that contains the address of the required text string.

(V,*addr*)

An RX-addressable symbol that identifies the text string.

spec2 Value

Specify one of the following for *spec2*:

num

A numeric value that will be assembled as a literal.

((*reg*))

Any general purpose register, in the range of 2 through 12, that contains the address of the text string containing the numeric value.

(A,*addr*)

An RX-addressable symbol that identifies a fullword that contains the address of the numeric value.

(*,*addr*)

An RX-addressable symbol that identifies a fullword that contains the address of another fullword that contains the address of the numeric value.

(V,*addr*)

An RX-addressable symbol that identifies the numeric value.

spec3 Value

Specify one of the following for *spec3*:

symbol

The RX-address of the value.

((*reg*))

Any general purpose register, in the range of 2 through 12, that contains the address of the symbol.

(A,*addr*)

An RX-addressable symbol that identifies a fullword that contains the address of the symbol.

(*,*addr*)

An RX-addressable symbol that identifies a fullword that contains the address of another fullword that contains the address of the symbol.

spec4 Value

Specify the following for *spec4*:

symbol

symbol is an RX address.

SRVMAINT

The SRVMAINT program is used to create and maintain IMOD files (ISETs).

The following table explains the JCL.

JCL Statement	Function
SYSIN	Identifies the source of fixed or variable length records containing the commands to be processed by SRVMAINT.
SYSPRINT	Specifies a SYSOUT data set to receive all logged information pertaining to SRVMAINT execution.

Include DD statements for ISET and sequential files referenced by commands, as required.

SRVMAPS

The SRVMAPS load module consists of a set of maps of internal CA-GSS data areas. This module is suitable for use with CA SYSVIEW Performance Management and will be useful to CA Technologies Support if you report problems with CA-GSS.

SRVOPER

The SRVOPER program is executed from within an ISPF session and invokes the CA-GSS Operator Facility. The required data sets are dynamically allocated from the CA-GSS address space.

Files

File Name	Function
SRVPLIB	Defines the CA-GSS panel library.
SRVMLIB	Defines the CA-GSS message library.

SRVOPS

SRVOPS is a CA-distributed load module that can be used as a REXX function to execute IMODs in a CA-GSS address space from a CA OPS/MVS Event Management and Automation address space. This function is equivalent to the SRVCALL function.

SRVSYS

The SRVSYS program allows you to execute secondary ISERVEs. You may run as many secondary ISERVEs as you want. They are useful for testing and charge-back purposes.

The following table explains the JCL. Except for the program name, SRVSYS JCL is identical to GSSMAIN.

JCL Statement	Function
PARMLIB DD	Identifies the source of the initialization parameters that define the secondary ISERVE.
ISRVLOG DD	Specifies a SYSOUT data set that receives all logged information pertaining to ISERVE initialization and execution. Specify FREE=CLOSE on this data set, as it may be spun-off during execution and a new log started.
ISRVOUTP OUTPUT	This is an OUTPUT statement that is referenced when allocating a replacement for ISRVLOG.
GNETLOG DD	Specifies a SYSOUT data set that receives all logged information pertaining to GoalNet initialization and execution. Specify FREE=CLOSE on this data set, as it may be spun-off during execution and a new log started.

SRVTSO1

Executed from within an ISPF session, the SRVTSO1 program executes an IMOD and invokes the ISPF/PDF editor to display the resulting stack data. All input to SRVTSO1 is using the input command string. Parameters are positional and are separated by a period (.). After all parameters have been extracted, all remaining text is passed to the requested IMOD as an argument string. Before executing SRVTSO1, files must be allocated to the TSO session, either by CLIST or by placing DD statements in the logon PROC.

The first table lists the files; the second table lists parameters.

File Name	Function
SRVPLIB	Defines the CA-GSS panel library.
SRVMLIB	Defines the CA-GSS message library.

Parameter Position	Specify	Required
Word 1	The IMOD name.	Yes
Word 2	The GoalNet node where the IMOD is to be executed. The default is the node of the primary ISERVE.	No
Word 3	The subsystem ID that identifies the ISERVE where the IMOD is to execute. If you specify word 2, the subsystem ID is applied at the target node. The default subsystem ID is as follows: If you do *not* specify word 2, the default is the subsystem ID of the primary ISERVE. If you *do* specify word 2, the default is the subsystem ID of the target GoalNet node.	No
Word 4	The name of the panel to be used by the ISPF editor to display the returned stack data. The default is the IBM-provided panel.	No
Word 5	The name of the initial macro to be executed by the ISPF editor. By default, no initial macro is executed.	No
Word 6	The name of the edit profile. The default is IMOD.	No
Remainder	The remaining text is passed to the IMOD as the initial argument string.	No

SRVTSO2

Executed from within an ISPF session, the SRVTSO2 program executes an IMOD and invokes the ISPF/PDF browse facility to display the resulting stack data. All input to SRVTSO2 is using the input command string. Parameters are positional and are separated by a period (.). After all parameters have been extracted, all remaining text is passed to the requested IMOD as an argument string. Before executing SRVTSO2, files must be allocated to the TSO session, either by CLIST or by placing DD statements in the logon PROC.

The first table lists the files; the second table lists parameters.

File Name	Function
SRVPLIB	Defines the CA-GSS panel library.
SRVMLIB	Defines the CA-GSS message library.

Parameter Position	Specify	Required
Word 1	The IMOD name.	Yes
Word 2	The GoalNet node where the IMOD is to be executed. The default is the node of the primary ISERVE.	No
Word 3	The subsystem ID that identifies the ISERVE where the IMOD is to execute. If you specify word 2, the subsystem ID is applied at the target node. The default subsystem ID is as follows: If you do *not* specify word 2, the default is the subsystem ID of the primary ISERVE. If you *do* specify word 2, the default is the subsystem ID of the target GoalNet node.	No
Word 4	The name of the panel to be used by the ISPF editor to display the returned stack data. The default is the IBM-provided panel.	No
Remainder	The remaining text is passed to the IMOD as the initial argument string.	No

SRVTSO3

The SRVTSO3 program executes an IMOD and returns the resulting stack data to a data set. All input to SRVTSO3 is using the input command string. Parameters are positional and are separated by a period (.). After all parameters have been extracted, all remaining text is passed to the requested IMOD as an argument string. Before executing SRVTSO3, files must be allocated to the TSO session, either by CLIST or by placing DD statements in the logon PROC. The first table lists the files and the second table lists the parameters.

File Name	Function
The file specified in arg 4	Receives the current stack records.
SRVMLIB	Defines the CA-GSS message library.

Parameter Position	Specify	Required
Word 1	The IMOD name.	Yes
Word 2	The GoalNet node where the IMOD is to be executed. The default is the node of the primary ISERVE.	No
Word 3	The subsystem ID that identifies the ISERVE where the IMOD is to execute. If you specify word 2, the subsystem ID is applied at the target node. The default subsystem ID is as follows: If you do not specify word 2, the default is the subsystem ID of the primary ISERVE. If you do specify word 2, the default is the subsystem ID of the target GoalNet node.	No
Word 4	The DDname where the stack contents are to be placed. Each stack record translates to one data record in this file.	Yes
Word 5	The RECFM value to be used for the output file. The default is the RECFM of the file.	No
Word 6	The LRECL value to be used for the output file. The default is the LRECL of the file.	No
Word 7	The BLKSIZE value to be used for the output file. The default is the BLKSIZE of the file.	No
Remainder	Passes remaining text to the IMOD as the initial argument string.	No

Chapter 11: CA-L-Serv Commands

This chapter details CA-L-Serv Commands and provides information about their usage. For more information on any of the CA-L-Serv commands, see the chapter "CA-L-Serv" in the *Administration Guide*.

This section contains the following topics:

Where to Issue Commands or Statements

Commands can be issued from a console, extended console, or from the CA-L-Serv parameter data set.

To issue a CA-L-Serv command from a console, prefix it with an MVS MODIFY command (or its abbreviation, F). Include the name of the CA-L-Serv started task. For example, to direct an ATTACH FILESERVER command to the CA-L-Serv started task, issue this command from a console:

```
F LSERV,ATTACH FILESERVER
```

To execute a sequence of CA-L-Serv commands from a member in the CA-L-Serv parameter data set, issue the READ command as follows:

```
F LSERV,READ member
```

You need system-level authority to issue CA-L-Serv commands from a console.

Statements can be issued from a partitioned data set member or in the JCL of the job (depending on the statement being used).

When Commands and Statements Take Effect

Commands take effect immediately if you issue them from a console. A command remains in effect until you issue a new command that overrides it or until you restart CA-L-Serv.

To make a command take effect automatically at start-up time, specify it in a command member that CA-L-Serv reads at start-up time. For details on issuing commands at start-up time, see the chapter "CA-L-Serv" in the Administration Guide.

Statements take effect when CA-L-Serv reads the partitioned data set member or executes the job containing the statement.

Truncate Names and Parameters

You can truncate command names and parameter keywords. In syntax diagrams, the shortest truncation for a command name (but not for parameters) is shown.

You cannot truncate statement names or parameters.

Continuation Characters and Leading Spaces

- To continue a command or statement on the next line, use a comma as an end-of-line continuation character.
- Leading spaces are ignored.

Delimit Names, Parameters, and Values

- To delimit a command or statement name from its parameters, use either a space or a comma. For example, you can specify either ATTACH,FILESERVER or ATTACH FILESERVER.
- To delimit parameters from their values, use an equal sign or parentheses. For example, specify REUSE=YES or REUSE(YES).
- When you specify multiple values for a parameter, always enclose the values in parentheses and separate them with commas or spaces. For example, specify OPTION(TRIM,DEFER).

Routing Command Output

By default, CA-L-Serv routes the command output to the console or TSO session from which the command was issued. If a console display area is defined, CA-L-Serv routes output to area A (out-of-line); otherwise, it routes output to display area Z (in-line).

To route output differently, append the MCS L parameter to any CA-L-Serv command. In addition to the standard L parameter values, you can specify L(LOG) to route output to the CA-L-Serv trace data set.

ACTIVATE Command

The ACTIVATE command activates a VTAM communication node. The ACTIVATE command must be executed once on each of two MVS systems between which you want to set up a VTAM communication route.

Syntax

ACtivate *node* [CONTYPE (*lutype*)]... [RETRMAX (*nn*)]...[RETRY (*seconds*)]...

The following table explains the operands:

Operand	Explanation
node	Identifies the ACB name of CA-L-Serv on the target system with which this system will communicate. This ACB name must match the ACB name specified on the ATTACH COMMSERVER command on the target system.
CONTYPE	For this node, overrides the default VTAM communications protocol that CA-L-Serv is using. This parameter must have the same value on the two nodes that make up a communication route. (The default value is set through the CONTYPE parameter on the ATTACH command.)
RETRMAX	For this node, overrides the default setting that determines how many times CA-L-Serv tries to reactivate routes that become inactive. (The default value is set through the RETRMAX parameter on the ATTACH command.)
RETRY	For this node, overrides the default setting that tells CA-L-Serv whether to reactivate routes when they become inactive. (The default value is set through the RETRY parameter on the ATTACH command.)

ACTIVATE commands are usually issued from the CA-L-Serv parameter data set. Specify them after the ATTACH command for the communications server. For information on creating cross-system routes for CA-L-Serv, see the chapter "CA-L-Serv" in the Administration Guide.

Example

To activate a VTAM communications route between system 1 (where the CA-L-Serv ACB name is SYS1) and system 2 (where the CA-L-Serv ACB name is SYS2), issue the following command on system 1:

```
ACTIVATE SYS2
```

And issue the following command on system 2:

```
ACTIVATE SYS1
```

ADDFILE Command

The ADDFILE command adds a file to the CA-L-Serv management. The CA-L-Serv file server controls access to all managed files.

Syntax

```
ADDFILE ddname [dsname] [[GROUP(id)] [ LOGID(name)] [POOL(nn)
               [[BUFND(nn)] [BUFNI(nn)] [STRNO(nn)]]]...]
               [OPTION processing_option_list] [[JCLMEMBER(name)]
                [PASSWORD(xxxx)]]... [DISP (SHR|OLD)]
```

The following table explains the operands:

Operand	Explanation
BUFND	Determines how many data buffers are allocated for this file. Specify a value from 3 to 9999. **Default Value:** 5
BUFNI	Determines how many index buffers are allocated for this file. Specify a value from 3 to 9999. **Default Value:** 5
ddname	Represents the DD name of the file you want CA-L-Serv to manage.

Operand	Explanation
DISP	Determines whether the file server requires exclusive use of the file. Specify one of the following values: SHR Specify SHR to allow shared access to the file. The file may be allocated to other users, but VSAM sharing options apply. The SHROPTIONS must be set as follows or integrity exposures will exist: SHROPTION=(1,3) or SHROPTION(2,3). OLD Specify OLD to require exclusive use of the file. VSAM sharing options are ignored and treated as SHROPTION=(1,3). **Default Value:** SHR
dsname	Represents the dsname of the file you want CA-L-Serv to manage. If you allocate the file through a DD statement in the CA-L-Serv start-up procedure, you can omit this parameter. **Note:** With CA-L-Serv 3.5, the allocation of files using DD statements with a DISP of OLD to ensure that the CA-L-Serv region has exclusive control is no longer necessary. CA recommends that the DISP parm of the ADDFILE command be used instead.
GROUP	Associates the file with a file group. If the file group does not already exist, CA-L-Serv the file group under the group ID you provide. The group ID can be up to 8 characters long.
JCLMEMBER	Identifies the partitioned data set member that contains JCL for archiving file groups automatically. The name of the data set must be included in the concatenation the //JCLLIB DD statement in the CA-L-Serv start-up procedure. Specify JCLMEMBER only when you also specify the GROUP and OPTION(SUBMIT) parameters.
LOGID	Associates the file with a change log. Updates to the file are automatically recorded in this log.

Operand	Explanation
OPTION	Sets processing options for a file. Specify one or more of the following values: APPEND For a file group, selects the last file that was written to and appends data to the end of the file. This parameter affects only initial file selection. Specify OPTION(APPEND) only when the GROUP parameter is also specified. DEFER Defers write operations for a record until another operation needs the VSAM buffer or until MAXDORM (a parameter on the OPTIONS command) seconds have elapsed. Use DEFER only with LSR buffer pools. INPUT Restricts access to input functions only. Update, insert, and erase functions are not allowed. MINLEN Pads the record being read with blanks until it is *nn* bytes long. Do not specify MINLEN unless directed to do so by the client documentation. SUBMIT For a file group, automatically archives a file when it becomes full. Use the JCLMEMBER parameter to indicate where the JCL is for this batch job. TRIM Trims trailing blanks from the data portion of variable-length records. DEFER, MINLEN, and TRIM have a critical impact on performance. For details, see the chapter "CA-L-Serv" in the *Administration Guide.*
PASSWORD	Identifies the 1- to 8-character password that must be used to open the file.
POOL	Indicates the type of VSAM buffer pool CA-L-Serv should use for this file. Specify one of these values: To use a private buffer pool, specify 0. To use an LSR buffer pool, specify a value from 1 to 15. When you use an LSR buffer pool, CA-L-Serv ignores the BUFND, BUFNI, and STRNO parameters. **Default Value:** POOL(0)

Operand	Explanation
STRNO	Determines how many VSAM placeholders are allocated for a file assigned to a private buffer pool. This parameter limits the number of current requests that clients can make for the file. Specify a value from 3 to 255. **Default Value:** 16

Note:

- ADDFILE commands are usually issued from the CA-L-Serv parameter data set. Specify them after the ATTACH command for the file server and after the ADDPOOL commands that define LSR buffer pools.
- When you share CA-L-Serv files in a multiple-MVS system environment, issue ADDFILE commands only on the system where your host server is running.

Example

To place the DPMFSIF file under the CA-L-Serv management and assign it to buffer pool 1, issue this command:

```
ADDFILE DPMFSIF BUNDL.V47GA.DPMFSIF POOL(1),
  OPTION(MINLEN(270),TRIM,DEFER)
```

ADDLOG Command

The ADDLOG command lets you define a new log and start recording information in it.

Syntax

```
ADDLog name [[DDNAMES(ddlist)] [DSNAMES(dsnlist)] [SYSOUT(class)]]
```

The following table explains the operands:

Operand	Explanation
DDNAMES	References DD statements in the CA-L-Serv start-up procedure where log files are defined. Specify up to four DD names, separated by commas or spaces.
DSNAMES	Identifies cataloged data sets that store log files. Specify up to four data set names, separated by commas or spaces.
name	Identifies the 1- to 8-character name for a log. To define a message log or trace log, specify MSGLOG or TRACE as the log name.

Operand	Explanation
SYSOUT	Defines log files as SYSOUT data sets in the class you specify here. Specify a single SYSOUT class. If you want to print log files, define them as SYSOUT data sets.

Note:

- ADDLOG commands are usually issued from the CA-L-Serv parameter data set.
- ADDLOG commands that define message logs should precede ATTACH commands. ADDLOG commands that define change logs should precede the ADDFILE commands for managed files.

Examples

To define LOG1 as a SYSOUT data set in class A, issue this command:

```
ADDLOG LOG1 SYSOUT(A)
```

- To define the CHNGLOG log through DD statements in the CA-L-Serv start-up procedure, issue this command:

```
ADDLOG CHNGLOG DDNAMES(CHNGLOG1,CHNGLOG2,CHNGLOG3)
```

ADDPOOL Command

The ADDPOOL command LSR buffer pools for files that CA-L-Serv is managing.

Syntax

```
ADDPool nnnn (size, count)... [STRNO (nnn)]
```

The following table explains the operands:

Operand	Explanation
count	Represents the number of buffers you want to allocate in the size specified with the *size* variable. The minimum value that you may define is 3.
nn	Assigns a number to the buffer pool you are defining. Specify a value from 1 to 15.
size	Represents the buffer size in bytes.

Operand	Explanation
STRNO	Determines how many VSAM placeholders are concurrently allocated for the file. This option limits the number of current requests that clients can make for the file. Specify a value from 3 to 255. **Default Value:** 16

Note:

- ADDPOOL commands are usually issued from the CA-L-Serv parameter data set. Specify them between the ATTACH command for the file server and the ADDFILE commands for files.
- You can specify up to four (*size,count*) pairs.
- When you share files in a multiple-MVS system environment, issue ADDPOOL commands only on the system where your host server is running.
- VSAM allocates approximately 340 bytes of ESQA per placeholder.

Example

To define a buffer pool that is suitable for the page files in CA Bundl, issue this command:

```
ADDPOOL 1 (1024,10) (32768,10)
```

ATTACH Command

The ATTACH command lets you start CA-L-Serv servers on a system and set operating values for them.

Syntax

```
ATTach COMMSERVER [[LOGID(name)] [XMVS(YES|NO) [[ACBNAME(name|NONE)]
                  [CONTYPE(luvalue)] [HOLDBUF(nnnn)] [MAXSENDSIZE(kilobytes)
                  [MAXSESSIONS(nnnn)] [RECBUFFSIZE(nK)] [RETRMAX(nn)]
                  [RETRY(seconds)] [SENDLIMIT(nnn)] [XCF(YES|NO)]]]

ATTach FILESERVER [[BUFFERSIZE(bytes)] [COMMSERVERSSN(ssname)]
                 [SERVERTYPE(type)]]

ATTach SQLSERVER  [[AUDIT(msgtype)] [HOSTVARPFX(char)] [LOGID(name)]
                  [MATCHLIMIT(nnnn)] [OPERATORPFX(char)] [SCANLIMIT(nnnn)]]
```

The following table explains the operands:

Operand	Explanation
ACBNAME	Identifies the CA-L-Serv ACB name on this system. This value must be provided if you are using VTAM communication. The value corresponds to the application ID that identifies CA-L-Serv to VTAM on the current system. This ID must be unique within your network. Specify NONE if you do not want to use VTAM. **Default Value:** NONE
AUDIT	Determines which SQL messages are logged. Specify one of these values: To log only error messages, specify ERROR. To log all messages, specify STATEMENTS. **Default Value:** ERROR
BUFFERSIZE	Sets the size (in bytes) of the data buffer that the file server uses to receive data from the communications server. Specify a value from 1024 to 32768. **Default Value:** 4096
COMMSERVER	Activates the CA-L-Serv communications server.

Operand	Explanation
COMMSERVERSSN	Tells the file server what copy of the communications server to use. Specify one of these values: To use the communications server running in the same address space as the file server, specify *. To use a different communications server, specify the subsystem name for the appropriate CA-L-Serv started-task. **Default Value:** *
CONTYPE	Identifies the protocol when using VTAM communication. Specify one of these values: To use VTAM LU 0, specify LU0. To use VTAM LU 6.2, specify LU62. **Default Value:** LU0
FILESERVER	Activates the CA-L-Serv file server.
HOLDBUF	Indicates the maximum number of data buffers that CA-L-Serv should use per client for incoming data. Once this maximum is reached, incoming data is rejected until the client receives the data buffers' contents. Specify a value from 0 to 999. **Default Value:** 10
HOSTVARPFX	Assigns a prefix character that the SQL server uses to identify host variables. **Default Value:** (colon)
LOGID	Identifies an auxiliary log for certain messages issued by the communications server or the SQL server: For the communications server, the auxiliary log collects messages about session initiations and terminations, as well as cross-system transactions. For the SQL server, the auxiliary log collects messages about SQL statements. **Default Value:** MSGLOG
MATCHLIMIT	Sets the maximum number of rows that the SQL server can return per SQL statement. Specify one of these values: If you do not want to set a maximum, specify 0. Otherwise, specify a value from 1 to 999999999. **Default Value:** 0
MAXSENDSIZE	Sets the maximum number of kilobytes for data transmission when using XMVS communication. Specify a value from 4 to 64. **Default Value:** 32

Operand	Explanation
MAXSESSIONS	Indicates the maximum number of clients that can be in session with the communications server. Specify a value from 1 to 100. **Default Value:** 32
OPERATORPFX	Assigns a prefix character that identifies special operators to the SQL server. **Default Value:** . (period)
RECBUFFSIZE	Sets the size of the transmission buffer for the communications server. This value applies only to VTAM communication and it must be set to the same value on all systems. Specify 1K, 2K, 4K, 8K, or 16K. **Default Value:** 4K
RETRMAX	When you tell CA-L-Serv to reactivate VTAM communication routes (through the RETRY parameter), this parameter sets the maximum number of retries: To retry indefinitely, specify 0. To retry a designated number of times, specify a value from 1 to 100. **Default Value:** 0
RETRY	Indicates whether CA-L-Serv should try to reactivate inactive VTAM communication routes: If you do not want to reactivate routes, specify 0. Otherwise, set the retry interval (in seconds) by specifying a value from 1 to 86400. **Default Value:** 0
SCANLIMIT	Determines the maximum number of rows that the SQL server can search per SQL statement: If you do not want to set a maximum, specify 0. Otherwise, set a maximum by specifying a value from 1 to 999999999. **Default Value:** 0
SENDLIMIT	Sets the maximum number of outstanding send requests per client. Once this limit has been reached, the client must wait for the next data transmission to occur. Specify a value from 1 to 999. **Default Value:** 32

Operand	Explanation
SERVERTYPE	Indicates how requests to access CA-L-Serv files should be handled by the file server. Specify one of these values: **HOST** - Identifies a server as the host server. This server handles all requests for access to files that CA-L-Serv is managing. Requests from remote servers are forwarded through the communications server. **LOCAL** - Handles access requests from the local system only. **REMOTE** - Forwards access requests for files to your host server. Requests are forwarded through the communications server. **Default Value:** LOCAL
SQLSERVER	Activates the CA-L-Serv SQL server.
XCF	Indicates whether you want to use XCF communication. Specify YES or NO. By default, CA-L-Serv does not use XCF. Specifying YES directs CA-L-Serv to automatically determine which systems are in the associated sysplex and dynamically initialize the corresponding XCF group and its members. **Default Value:** NO
XMVS	Indicates whether CA-L-Serv will be used to communicate between MVS systems. Unless directed to do so by the documentation for the application, specify YES. **Default Value:** YES

Notes:

- ATTACH commands are usually issued from the CA-L-Serv parameter data set.
- In the CA-L-Serv parameter data set, make sure that the ATTACH command always precedes any command which targets the corresponding server. For instance, CA-L-Serv will not accept ADDFILE and ADDPOOL commands unless it has successfully processed an ATTACH FILESERVER command.

  If you shutdown a server (with the DETACH command) and restart it using ATTACH *server*, the parameter values default to the values in effect at the time of the DETACH (whether set by the ATTACH command or modified by the OPTIONS command).

Examples

- To start a remote file server, issue the following command:

  ```
  ATTACH FILESERVER SERVERTYPE(REMOTE)
  ```

- To start the communications server using both XCF communication and VTAM LU 0 (ACB name SYS01), issue the following command:

  ```
  ATTACH COMMSERVER ACBNAME(SYS01) XCF(YES)
  ```

Note: Since LU 0 is used by default, it is not necessary to specify the associated parameter.

CLOSEFILE Command

The CLOSEFILE command disables a file so that requests for it fail. By doing this, you make the file unavailable to jobs and users. To make the file available, use the OPENFILE command.

```
CLOSEFile ddname
```

ddname

Identifies the file you are closing.

Note:

- Issue the CLOSEFILE command from a console of an operator.
- To place requests for a file in a queue (rather than failing them), use the HOLDFILE command.

Example

To make the DPMFMAP file unavailable, issue this command:

```
CLOSEFILE DMPFMAP
```

CLOSELOG Command

The CLOSELOG command closes a log and all log files associated with it. By doing this, you tell CA-L-Serv to stop recording information in that log. To start recording again, issue the OPENLOG command.

Syntax

```
CLOSELOG ddname
```

ddname

Identifies the log that you are closing.

Note: Issue the CLOSELOG command from a console of an operator.

Example

To close log file LOG1, issue this command:

```
CLOSELOG LOG1
```

DEACTIVATE Command

The DEACTIVATE command deactivates a VTAM communication route.

Syntax

```
DEActivate route
```

route

Identifies the ACB name of CA-L-Serv on the target system. This ACB name must be the same ACB name that was specified on the ATTACH COMMSERVER command on the target system.

Note: DEACTIVATE commands are usually issued from consoles, rather than from the CA-L-Serv parameter data set. When you deactivate a route and then reactivate it (using ACTIVATE), the RETRY and RETRMAX values are set to the default value of zero unless you specify otherwise on the ACTIVATE command.

Example

To deactivate the communication route between system 1 and system 2 (where the CA-L-Serv ACB name on system 2 is ACBSYS2), issue this command on system 1:

```
DEACTIVATE ACBSYS2
```

DETACH Command

The DETACH command stops servers that run in the CA-L-Serv address space.

Syntax

```
DEtach COMMSERVER

DEtach FILESERVER

DEtach SQLSERVER
```

The following table explains the operands:

Operand	Explanation
COMMSERVER	Stops the CA-L-Serv communications server.
FILESERVER	Stops the CA-L-Serv file server.
SQLSERVER	Stops the CA-L-Serv SQL server.

Note: DETACH commands are usually issued from a console of an operator.

Example

To stop the file server, issue this command:

```
DETACH FILESERVER
```

DISPLAY Command

The DISPLAY command displays information about CA-L-Serv and its servers.

Syntax

```
Display [[ACTIVE] [ALL] [INIT] [LOGS] [MSGTABLE] [OPTIONS] [SSNAME] [STORAGE]
        [VERSION]]
Display [TASK(COMMSERVER) [[ALL] [APPLICATIONS] [INIT] [OPTIONS]
        [ROUTES[(ACTIVE|INACTIVE)] ]... ]]
Display [TASK(FILESERVER) [[ALL] [BUFFERPOOL] [DATABASE [[DDNAME(ddname)]
        [GROUP(name)] [RESET]]] [OPTIONS] [STATISTICS [[DDNAME(ddname)]
        [RESET] [=SERVICE]]] [SYSTEMS]]]
Display [TASK(SQLSERVER) [[ALL] [OPTIONS] [STATISTICS] [TABLE]]]
```

The following table explains the operands:

Operand	Explanation
ACTIVE	Displays a list of active CA-L-Serv servers. (See message LDM0420I.)
ALL	Displays all available information for CA-L-Serv and its active servers. Use the TASK parameter to limit the display to a particular server.
APPLICATIONS	Displays a list of clients using the communications server and the number of send and receive requests from each client. (See message LDM0920I.)
BUFFERPOOL	Displays information about LSR buffer pools defined for files that CA-L-Serv is managing. (See message LDM0552I). See the chapter "CA-L-Serv" in the Administration Guide and the IBM Using Data Sets manuals for a discussion of the fields displayed.
DATABASE	Displays information about each file that CA-L-Serv is managing. Information includes options set through the ADDFILE command and the status of the file. (See message LDM0522I). By default, CA-L-Serv displays all managed files: To display only one file, specify the name of the file name on the DDNAME parameter. To display only one file group, specify a group ID on the GROUP parameter. To display cumulative information and then reset the display, specify RESET.

Operand	Explanation
INIT	Displays values that cannot be changed while the CA-L-Serv shell or a CA-L-Serv server is running. In response, CA-L-Serv issues a series of messages (one for the shell and one per applicable server). By default, all of these messages are displayed. To display values of only a certain server, specify the TASK parameter also.
LOGS	Displays status information about the CA-L-Serv message, trace, and change logs. Each log file is listed separately. (See message LDM0750I.)
MSGTABLE	Displays a list of message tables, what language each table is in, and how many messages each table contains. (See message LDM0422I.)
OPTIONS	Displays values that you can change while the CA-L-Serv shell or a CA-L-Serv server is running. In response, CA-L-Serv issues a series of messages (one for the shell and one per applicable server). By default, all of these messages are displayed. Use the TASK parameter to limit the display to a particular server.
ROUTES	Displays information about VTAM and XCF communication routes associated with XMVS communication. (See message LDM0912I.) By default, the display includes all routes. To display only active routes, specify ROUTES(ACTIVE). To display only inactive routes, specify ROUTES(INACTIVE).
SSNAME	Displays a list of subsystems (including CA-L-Serv) that are running on a local system. Status information for each subsystem is also shown. (See message LDM0410I).

Operand	Explanation
STATISTICS	Displays information about I/O activity for VSAM files (in message LDM0546I) and SQL tables (in message LDM4080I). By default, CA-L-Serv displays one line per file in the display of the file server. Information is cumulative, as of the last time the statistics display was reset. To change the display, specify one or more of these parameters: To display only one file, specify the name of the file on the DDNAME parameter. To display cumulative statistics and then reset the display, specify RESET. To display an additional line that shows averages for all files, specify =SERVICE. You can also use the TASK parameter to limit the display to the file server or the SQL server.
STORAGE	Displays information about storage use, including the size of a storage block, how many blocks were allocated, and how many blocks were used. (See message LDM0425I.)
SYSTEMS	Displays a list of systems where the file server is running. For each system, the display includes statistics about send and receive requests, what type of server is running, and how many times CA-L-Serv split and reassembled data to fit its data buffer. (See message LDM0555I.)
TABLE	Displays information about SQL relational tables. (See message LDM4521I.) To limit the display to a particular table, specify the name of the table in place of *name*.
TASK	Limits the display to one of the CA-L-Serv servers. Use TASK with the INIT, OPTIONS, and/or STATISTICS parameters, or use TASK if you have an ambiguous truncation for a parameter. Specify COMMSERVER, FILESERVER, or SQLSERVER.
VERSION	Displays the CA-L-Serv release and maintenance level. (See message LDM0402I.)

Note: Issue the DISPLAY command from the CA-L-Serv parameter data set or from a console of an operator.

Example

To display information about the DPMFVIF file, issue this command:

```
DISPLAY DATABASE DDNAME(DPMFVIF)
```

ELSE Statement

The ELSE statement is used within an IFSYS/ENDIF block to direct commands to all systems that do not match the system names specified on the IFSYS statement.

Syntax

```
ELSE
```

There are no parameters that qualify the ELSE statement. However, the ELSE statement must be followed by a command or command block that is to be executed.

Note: You can specify this statement in the CA-L-Serv parameter data set.

For information on directing commands or statements to only particular systems, see the chapter "CA-L-Serv" in the Administration Guide.

Example

To execute an ATTACH FILESERVER SERVERTYPE(HOST) command on system SYS01 and to execute an ATTACH FILESERVER SERVERTYPE(REMOTE) on all other systems, specify these statements in the CA-L-Serv partitioned data set:

```
IFSYS SYS01
   ATTACH FILESERVER SERVERTYPE(HOST)
 ELSE
   ATTACH FILESERVER SERVERTYPE(REMOTE)
 ENDIF
```

ENDIF Statement

The ENDIF statement marks the end of a block of commands or statements (that begins with IFSYS) that should be executed on only particular systems.

Syntax

```
ENDIF
```

There are no parameters for this statement.

Note: You can specify this statement in the CA-L-Serv parameter data set.

Example

To execute an ATTACH FILESERVER SERVERTYPE(HOST) command only on system SYS01, specify these statements in the CA-L-Serv partitioned data set:

```
IFSYS SYS01
   ATTACH FILESERVER SERVERTYPE(HOST)
 ENDIF
```

HOLDFILE Command

The HOLDFILE command holds a file so that requests for it are placed in a queue. By doing this, you make the file unavailable to jobs and users.

Syntax

```
HOLDFile ddname
```

ddname

Identifies the file you are holding.

Note:

- HOLDFILE commands are usually issued from a console of an operator.
- To fail requests for a file (rather than placing them in a queue), use the CLOSEFILE command.

Example

To close the DPMFMAP file so users must wait for it to become available again, issue this command:

```
HOLDFILE DPMFMAP
```

IFSYS Statement

The IFSYS statement marks the beginning of a block of commands or statements that should be executed only on particular systems.

Syntax

```
IFSYS sysname
```

sysname

Identifies one or more systems that should execute the commands or statements following IFSYS. Each name must match a system name defined on the SYSNAME parameter for the OPTIONS command. If you specify several system names, separate them with commas.

Note:

- You can specify this statement in the CA-L-Serv parameter data set.
- Unless directed otherwise by the SYSNAME= parm of the CA-L-Serv start-up procedure (LSVPROC) the value specified on the IFSYS statement is compared with the SMFID of the MVS system.

Example

To execute an ATTACH FILESERVER SERVERTYPE(REMOTE) command only on systems SYS02 and SYS03, specify these statements in the CA-L-Serv partitioned data set:

```
IFSYS SYS02,SYS03
   ATTACH FILESERVER SERVERTYPE(REMOTE)
 ENDIF
```

INCLUDE Statement

The INCLUDE statement identifies a partitioned data set member containing CA-L-Serv commands and statements or a message table.

Syntax

```
INCLUDE member
```

member

Identifies the partitioned data set member. This member must be part of the CA-L-Serv parameter data set, or its dsname must be included in the concatenation of the //LDMCMND DD statement in the CA-L-Serv start-up procedure.

Note:

- You can specify this statement in a command member of the CA-L-Serv parameter data set.
- The INCLUDE statement may be used in a member that was itself called by another INCLUDE statement. However, make sure that you avoid loops, which are caused by including a member that directly or indirectly includes the current member.

Example

To execute commands in member BNDPARMA at start-up time, specify this statement in a command member:

```
INCLUDE BNDPARMA
```

MSG Statement

The MSG statement provides a message definition for CA-L-Serv or for a client.

Syntax

```
MSG 'text ' [routinginfo]
```

The following table explains the operands:

Operand	Explanation
text	Provides the message prefix and text that you want CA-L-Serv to use. The text must be enclosed in single quotation marks.
routinginfo	Provides routing information.

Note:

- You can specify this statement only in a message table.
- Specify MSG statements after the TABLE statement.
- For the complete syntax of the MSG statement and information on how to use it, see the LSERVMSG member of the CA-L-Serv parameter data set.

Example

To use the text TASK NOT ACTIVE for message LDM0475E, specify this statement in your message table:

```
MSG 'LDM0475E   TASK NOT ACTIVE'
```

MSGTABLE Command

The MSGTABLE command immediately loads the message definitions contained in (or pointed to from) a partitioned data set member that you identify. By doing this, you make those message definitions take effect immediately.

Syntax

```
MSGTABLE member
```

member

Identifies the partitioned data set member that contains (or points to) the message definitions. This member must be part of the CA-L-Serv parameter data set, or its dsname must be included in the concatenation of the //LDMCMND DD statement in the CA-L-Serv start-up procedure.

Note: Issue the MSGTABLE command from the CA-L-Serv parameter data set or from a console of an operator.

Example

To load message definitions stored in the MSGDEF member, issue this command:

```
MSGTABLE MSGDEF
```

OPENFILE Command

The OPENFILE command enables a file that was disabled with the CLOSEFILE command. By doing this, you make the file available again to jobs and users.

Syntax

```
OPENFile ddname
```

ddname

Identifies the file being enabled.

Note: OPENFILE commands are usually issued from a console of an operator.

Example

To make the DPMFVIF file available, issue this command:

```
OPENFILE DPMFVIF
```

OPENLOG Command

The OPENLOG command opens a log that was closed with the CLOSELOG command. After a log is opened, CA-L-Serv can start recording information in that log.

Syntax

```
OPENLog name
```

name

Identifies the log that you are opening.

Note: OPENLOG commands are usually issued from a console of an operator. You do not need to issue an OPENLOG command after you issue an ADDLOG, PRINTLOG, or SWITCHLOG command.

Example

To start recording information in log LOG1, issue this command:

```
OPENLOG LOG1CA-L-Serv
```

OPTIONS Command

The OPTIONS command lets you set operating values for CA-L-Serv.

Syntax

```
OPTions SVCDUMP(YES|NO)

OPTions [TASK(COMMSERVER)] [[LOGID(name)] [ XMVS(YES|NO)
        [[CONTYPE(value)] [HOLDBUF(nnnn)] [MAXSENDSIZE(kilobytes)]
        [MAXSESSIONS(nnnn)] [RETRMAX(nn)]
        [RETRY(seconds)] [SENDLIMIT(nnn)] [XCF(YES|NO)]]]

OPTions [TASK(FILESERVER)] MAXDORM(seconds)

OPTions [TASK(SQLSERVER)] [[HOSTVARPFX(char)] [LOGID(name)] [MATCHLIMIT(nnnn)]
        [OPERATORPFX(char)] [SCANLIMIT(nnnn)]]]
```

The following table explains the operands:

Operand	Explanation
CONTYPE	Overrides the default VTAM communications protocol for the communications server set by the CONTYPE parameter of the ATTACH command.
HOLDBUF	Overrides the default number of transmission buffers per client set by the HOLDBUF parameter of the ATTACH command.
HOSTVARPFX	Overrides the SQL prefix character set by the HOSTVARPFX parameter of the ATTACH command.
LOGID	Overrides the name of an auxiliary log set by the LOGID parameter of the ATTACH command.
MATCHLIMIT	Overrides the maximum number of SQL rows returned set by the MATCHLIMIT parameter of ATTACH command.
MAXDORM	Sets the maximum number of seconds CA-L-Serv waits to write buffers to disk after the last deferred write request occurred. Specify a value from 1 to 3600. **Default Value:** 60
MAXSENDSIZE	Overrides the maximum data size for transmissions set by the MAXSENDSIZE parameter of the ATTACH command.
MAXSESSIONS	Overrides the maximum number of clients that can use the communications server set by the MAXSESSIONS parameter of the ATTACH command.
OPERATORPFX	Overrides the special operator prefix character set by the OPERATORPFX parameter of the ATTACH command.

Operand	Explanation
RETRMAX	Overrides the setting that tells CA-L-Serv how often to try reactivating VTAM communication routes that become inactive. This value is set by the RETRMAX parameter of the ATTACH command and only applies to VTAM communication.
RETRY	Overrides the setting that tells CA-L-Serv whether to reactivate VTAM communication routes that become inactive. This value is set by the RETRY parameter of the ATTACH command and only applies to VTAM communication.
SCANLIMIT	Overrides the maximum number of SQL rows searched set by the SCANLIMIT parameter of the ATTACH command.
SENDLIMIT	Overrides the maximum number of pending send requests per client set by the SENDLIMIT parameter of the ATTACH command.
SVCDUMP	Indicates whether CA-L-Serv should generate an SVC dump if an error occurs within its address space. Specify YES or NO. **Default Value:** YES
TASK	Identifies the server that should execute the command. Use TASK with the LOGID parameter or whenever you use an ambiguous truncation for a parameter. Specify COMMSERVER, FILESERVER, or SQLSERVER.
XCF	Indicates whether you want to activate or deactivate XCF communication. Specify YES or NO.
XMVS	Indicates whether you want to activate or deactivate the XMVS communication service. Specify YES or NO.

Note: Issue the OPTIONS command from the CA-L-Serv parameter data set or from a console of an operator.

Example

To tell CA-L-Serv not to generate an SVC dump if an error occurs, issue this command:

```
OPTIONS SVCDUMP(NO)
```

PRINTLOG Command

The PRINTLOG command prints a log to a SYSOUT class.

Syntax

```
PRintlog name
```

name

Identifies the log that you want to print.

Note:

- PRINTLOG commands are usually issued from a console of an operator.
- You can only print logs that were defined as SYSOUT data sets.
- When you print the log, CA-L-Serv closes and deallocates the current SYSOUT data set. Then it allocates and opens a new SYSOUT data set for the log.

Example

To print log file LOG1 to the SYSOUT class specified on the ADDLOG command, issue this command:

```
PRINTLOG LOG1
```

READ Command

The READ command directs CA-L-Serv to read the specified partitioned data set member and execute each command in it.

Syntax

```
READ member
```

member

Identifies the partitioned data set member that contains the commands you want to execute. This member must be part of the CA-L-Serv parameter data set, or its dsname must be included in the concatenation of the //LDMCMND DD statement in the CA-L-Serv start-up procedure.

Note: READ commands are usually issued from a console.

Example

To read the FILECMDS member and execute the commands in that member, issue this command:

```
READ FILECMDS
```

RELEASEFILE Command

The RELEASEFILE command releases a file that was held using the HOLDFILE command. By doing this, you make the file available again to users and jobs.

`RELeasefile` *`ddname`*

ddname

Identifies the file you are releasing.

Note: RELEASEFILE commands are usually issued from a console.

Example

To release file DPMFVIF, issue this command:

```
RELEASEFILE DPMFVIF
```

REMOVEFILE Command

The REMOVEFILE command removes a file from the management of the file server.

Syntax

`REMOVEFile` *`ddname`*

ddname

Identifies the file you are removing.

Note: REMOVEFILE commands are usually issued from a console. If a file is allocated through a DD statement in the CA-L-Serv start-up procedure, the REMOVEFILE command will still remove that file from the management of the file server, but it will not be deallocated from the associated CA-L-Serv address space.

Example

To remove file DPMFVIF from the management of the file server, issue this command:

```
REMOVEFILE DPMFVIF
```

REMOVELOG Command

The REMOVELOG command removes a log from the CA-L-Serv management. By doing this, you permanently close the log and all log files associated with it.

Syntax

```
REMOVELog name
```

name

Identifies the log that you are removing.

Note:

- REMOVELOG commands are usually issued from a console.
- If the log file is open when you issue the REMOVELOG command, CA-L-Serv writes all active buffers and then closes the file.

Example

To remove log LOG1 from the CA-L-Serv management, issue this command:

```
REMOVELOG LOG1
```

SHUTDOWN Command

The SHUTDOWN command stops CA-L-Serv on the local system.

Syntax

```
SHutdown [IMMEDIATE]
```

IMMEDIATE

Stops CA-L-Serv immediately, even if all servers have not terminated. If any servers are still running, CA-L-Serv may experience an A03 abend.

Use IMMEDIATE only when normal shutdown requests fail. It may be most useful when you need to stop a remote system and an active SQL request is executing on the host system.

Note:

- Issue the SHUTDOWN command from a console.
- You do not need to issue a DETACH command before a SHUTDOWN command. CA-L-Serv automatically notifies servers that it is terminating.
- You can also use the MVS STOP command to stop CA-L-Serv.

Example

To stop CA-L-Serv, issue this command:

```
F LSERV,SHUTDOWN
```

SWITCHFILE Command

The SWITCHFILE command allows you to immediately switch to a new file in a file group. By doing this, you tell CA-L-Serv to close any file that it is currently using and to start recording data in a new file.

Syntax

```
SWITCHFile groupid [ddname]
```

The following table explains the operands:

Operand	Explanation
ddname	Selects a particular file in the file group. **Default Value:** CA-L-Serv selects the next available file.
groupid	Indicates what file group you want to use.

Note: SWITCHFILE commands are usually issued from consoles.

Example

To switch to file JRNL1 in file group UGRPID, issue this command:

```
SWITCHFILE UGRPID JRNL1
```

SWITCHLOG Command

The SWITCHLOG command lets you switch to a new log file. By doing this, you tell CA-L-Serv to start recording information in the next file associated with the log.

Syntax

```
SWITCHLog name
```

name

Identifies the log.

Note:

- Issue the SWITCHLOG command from a console.
- Use this command only if you provided DD names or dsnames for the log. Do not use it when the log is defined as a SYSOUT data set.
- Use this command only when multiple DD names are defined for a single log.
- The order in which you defined log files (on the ADDLOG command) determines which file CA-L-Serv selects.
- When CA-L-Serv is using the last log file and you issue a SWITCHLOG command, CA-L-Serv starts using the first log file and overwrites its contents.

Example

To switch to the next file for log LOG1, issue this command:

```
SWITCHLOG LOG1
```

TABLE Statement

The TABLE statement identifies a message table for CA-L-Serv or a client.

Syntax

```
TABLE name [LANGUAGE (xxxx)] [DEFAULT]
```

The following table explains the operands:

Operand	Explanation
DEFAULT	Identifies this language as the default language. **Default Value:** The English language message table is used.
LANGUAGE	Identifies a language.
name	Identifies the message table. Each client determines what table name to use.

Note: You can specify this statement in message table members only. Specify the TABLE statement before MSG statements.

Example

To provide an English message table for CA Bundl, specify this statement in your message table:

```
TABLE BUNDLTAB LANGUAGE(ENGLISH) DEFAULT
```

WRITELOG Command

The WRITELOG command ensures that all log entries are written to a log.

Syntax

```
WRITELOG name
```

name

Identifies the log that should be updated.

Note: Issue the WRITELOG command from a console. CA-L-Serv automatically writes data buffers to log files when the buffers become full. However, if CA-L-Serv is recording log information for a rarely used file, the log may not contain the most recent entries because the buffers have not been filled. You can issue the WRITELOG command to write the buffers' contents to the log so that the log is up to date. The WRITELOG command does not close the file.

Example

To write the active buffers for LOG1, issue this command:

```
WRITELOG LOG1
```

LDMAMS Statements

ARCHIVE Statement

The ARCHIVE statement saves the contents of a file group or one of its files. This statement is used by the LDMAMS utility during an archive process.

Syntax

```
ARCHIVE GROUP (id) OUTFILE(ddname) [SWITCH]

ARCHIVE INFILE(ddname) OUTFILE(ddname) [SWITCH]
```

GROUP

Archives all non-empty files associated with the group ID you specify.

INFILE

Archives only the specified file.

OUTFILE

Identifies the sequential file that will store the archived data.

SWITCH

Closes the currently active file, archives it, and makes a new file active.

Note: You can specify this statement only as input for an LDMAMS job. The files that you reference on this statement must be defined on DD statements in the JCL for the LDMAMS job.

Example

To archive file group UGRPID and store the archive data in the OUTGDG file, specify this statement as input to the LDMAMS job:

```
ARCHIVE GROUP(UGRPID) OUTFILE(OUTGDG) SWITCH
```

COMMTEST Statement

The COMMTEST statement creates a test application that the LDMAMS utility uses to test data transmission associated with XMVS communication.

Syntax

```
COMMTEST [[APPL(name1)] [QUAL(name2)]]...
```

APPL

Represents the 1- to 8-character primary ID of the test application.

Default Value: Name of the current LDMAMS job.

QUAL

Represents the 1- to 8-character secondary ID of the test application.

Default Value: COMMTEST

Note:

- You can specify this statement only as input for the LDMAMS utility. The file server must be active on the local system.
- Specify this statement before any other testing statements (WAIT, SEND, RECEIVE, QUERY, or END).

Example

To create a test application named TESTAPP SYS1 that receives data, specify this statement as input to your LDMAMS job:

```
COMMTEST APPL(TESTAPP) QUAL(SYS1)
```

COMPRESS Statement

The COMPRESS statement compresses a VSAM file that CA-L-Serv is managing. The LDMAMS utility uses this statement during a compression process.

Syntax

```
COMPRESS INFILE(ddname) WORKFILE(ddname)
```

INFILE

Identifies the VSAM file that you are compressing. This DD name must match the DD name on the ADDFILE command.

WORKFILE

Identifies the sequential file that serves as a temporary work file for the compression process. The LDMAMS JCL must include a DD statement for this file.

Note: You can specify this statement as input for an LDMAMS job. A file can be compressed only if it was defined with the REUSE option of the DEFINE CLUSTER control statement using the IDCAMS utility.

Example

To compress file FILE1, using TEMPFILE as a work file, specify this statement as input to the LDMAMS job:

```
COMPRESS INFILE(FILE1) WORKFILE(TEMPFILE)
```

END Statement

The END statement ends a data transmission test performed by the LDMAMS utility and terminates the test application.

```
END
```

Note:

- You can specify this statement only as input to the LDMAMS utility.
- Specify this statement at the end of every LDMAMS job that you use to test data transmission.
- Specify this statement after all other statements in your data transmission stream.

Example

To end a data transmission test and delete the test application you created, specify this statement as input to your LDMAMS job:

```
END
```

QUERY Statement

The QUERY statement displays the list of clients known to the communication server. The LDMAMS utility uses this statement to test cross-system communication.

Syntax

```
QUERY
```

Note: You can specify this statement only as input to the LDMAMS utility.

Example

To see what clients have sessions with the communications server, specify this statement as input to your LDMAMS job:

```
QUERY
```

RECEIVE Statement

The RECEIVE statement retrieves data sent by a test application running on another system.

Syntax

```
RECEIVE [COUNT(nn)]...
```

COUNT

Represents the number of times that data will be received. This value must match the value for the COUNT parameter on the SEND statement for the test application transmitting the data.

Default Value: 1 or previous value - see Notes

Notes:

- You can specify this statement only as input to the LDMAMS utility.
- The value for the COUNT parameter may be set from a previous RECEIVE or SEND statement. If this parameter is not specified in the RECEIVE statement, CA-L-Serv uses the most recent value established within a COMMTEST application.

Example

To receive all test data, specify this statement as input to your LDMAMS job:

```
RECEIVE
```

REPRO Statement

The REPRO statement copies data between VSAM files and sequential files. You can use REPRO statements in an LDMAMS job to back up files or to restore files from backup copies.

Syntax

```
REPRO INFILE(inddname) OUTFILE(outddname)
        [[FROMKEY(key)] [REPLACE] [REUSE] [TOKEY(key)]]
```

FROMKEY

Identifies the first input record. Specify a full or partial record key. If the key contains special characters (such as blanks or punctuation characters), enclose it in quotes.

Default Value: CA-L-Serv starts with the first record in the file.

INFILE

Identifies the source file. For a backup operation, this is the VSAM file that you are backing up. For a restore operation, this is the sequential file that contains the backup copy you want to use.

OUTFILE

Identifies the target file. For a backup operation, this is the sequential file that will contain the backup copy. For a restore operation, this is the VSAM file that you are restoring.

REPLACE

Replaces existing records in the target file with records from the source file. If CA-L-Serv finds a duplicate record, it uses the last copy of the record. Omit this parameter on backup operations.

Default Value: CA-L-Serv skips duplicate records.

REUSE

Deletes the contents of the target file before adding records to it. Omit this parameter on backup operations. A file can be reused only if it was defined with the REUSE option of the DEFINE CLUSTER control statement using the IDCAMS utility.

TOKEY

Identifies the last input record. Specify a full or partial record key. If the key contains special characters (such as blanks or punctuation characters), enclose it in quotes.

Default Value: CA-L-Serv copies through the last record in the file.

Notes:

- You can specify this statement only as input to the LDMAMS utility.
- The name of the VSAM file that you reference must match the name of the file in its ADDFILE command. It must not be referenced by a DD statement in the LDMAMS JCL.
- The LDMAMS JCL must contain a DD statement for any sequential file that you use.

Example

- To back up FILE1 and store its contents in BCKFILE1, specify this statement as input to the LDMAMS job:

```
REPRO INFILE(FILE1) OUTFILE(BCKFILE1)
```

- To replace the contents of FILE2 with the contents of BCKFILE2, specify this statement as input to the LDMAMS job:

```
REPRO INFILE(BCKFILE2) OUTFILE(FILE2) REPLACE REUSE
```

RESET Statement

The RESET statement deletes the contents of a VSAM file that CA-L-Serv is managing.

Syntax

```
RESET OUTFILE(ddname)
```

OUTFILE

Identifies the file. This DD name must match the DD name on the ADDFILE command.

Note:

You can specify this statement only as input to the LDMAMS utility.

- A file can be reset only if it was defined with the REUSE option of the DEFINE CLUSTER control statement using the IDCAMS utility.

Example

To delete the contents of file FILE1, specify this statement as input to the LDMAMS job:

```
RESET OUTFILE(FILE1)
```

SEND Statement

The SEND statement transmits data to a test application executing on another MVS system.

Syntax

```
SEND [[APPL(name1)] [COUNT(nn)] [LENGTH(bytes)] [QUAL(name2)]]
```

APPL

Represents the primary ID of the target test application. This ID must match the primary ID on the COMMTEST statement that creates the application.

Default Value: COMMTEST or previous value-see Notes.

COUNT

Represents the number of times that data will be sent.

Default Value: 1 or previous value - see Notes.

LENGTH

Represents the size of the data block that will be transmitted.

Default Value: 512

QUAL

Represents the secondary ID of the target application. This ID must match the secondary ID on the COMMTEST statement that creates the target application.

Default Value: COMMTEST or previous value-see Notes.

Notes:

- You can specify this statement only as input to the LDMAMS utility.
- Specify this statement after any WAIT statement you have included.
- The values for the APPL and QUAL parameter may be set by a previous SEND or WAIT statement, and the value for the COUNT parameter may be set by a previous RECEIVE or SEND statement. If these parameters are not specified in the SEND statement, the default is the most recent value established within a COMMTEST application.

Example

To send data to test application TESTAPP SYS2, specify this statement as input to your LDMAMS job:

```
SEND APPL(TESTAPP) QUAL(SYS2)
```

WAIT Statement

The WAIT statement causes the LDMAMS utility to wait until an application on a remote system becomes active.

Syntax

```
WAIT [[APPL(name1)] [LIMIT(minutes)] [QUAL(name2)]]
```

APPL

Represents the primary ID of the test application that will receive data. This ID must match the primary ID of the receiving application, which is specified on the COMMTEST statement that creates the application.

Default Value: Name of the current LDMAMS job or previous value. See the Notes section.

LIMIT

Represents the number of minutes that the test application should wait. If the receiving application is not running by then, the sending application will end.

Default Value: 60 or previous value. See the Notes section.

QUAL

Represents the secondary ID of the test application that will receive data. This ID must match the secondary ID of the receiving application, which is on the COMMTEST statement that creates the application.

Default Value: COMMTEST or previous value. See the Notes section.

Notes:

- You can specify this statement only as input to the LDMAMS utility.
- The WAIT statement should precede any SEND statements in the LDMAMS JCL.
- The values for the COUNT and LIMIT parameters may be set in a previous WAIT statement. If these parameters are not specified in the WAIT statement, the default is the most recent value established within a COMMTEST application.

Example

To wait until application TESTAPP SYS2 is active before sending test data, specify this statement as input to your LDMAMS job:

```
WAIT APPL(TESTAPP) QUAL(SYS2)
```

SQL Statements

Usage Rules

- To issue an SQL statement, you must prefix that statement with the LSQL command. You do not need to prefix sub-queries within a statement with the LSQL command.
- Syntax is diagrammed along the line from left to right. If the entire syntax cannot fit on one line, it splits and continues below the first line. A right arrow at the end of the line indicates that the diagram is continued on the next line.
- Required parameters fall *on* the line. Optional parameters fall *below* the line. Required and optional parameters are listed vertically in groups.
- Parameters that can be repeated are indicated by a repeating arrow.
- Variables are printed in italics. When there is a choice of variables, they are separated by a bar (|).

Issue LSQL Commands

You can issue the LSQL command from a CLIST, from a REXX procedure, from a TSO terminal, or from a batch job. For Notes on the LSQL command, see the chapter "CA-L-Serv" in the Administration Guide.

Restrictions on Use

You can issue most LSQL commands from a TSO terminal or a batch job. However, you cannot use cursors from a TSO terminal or a batch job.

You can issue any LSQL command from a CLIST or from a REXX procedure. However, you should keep two things in mind when issuing LSQL commands from a CLIST or from a REXX procedure:

- When using a SELECT statement, values from only one row at a time are returned. If you need to obtain values from multiple rows, you must use a cursor.
- In a REXX procedure, you must enclose an LSQL command in quotation marks if it contains special characters (such as an equal sign).

When the Commands Take Effect

LSQL commands take effect immediately if you issue them from a console.

To make a command take effect automatically at start-up time, specify it in the CA-L-Serv parameter data set.

Command Output

The SQL server returns several elements after executing an LSQL command. The type of information you receive depends on where you issue the command:

- If the command is issued from a TSO terminal, the following occurs:
  - The returned values, any error messages, or both are sent to the terminal.
  - No return codes are set.
- If the command is issued from a batch job, the following occurs:
  - The job step return code is set with the SQL server return code.
  - The output from the command is sent to SYSPRINT.
- If the command is issued from a CLIST, the following occurs:
  - Return codes are set in the &LASTCC variable.
  - If the return code is 0, 4, or 8, the SQL server also assigns a value to the SQLCODE variable.
  - The specified host variables are set to the returned values; otherwise, a variable named by the SQL server is set to the returned value.
  - Error messages are sent to the terminal.
- If the command is issued from a REXX procedure, the following occurs:
  - Return codes are set in the RC variable in REXX.
  - If the return code is 0, 4, or 8, the SQL server also assigns a value to the SQLCODE variable.
  - The specified host variables are set to the returned values; otherwise, a variable named by the SQL server is set to the returned value.
  - Error messages are sent to the terminal.

For more information on the information returned by an LSQL command, see the chapter "CA-L-Serv" in the Administration Guide.

ALTER TABLE Statement

The ALTER TABLE statement adds a column to or removes a column from an existing database table.

Syntax

```
LSQL ALTER TABLE tablename [[ADD COLUMN] [ADD_COLUMNcolname] [datatype [[UPPER CASE]
[NOT NULL ] [DEFAULT value]]] [DROP COLUMN] [colname]]
```

ADD COLUMN

The clause used to add a new column to an existing table.

colname

The 1- to 18-character name of the column you are adding or dropping.

datatype

The type of data the column can store. The data type can be one of the following:

CHAR(*length*) - character data, with *length* being the maximum number of characters. A column of this type must be between 1 and 32000 bytes in length.

DATE - a date indicator of the form *yyyy-mm-dd*; for example, 2001-08-31.

DECIMAL(*nn*,*nn*) - decimal data, with the maximum number of digits being 15.

DOUBLE PRECISION - approximate numeric data, 8-byte length.

FLOAT(*nn*) - approximate numeric data of variable length; mantissa 2-16 digits, exponent range e-60 to e60.

HEX(*length*) - hexadecimal data, with *length* as the maximum number of hexadecimal bytes.

INTEGER - 32-bit integer data, with a maximum value of 2147483647.

REAL - approximate numeric data, 4-byte length.

SMALLINT - 16-bit integer data with a maximum value of 32767.

TIME or TIME(*nn)* - a time indicator of the form *hh:mm:ss*; for example, 13:21:53.

TIMESTAMP or TIMESTAMP(*nn)* - a date/time indicator, the format being a combination of the DATE and TIME formats.

Note: The DATE, TIME, and TIMESTAMP data types are stored as unsigned packed decimal numbers. When inserting, updating, deleting, or searching for these values, you must specify the data type along with the value. For example:
WHERE CLOSE_DATE = DATE '2001-08-31'

DEFAULT *value*

The value to be set for this column if an INSERT statement does not provide one. The default value can be either a character string or a numeric value; however, it must be compatible with the data type of the column.

DROP COLUMN

The clause used to drop a column from an existing table.

NOT NULL

Indicates that the column cannot contain a null value.

tablename

The name of the relational table to which you are adding or from which you are dropping a column.

UPPER CASE

Converts entries in the column to upper case characters.

Notes:

- Issue an ALTER TABLE statement with an ADD COLUMN clause to add a column to an existing relational table.
- Issue an ALTER TABLE statement with a DROP COLUMN clause to drop a *non-primary key* column from an existing relational table; you cannot drop a *primary key* column from an existing table.
- You cannot add a new column to and delete an existing column from a table within the same ALTER TABLE statement.
- You can define up to 250 columns per table.
- You can use host variables for *tablename* or *colname*.
- If you do not specify a value for the DEFAULT parameter for a new column, the column will contain binary zeros.
- When a column is added, it is positioned to the right of the right-most column.
- When a column is dropped, the columns to the right of the dropped column are shifted to the left by the number of bytes formerly occupied by the dropped column.

Example

To add a 2-character column called RET_CODE to the APPLICATIONS table, issue this LSQL command:

```
LSQL ALTER TABLE APPLICATIONS ADD COLUMN RET_CODE CHAR(2)
  DEFAULT   '0'
```

CLOSE Statement

The CLOSE statement closes a cursor. It is important to close a cursor to release resources.

Syntax

```
LSQL CLOSE cursorname
```

cursorname

The name of the cursor you are closing.

Note: The LSQL command that executes the CLOSE statement must be issued from a CLIST or from a REXX procedure.

Example

To close the cursor named CURSTAT, issue this LSQL command from a CLIST:

```
LSQL CLOSE CURSTAT
```

CREATE TABLE Statement

The CREATE TABLE statement defines a new VSAM file to the SQL server as data in a relational table.

Syntax

```
LSQL CREATE TABLE tablename IN DATABASE (ddname,offset[ PREFIX(xx)])
        (colname datatype [[PRIMARY KEY] [UPPER CASE] [NOT NULL]
        [DEFAULT value]])...
```

colname

The 1- to 18-character name of a column that you are defining in this table. Define columns in the left-to-right order in which the data occurs in the VSAM file.

datatype

The type of data the column can store. The data type can be one of the following:

CHAR(*length*) - character data, with *length* being the maximum number of characters. A column of this type must be between 1 and 32000 bytes in length.

DATE - a date indicator of the form *yyyy-mm-dd*; for example, 2001-08-31.

DECIMAL(*nn*,*nn*) - decimal data, with the maximum number of digits being 15.

DOUBLE PRECISION - approximate numeric data, 8-byte length.

FLOAT(*nn*) - approximate numeric data of variable length; mantissa 2-16 digits, exponent range e-60 to e60.

HEX(*length*) - hexadecimal data, with *length* as the maximum number of hexadecimal bytes. A column of this type must be between 1 and 256 bytes in length.

INTEGER - 32-bit integer data, with a maximum value of 2147483647.

REAL - approximate numeric data, 4-byte length.

SMALLINT - 16-bit integer data with a maximum value of 32767.

TIME or TIME(*nn)* - a time indicator of the form *hh:mm:ss*; for example, 13:21:53.

TIMESTAMP or TIMESTAMP(*nn)* - a date/time indicator, the format being a combination of the DATE and TIME formats.

Note: The DATE, TIME, and TIMESTAMP data types are stored as unsigned packed decimal numbers. When you are inserting, updating, deleting, or searching for these values, you must specify the data type along with the value; for example:

WHERE CLOSE_DATE = DATE '2001-08-31'

ddname

The ddname of a VSAM file that contains the data that you want to map to a relational table.

DEFAULT *value*

The default value to be set for this column if an INSERT statement does not provide one. The default value can be either a character string or a numeric value; however, it must be compatible with the data type of the column.

IN DATABASE

Allows you to specify the VSAM file and the data within it that is to be contained in the relational table.

NOT NULL

Indicates that the column cannot contain a null value.

offset

The column in the VSAM data set where the data that you want to map begins. For example, if the third byte of data in the VSAM record is to be the first byte of data in the relational table, the offset is 3.

Default: 1

PREFIX(*xx*)

The 2-character prefix that identifies the table a record belongs to when a VSAM file contains records for more than one table. The prefix you enter here becomes the first 2 bytes of the VSAM key field.

PRIMARY KEY

Designates a column as the primary key for this table. The primary key parallels the traditional concept of keyed files and improves performance in your relational tables.

You can designate more than one column as a primary key. To do this, either type PRIMARY KEY beside each column definition or type in the column definitions and then type PRIMARY KEY(*firstcol-lastcol*) at the end of the column definitions. (The *firstcol* and *lastcol* variables represent the first and last columns of the primary key.)

tablename

The 1- to 18-character name of the table.

UPPER CASE

Converts entries in the column to upper case characters.

Notes:

- You can define up to 250 columns per table.
- Each record in the VSAM data set becomes a row in your relational table.
- For each column you define, include separate column definition clauses in the CREATE TABLE statement. Separate the column definition clauses with commas.
- You can use host variables for *tablename* or *colname*.
- If you do not specify a DEFAULT value and data is not provided for a column, binary zeros are inserted.

Example

To define a table named RETCODES that contains four columns, (with the APPLICATION and USER_ID columns as the primary key), issue the following LSQL command:

```
LSQL CREATE TABLE RETCODES IN DATABASE (LDMSQL,1)
   (APPLICATION CHAR(15) PRIMARY KEY,
   USER_ID CHAR(8) PRIMARY KEY,
   STATUS CHAR(4) DEFAULT 'DOWN',
   RET_CD CHAR(2))
```

DECLARE CURSOR Statement

The DECLARE CURSOR statement defines a cursor. Include a SELECT statement to establish the location in a table where the cursor is to appear.

```
LSQL DECLARE cursorname CURSOR FOR selectstatement
```

cursorname

The 1- to 18-character name of the cursor that you are defining.

selectstatement

The statement that contains the selection criteria for the cursor. See “SELECT Statement” for more information.

Note: The LSQL command that executes the DECLARE CURSOR statement must be issued from a CLIST or from a REXX procedure.

Example

To define a cursor named STATDOWN that selects the application ID, update time, and status from the APPLICATIONS table when the status is DOWN, issue this LSQL command from a CLIST:

```
“LSQL DECLARE STATDOWN CURSOR FOR
   SELECT APPL_ID, UPDATE, STATUS FROM APPLICATIONS
   WHERE STATUS='DOWN'”
```

DELETE FROM Statement

The DELETE FROM statement deletes one or more rows that meet the specified criteria on a SELECT clause. When using a cursor, DELETE FROM deletes the row currently being processed.

Syntax

```
LSQL DELETE FROM tablename [WHERE criteria]
```

criteria

Search criteria. When using a cursor, the criteria on a WHERE clause must include CURRENT OF *cursorname,* which causes the function to be performed on the current row being processed in the *cursorname* operation. (See the example that follows.)

tablename

The name of the table from which you are deleting.

WARNING! If you use the DELETE FROM statement for a table and do not include a WHERE clause, all rows from the table are deleted. Use the DELETE FROM statement with extreme caution.

Notes:

- When the DELETE FROM statement is used with a cursor, it must be issued from a CLIST or from a REXX procedure.
- You can use a host variable for *tablename*.

Examples

- To delete rows from the APPLICATIONS table when those rows were last updated on 02/13/02, issue this LSQL command:

  ```
  LSQL DELETE FROM APPLICATIONS WHERE UPDATE=DATE '2002-02-13'
  ```

  Note that the data type definition precedes the literal date value.

- To delete the current row being processed by a cursor named STATDOWN, issue this LSQL command from a CLIST:

  ```
  LSQL DELETE FROM APPLICATIONS WHERE CURRENT OF STATDOWN
  ```

DROP TABLE Statement

The DROP TABLE statement removes the SQL dictionary definition for a relational table, making it impossible to access the data using SQL. However, the VSAM data set containing the data is not affected by the DROP TABLE statement.

By issuing a CREATE TABLE command for the relational table, the data is once again accessible using SQL.

Syntax

```
LSQL DROP TABLE tablename
```

tablename

The name of the table to be removed.

Note: You can use a host variable for *tablename*.

Example

To remove the APPLICATIONS table, issue this LSQL command:

```
LSQL DROP TABLE APPLICATIONS
```

FETCH Statement

Issue a FETCH statement to retrieve the values in the row currently occupied by a cursor. Usually this statement is executed within a loop controlled by the SQLCODE variable so that the SQL server processes each row accessed by a cursor and exits the loop when the end of the table is detected.

Syntax

```
LSQL FETCH cursorname INTO (hostvarlist )
```

cursorname

The name of the cursor from which you are retrieving values.

hostvarlist

A set of host variable names in which to store the selected column values.

Specify a name for each column defined in the SELECT clause of the DECLARE CURSOR statement in the appropriate order.

Note: If you do not specify a host variable for each column defined in the cursor, the statement executes and a warning message is returned.

Example

To retrieve the values from the example cursor called STATDOWN, issue this LSQL command from a CLIST:

```
LSQL FETCH STATDOWN INTO (:APPLID,:UPDATE,:STATUS)
```

INSERT Statement

The INSERT statement inserts new rows into a table. You can insert as many rows as needed.

Syntax

```
LSQL INSERT INTO tablename [(columnlist)][[VALUES(list)] [querystatement]]
```

columnlist

The column names that store the values specified with valuelist. If you do not specify columnlist values, the SQL server stores the values into the columns in the order in which they were defined on the CREATE TABLE statement for this table.

querystatement

A query statement that retrieves the values to be inserted into the table. This can be any valid search criteria.

tablename

The name of the table into which you are inserting a row.

VALUES(*list*)

The column values for the new row. A value can be a character string, a numeric string, or a host variable. Values in the list are separated by commas.

Notes:

- You can use host variables for *columnlist* or *tablename*.
- The order of the columns in *columnlist* must match the order of the values in *list*. You do not need to match the actual order of the columns in the table with *columnlist*. As long as the order of *columnlist* and *list* match each other, the values are inserted into the columns properly.

Examples

- Suppose you want to insert a row into the APPLICATIONS table. This row will have the value APPL29 in the APPL_ID column, TSOUSR29 in the USER_ID column, 2002-03-13 in the UPDATE column, and UP in the STATUS column. To do this, you could issue the following LSQL command:

  ```
  LSQL INSERT INTO APPLICATIONS (APPL_ID,USER_ID,UPDATE,STATUS)
       VALUES ('APPL29','TSOUSR29', DATE '2002-03-13','UP')
  ```

  Note: The DATE data type definition precedes the literal date value. See the description of the Select Statement for more details about the DATE, TIME, and TIMESTAMP data types. Also, that the column values match the order in which the columns appear in the table, so the column list could have been left out.

- If you wanted to add a row with only the APPL_ID and STATUS columns, you could use the following statement:

  ```
  LSQL INSERT INTO APPLICATIONS (APPL_ID, STATUS) VALUES   ('APPL29', 'UP')
  ```

- Now, suppose there is a table called NEWAPPS that has the identical column structure of the APPLICATIONS table. You might want to add a row to the APPLICATIONS table for each of the new applications that has been altered by users TSOUSR1, TSOUSR2, and TSOUSR8. In this case, you could substitute the VALUES clause with a query statement, as follows:

  ```
  INSERT INTO APPLICATIONS SELECT * FROM NEWAPPS WHERE USER_ID
    IN ('TSOUSR1', 'TSOUSR2', 'TSOUSR8')
  ```

OPEN Statement

The OPEN statement opens a cursor. When the OPEN statement is issued, the select criteria for the cursor (specified with the DECLARE CURSOR statement) is immediately executed.

Syntax

```
LSQL OPEN cursorname
```

cursorname

The name of the cursor you are initiating (previously defined using the DECLARE CURSOR statement).

Note: The LSQL command that executes the OPEN statement must be issued from a CLIST or from a REXX procedure.

Example

To open the cursor named STATDOWN, issue this LSQL command from a CLIST:

```
LSQL OPEN STATDOWN
```

SELECT Statement

The SELECT statement extracts data from a relational table that meets the criteria you specify. You can use the SELECT statement by itself or within another statement (then called a subquery).

Syntax

```
LSQL SELECT columnlist|*asterisk [INTO (hostvarlist)] FROM tablename
 [WHERE criteria] [GROUP BY criteria [HAVING criteria]] [ORDER BY criteria]
```

*****

Selects all columns in a table.

columnlist

Selects values only from the columns you name. Each column name must be separated by a comma.

criteria

The search criteria that you are using to match values in the specified columns.

FROM

Clause used to specify the *tablename* from which data is being selected.

GROUP BY

Clause used to summarize multiple rows of data into single rows based on the specified *criteria*.

HAVING

Clause used, only with a GROUP BY clause, to further limit the output of the statement based on the specified *criteria*.

hostvarlist

The host variables in which to store the selected column values. In a cursor declaration, do not use this parameter since this function is accomplished with the FETCH statement.

ORDER BY

Clause used to specify the order of the statement output.

tablename

The name of the relational table from which you are selecting.

Notes:

- You can use host variables for *columnlist* or *tablename*.
- You can issue the SELECT statement from two different environments:

- From a TSO terminal; selected rows are displayed on the terminal.
- From a REXX procedure or a stand-alone CLIST (that is, not part of a cursor declaration). When you use SELECT without using a cursor and you insert the returned values into variables, the variables would receive the values in only the first row that matched the search criteria.

- When using a cursor issued from a REXX procedure or a CLIST, the cursor points to each row that matches the search criteria, one row at a time. With this method, you would normally use a WHILE loop, with the SQLCODE variable controlling the duration of the loop, to receive values from all rows of the table that match the search criteria.

Examples

- To select all values in all columns of the APPLICATIONS table, issue this command:

  ```
  LSQL SELECT * FROM APPLICATIONS
  ```

- To select all of the application IDs from the APPLICATIONS table, issue this command:

  ```
  LSQL SELECT APPL_ID FROM APPLICATIONS
  ```

- To select all of the application IDs whose status is DOWN, issue this command:

  ```
  LSQL SELECT APPL_ID FROM APPLICATIONS WHERE STATUS='DOWN'
  ```

- To include a SELECT statement within a DECLARE CURSOR statement that includes the APPL_ID and STATUS columns and points to rows where the status is DOWN, issue this command:

  ```
  LSQL DECLARE CURSOR STATDOWN FOR SELECT APPL_ID, STATUS FROM APPLICATIONS WHERE
  STATUS = 'DOWN'
  ```

UPDATE Statement

The UPDATE statement updates values in selected columns in a table. You can use this statement with a cursor.

Syntax

```
LSQL UPDATE tablename SET colname=value... [WHERE criteria]
```

colname

The name of the column containing values to be updated.

criteria

The criteria on a WHERE clause can be any valid search criteria. However, when using a cursor, the criteria on a WHERE clause must include CURRENT OF *cursorname,* which causes the function to be performed on the current row being processed in the *cursorname* operation. (See the example that follows.)

tablename

The name of the table you are updating.

value

The value that you are inserting into this column. The value can be a character string, numeric string, or host variable.

Note: You can use host variables for *colname* or *tablename*.

Examples

Suppose you want to search the APPLICATIONS table for all applications with a status of DOWN and change that status to UP. Issue the following LSQL command:

```
LSQL UPDATE APPLICATIONS SET STATUS ='UP' WHERE STATUS ='DOWN'
```

To set the value of the STATUS column to UP for the current row when using a cursor called STATDOWN, issue the following LSQL command:

```
LSQL UPDATE APPLICATIONS SET STATUS='UP' WHERE CURRENT OF STATDOWN
```

Chapter 12: Environment Variables

This chapter describes the Environment Variables for Event Management, Calendar Management, Common, and Agent Technology.

This section contains the following topics:

Event Management

CA_CAIDEBUG(Event Management Debug)

Applies to UNIX/Linux and z/OS USS

Used internally for debug flag.

CA_OPERA_NODE (Console Daemon Node)

Applies to Windows and NetWare

Identifies the machine (node) where the Event Management service provider (CAOPRDMN) resides.

Size/Type: 1-15/alphanumeric bytes

Default: Assigned during installation

CA_OPR_AUTH_LIST (Users Authorized to Run Commands)

Applies to Windows and UNIX/Linux

Identifies the user IDs who have authorization to issue Event Management commands, acknowledge held messages, and reply to messages awaiting reply (WTOR) when security is not active. If security is active, this value is ignored.

Separate multiple user IDs with commas. Node names or domain names can be used as part of the user ID. The following wildcards can be used in node names, domain names, and user IDs:

* (asterisk)	For 0 or more characters
? (question mark)	For any single character

For node or domain information not supplied, an asterisk (*) is assumed. For example, each of the following designations is synonymous:

Scott
Scott@*
Scott@**
*\Scott
\Scott@
\Scott@

Syntax

```
user@node.
```

Example

Administrator, Ranger@Mars, Zip*@usil, caunint@*

Default: User ID under which CA NSM was installed (caunint) at the local machine (for Windows) *nt_manager*\ADMINISTRATOR

CA_OPR_BQ_ACTTIMEOUT (Action Back Quote Process Timeout (seconds))

Applies to Windows only

Specifies the number of seconds Event Management waits for the completion of a process that was created to run the command contained within a pair of back quotes in the Message Record Action Text field. If the process does not complete within this interval, the text within the back quotes is not used as a command, but is evaluated as is.

Since Message Record Action back quote substitution is performed asynchronously by Event Management, the wait interval does not have the same performance impact that is associated with message record processing.

Note: Message matching is performed synchronously by Event Management. Therefore, any commands within back quotes that require lengthy processing may impact system performance considerably. For additional information, see the CA_OPR_BQ_MSGTIMEOUT variable.

Range: 0 to 120

Default: 60

CA_OPR_BQ_MSGMATCH (Back Quote Processing for Message UDATA)

Applies to Windows only

If this option is set to YES, when Event Management performs real-time message matching, it considers any text within a pair of back quotes in the message record User Data field to be a command.

Event Management passes that command as a parameter to a created CMD.EXE process. This command text is then substituted by any stdout output produced by that command. If this option is set to NO, Event Management does not use text within back quotes in the message record User Data field as a command during message matching.

Note: Message matching is performed synchronously by Event Management. Therefore, any commands within back quotes that require lengthy processing may impact system performance considerably. For additional information see the CA_OPR_BQ_MSGTIMEOUT variable.

Valid Values: YES or NO

Default: Yes

CA_OPR_BQ_MSGTIMEOUT (Message Back Quote Process Timeout (seconds))

Applies to Windows only

Specifies the number of seconds Event Management waits, during real-time message matching, for a process that was created to run a command contained within a pair of back quotes in the message record User Data field to complete. If the process does not complete within this interval, the User Data text within the back quotes is not used as a command, but is evaluated as is.

Before changing the value of this field, you may want to consult with your systems administrator or CA Technologies Support personnel. Since message matching is performed synchronously by Event Management, increasing this interval value may considerably impact system performance under some conditions.

Range: 0 to 120

Default: 5

CA_OPR_CASDB

Applies to z/OS only

This environment variable specifies whether the optional CA Datacom/AD database will be used. The database is required only for using Calendars and Message Actions. If Y(yes) is specified, the OPR processes will not start if the database is not running.

Valid Values: Y/N

Default: N

CA_OPR_CASE (Message Matching With Case Sensitivity)

Applies to Windows only

When set to NO, case is ignored in all message record matching.

Note: This variable sets only the default matching method. You can override this value for individual message records.

Valid Values: Yes or No

Default: Yes

CA_OPR_DB_INTERVAL (# of seconds between database load retries)

Applies to Windows only

The number of seconds to wait between database operation (dbload/select) retries for the Event Management database.

Range: 0 to 1000

Default: 60

CA_OPR_DB_RETRIES (# of Database Load Retries)

Applies to Windows only

Specifies the number of times a database load operation against the Event Management database can be retried. If the Event Management policies database is not available when the Event Manager daemon is started, or when the opreload command is issued, Event Manager will retry loading the policies database as many times as specified.

Range: 0 to 1000

Default: 60 times

CA_OPR_DEFAULT_RUNID (Default User ID for Running Commands)

Applies to Windows only

This value is used when the CA_OPR_RUNAS_CMD variable is set to DEFAULT or when the user ID cannot be determined. This variable identifies the user ID under which commands or command actions are executed. If this user ID is not defined through Security Management, a password must be provided within the CA_OPR_DEFAULT_RUNPW variable.

Default: Guest

CA_OPR_DEFAULT_RUNPW (Default Password for Running Commands)

Applies to Windows only

This value is used when the CA_OPR_RUNAS_CMD variable is set to DEFAULT or when the user ID cannot be determined. This variable identifies the password associated with the user ID specified by the CA_OPR_DEFAULT_RUNID variable.

Note: If the default user ID is not defined through Security Management, you must specify a password in the variable.

Default: None

CA_OPR_DFLT_NODE (Default message node for DEFINE)

Applies to Windows only

Identifies the machine (node) to use as a default if none is specified during the definition of an Event Management message record.

Size/Type: 1-15/alphanumeric bytes

Default: * (asterisk) indicates all machines in the Enterprise Management domain

CA_OPR_DSB (Copy of MSG_DB)

Applies to Windows

Specifies the full path name of the file that contains the binary image (DSB) of the Event Management database.

Default: %CAIGLBL0000%\LOGS\CAOPR.DSB

CA_OPR_FORK_LIST (Actions to run in a thread)

Applies to Windows only

This field is for CA NSM internal use only.

CA_OPR_FREQ_OPT (Control Interval and Frequency for Message Record Matching)

This variable controls the behavior of message record matching for the interval and count fields under frequency on the message record.

INTERVAL_RESET--Counters are reset at the end of the interval. Guarantees that the *n*th message of the same type will match in any given interval.

MATCH_RESET--Counters are reset at the end of the interval, and after a match. Guarantees that the *n*th message of the same type will match in any given interval, and the interval is re-started on every match.

Range: 0-1

Default: 0

CA_OPR_LOG_FLUSH

Frequency of log flush. Controls how many log writes should occur before the actual data is committed to the log file.

Note: For CA Internal use only.

Value: Numeric 1 - 9999

Default: 1

CA_OPR_LOG_SUPMSG (Log suppressed messages)

Applies to Windows only

When this field is set to YES, suppressed messages are sent to the log and displayed in the console GUI with an icon marker.

When this field is set to NO, suppressed messages are discarded.

Valid Values: YES/NO

Default: YES

CA_OPR_MAX_GOTO (Max # of GOTO actions executed per match)

Applies to Windows only

Controls how many GOTO actions can be executed per message match. Prevents infinite loops such as:

100 SENDOPR xxxxx
200 GOTO 100

Range: 1 to 9999

Default: 100

CA_OPR_MAX_THREADS (Max # of threads)

Applies to Windows only

Specifies the maximum number of threads available to Event Management for processing message actions.

Before changing the value of this field, you may want to consult with your system administrator or CA Technologies Support personnel. Increasing the size of this field may cause the use of excessive system resources.

Range: 1 to 1000

Default: 128

CA_OPR_MAX_WAIT (Max # of seconds to wait for available thread)

Applies to Windows only

Specifies the number of seconds Event Management will wait for a thread to process a message action. If a thread does not become available in that time, the message is logged although no action is taken.

Before changing the value of this field, you may want to consult with your systems administrator or CA Technologies Support personnel. Increasing the size of this field may cause the use of excessive system resources.

Range: 1 to 3600

Default: 300

CA_OPR_RDR_AGE (Windows Log Reader Max Age of Record to Send (hh:mm))

Applies to Windows only

Specifies the maximum age of Windows log records that will be sent to the Event Management console log. If any Windows event log record is older than the age specified, in hours and minutes, the record will not be sent.

Default: 06:00 (six hours)

CA_OPR_RDR_SAF (Use SAF for Windows LOG rdr)

Applies to Windows only

This variable indicates whether the Windows event log reader will send Windows events to the Store and Forward (SAF) facility in case the Event Manager is not available on the target node.

By default, the event log reader will retry to send the message until it is successful or until a message is too old (see the CA_OPR_RDR_AGE variable).

If you choose YES, Windows messages will be sent to SAF, which will forward them to the console log when the Event Manager becomes available.

Default: No

CA_OPR_REGEX (Message Matching with Regular Expressions)

Applies to Windows only

This variable affects all message record columns except the message ID, which always uses wild card matching.

If set to NO, Event Management uses wild card matching for message records.

If set to YES, Event Management uses POSIX-compliant regular expression matching for message records.

Note: This variable sets only the default matching method. You can override this value for individual message records.

Valid Values: Yes or No

Default: No

CA_OPR_RESOLVE_VAR

Applies to Windows

Determines whether CA NSM resolves environment variables ($ENVVAR and &ENVVAR) in a message received from a remote node.

Valid values: Yes, No

Default: Yes

CA_OPR_RETAIN_LOGS (# of logs to retain)

Applies to Windows, UNIX/Linux, z/OS USS

Specifies the number of CA NSM console log files to be maintained on the server. When the number of log files exceeds this setting, Enterprise Management will remove log files (the oldest first) and their respective index files (.IDX and .LDX suffixes) until the threshold has been reached. If 0 is specified, no log files are purged. You are advised to periodically review the files in the directory Enterprise Management-install path/LOGS, and clean up as necessary. For z/OS USS, this environment variable is set in the $CAIGLBL0000/opr/scripts/envusr script.

Range: 0 to 1000

Default: 0

CA_OPR_RUNAS_CMD (User Context for Running Commands)

Applies to Windows only

Identifies the environment under which a command is executed through Event Management.

Valid Values:

HOST

The HOST ID

ORIGIN

The user ID that issued the command.

Note: Security must be active and the user ID must be defined to Security.

DEFAULT

The default user ID specified by the CA_OPR_DEFAULT_RUNID variable.

Note: If the default user ID is not defined through Security, a password must be provided in the CA_OPR_DEFAULT_RUNPW variable.

Default: HOST

CA_OPR_SAF

Applies to UNIX/Linux, and z/OS USS

Determines if the Event Management Store and Forward facility is started. $CAIGLBL0000/opr/scripts/envset sets this value based on user input during the Event Management installation. If the value is set to Y, oprsafd is started.

This is reserved for internal use only.

CA_OPR_SAF_CONFIG (SAF Config File)

Applies to Windows only

Specifies the name and location of the configuration file used by the Store and Forward (SAF) function.

Default: %CAILOCL0000%\SAF.CFG

CA_OPR_SAF_MAX_OPEN (Max # of Open SAF Files)

Applies to Windows only

Specifies the maximum number of Store and Forward (SAF) files that can be open.

Range: 5 to 500

Default: 20

CA_OPR_SAF_ROOT (SAF Root)

Applies to Windows and z/OS USS

Specifies the location of the various files created by the Store and Forward (SAF) function.

Default: $CAIGLBL0000/opr/saf/

CA_OPR_SAF_SCAN_INT (SAF Scan Interval (secs))

Applies to Windows only

Specifies the interval, in seconds, between scans or attempts to reconnect to nodes for which Store and Forward (SAF) files (message queues) exist.

Range: 0 to 1000

Default: 60

CA_OPR_TAG (Platform Name)

Applies to Windows

This variable indicates the platform where Enterprise Management is running. On Windows, it is set to WNT. A user can configure it to whatever value is wanted, for example, WNT3.51, HPUX80, and so on.

Default: The platform on which Enterprise Management is installed

CA_OPR_TEST_TRUE_RC

Applies to Windows

Sets the return code for the message action ACTION keyword of TEST when the test result is true.

Valid values: 0 or 1

Default: 0

CA_OPR_TRACE (OPR Trace 0-2)

Applies to Windows only

This field is for use by CA Support personnel only.

CA_OPR_USEDB (Load from database)

Applies to Windows

Specifies whether Event Management should use the actual Event Management database as opposed to its binary image (DSB) file.

Valid Values: YES or NO

Default: YES

CA_OPR_WV_STATUS (Notify of WorldView Object's Status Change)

Applies to Windows only

When set to YES, you will see messages in the console log about WorldView objects' status changes.

Valid Values: Yes or No

Default: No

CA_OPR_ZOSDB

Applies to z/OS only

This environment variable specifies if the optional CA Datacom/AD database will be used. The database is required only for using Calendars and Message Actions. If Y(yes) is specified, the opr processes will not start if the database is not up and running.

Valid Values: Y/N

Default: N

CA_SAF_TRACE (SAF Trace 0-2)

Applies to Windows only

This field is for use by CA Support personnel only.

CA_TRAPD_CONFIG (Trap Daemon Config File)

Applies to Windows only

Identifies the name of the configuration file that tells the CA trap daemon which SNMP traps to ignore.

Default: %CAILOCL0000%\CAIUSER\CATRAPD.CFG

CAI_CAMSGF_OPRDIRECT

Applies to UNIX/Linux and z/OS USS

Determines whether CA NSM messages (CAmsgf messages) are sent directly to the Event Manager. If this value is set to Yes, CA NSM messages are sent directly to the Event Manager and not to syslog. $CAIGLBL0000/scripts/global_vars sets it to a user-selected value (selected by the user during the installation of Event Management).

CAI_CONLOG (Console Files Directory)

Specifies the name of the directory where the Event Management service provider (CAIOPR daemon) writes its log files.

Each day, these files are created:

Applies to Windows only

yyyymmdd.LOG The console log for the specified day

yyyymmdd.IDX The index for held and WTOR messages in the console log.

yyyymmdd.LDX The index file for fast access.

Note: Both of the files contain binary data and are not intended to be browsed.

Default: %CAIGLBL0000%\LOGS

Applies to UNIX/Linux and z/OS USS

opano.yyyymmdd Annotate file

opldx.yyyymmdd Log Index

oplog.yyyymmdd The console log for the specified day.

opndx.yyyymmdd The index for held and WTOR messages in the console log.

Note: ALL of these files contain binary data and are not intended to be browsed through cat or any other UNIX utility.

Default: $CAIGLBL0000/opr/logs

CAI_NODENAME_DEBUG (Event Management Debug)

Applies to z/OS only

Used internally for debug flag.

CAI_OPR_CONFIG (CAIOPR Daemon)

Applies to UNIX/Linux only

The caiopr daemon writes the file opr.pid to the directory $CAI_OPR_CONFIG/nodename. This file prevents concurrent execution of more than one copy of the caiopr daemon.

Default: $CAIGLBL0000/opr/config

CAI_OPR_DBNAME (Opera Database Name)

Applies to Windows only

Specifies the name of the Event Management database.

Note: Do not alter this value unless instructed to do so by CA Technologies Support personnel.

Default: CAIOPRDB

CAI_OPR_OLDEVT (Read Old Events)

Applies to Windows only

Indicates whether Windows events, which occurred before Event Management was started, will appear in the Event Management console log.

If Yes is specified, we recommend that you clear the current Windows log of any extraneous events before starting Event Management. Also, the existing Windows events will be assigned the time and date stamp of when the Event Management service was started.

Default: No

CAI_OPR_REMOTEDB

Applies to UNIX/Linux and z/OS USS

Used by the Event Manager daemon in the CA Common Services environment. $CAIGLBL0000/scripts/envset, which is generated dynamically by fwsetup, sets this value to the host name of the CA Common Services manager node.

This is reserved for internal use only.

CAI_WTOR_NODE (Default Node for cawtor and careply Commands)

Applies to UNIX/Linux and z/OS USS

Identifies the default node to use when no node name is specified for a cawtor command, careply command, or a daemon that uses the underlying cawtor or careply functionality.

Specify this variable by supplying the default node name in a file named CAIGLBL0000/opr/config/`uname-n`/cai_wtor_node. If this file is found at startup time, the CAI_WTOR_NODE variable will be set for all users who invoke CAIGLBL0000/scripts/envset.

CAIACTOPRAG (Event Management Agent Active)

Applies to Windows only

The Event Management function consists of the Event Manager which receives, processes, and logs all messages (events), and the Event Management Agent, which reads the Windows event log file and sends the events to the Event Manager for processing. This variable indicates whether the Event Management Agent is activated when Enterprise Management starts.

Default: No

CAIACTOPRSV (Event Management Active)

Applies to Windows only

The Event Management function consists of the Event Manager, which receives, processes, and logs all messages (events), and the Event Management Agent, which reads the Windows event log file and sends the events to the Event Manager for processing. This variable indicates whether the Event Manager is activated when Enterprise Management starts.

Default: No

CAIACTSAFSV (Store and Forward Active)

Applies to UNIX/Linux and Windows

Indicates whether the Store and Forward capability of Event Management is activated when Enterprise Management starts. Store and Forward stores messages that temporarily cannot be sent and tries periodically to resend them. Store and Forward is a global setting.

Default: No

CAIACTTRAPD (SNMP Trap Server Active)

Applies to UNIX/Linux and Windows

Indicates whether the SNMP Trap Server capability of Event Management is activated when Enterprise Management starts. SNMP Trap receives Simple Network Management Protocol (SNMP) messages and integrates them into the Event Management console display.

Default: No

CAICATD0000 (Default Enterprise ID for CATRAP)

Applies to UNIX/Linux only

Identifies the default enterprise ID used by the catrap command.

Default: 1.3.6.1.4.1.791

CAICATD0001 (Default Listening Port for CATRAP Daemon)

Applies to z/OS USS

Identifies the TCP/IP port that the CATRAP daemon will listen on.

Default: 161

CAICATD0003 (CAICCI Connect Retry Interval)

Applies to UNIX/Linux only

Specifies the CAICCI connect retry interval, in minutes, used by the CATRAP daemon.

Default: 1

CAICATD0004 (CATRAP Connect Retry Attempts)

Applies to UNIX/Linux only

Specifies the number of times the CATRAP daemon should attempt a connect retry.

Default: 60

CAICATD0005 (CAMIBWALK MIB Path)

Applies to UNIX/Linux only

Identifies the filename of the MIB path used by the camibwalk command.

Default: $CAIGLBL0000/snmp/dat/mib.txt

CAICATD0006 (Node Name of SNMP Network Monitor)

Applies to UNIX/Linux only

Used if the SNMP Network Monitor, such as SunNet Manager, is active and must process cooperatively with CA NSM. If so, this variable contains the node name of the machine on which SunNet Manager executes.

Default: none

CAICATD0007 (SNMP Management X-Station IP Address)

Applies to UNIX/Linux only

This is the IP address for the SNMP Management x-station, which is defined during the install for SNMP.

Default: Current node.

CAICATD0008 (SNMP Trap Destination File Pathname)

Applies to UNIX/Linux only

This is the pathname of snmp.conf, which is the trap destination file for SNMP. It is defined during the install for SNMP.

Default: Depends on operating system.

CAIOPR_DEBUG (Event Management Debug)

Applies to UNIX/Linux and z/OS USS

Used internally for debug flag.

EMEVT_DEBUG (Event Management Debug)

Applies to z/OS only

Used internally for debug flag.

EMEVT_EXIT_ACTPOST (ActionPost Exit Control)

Applies to UNIX/Linux and Windows

Specifies whether the Action Post-processing exit is enabled. User exits are user-defined programs created using the CA Software Development Kit (SDK).

Valid Values: ON (enabled) or OFF (disabled)

Default: OFF

EMEVT_EXIT_ACTPRE (ActionPre Exit Control)

Applies to UNIX/Linux and Windows

Specifies whether the Action Preprocessing exit is enabled. User exits are user-defined programs created using the CA Software Development Kit (SDK).

Valid Values: ON (enabled) or OFF (disabled)

Default: OFF

EMEVT_EXIT_DLL (Name of User Exits Library)

Applies to UNIX/Linux and Windows

Specifies the name of the library that contains user exits for Event Management. If you do not provide a full path, the standard system search path will be used. For example, EMEVT77.DLL. If the library is not found, all exits will be disabled.

User exits are user-defined programs created using the CA Software Development Kit (SDK).

Default: None is specified

EMEVT_EXIT_LOGPOST (LogPost Exit Control)

Applies to UNIX/Linux and Windows

Specifies whether the Log Post-processing exit is enabled. User exits are user-defined programs created using the CA Software Development Kit (SDK).

Valid Values: ON (enabled) or OFF (disabled)

Default: OFF

EMEVT_EXIT_LOGPRE (LogPre Exit Control)

Applies to UNIX/Linux and Windows

Specifies whether the Log Pre-processing exit is enabled. User exits are user-defined programs created using the CA Software Development Kit (SDK).

Valid Values: ON (enabled) or OFF (disabled)

Default: OFF

EMEVT_EXIT_MAX_ERRORS (Number of Errors Allowed Per Exit)

Applies to UNIX/Linux and Windows

Specifies the number of errors allowed per user exit for Event Management. The exit will be disabled when more than the specified number of errors occur.

User exits are user-defined programs created using the CA Software Development Kit (SDK).

Default: 3

EMEVT_EXIT_MSGPOST (MessagePost Exit Control)

Applies to UNIX/Linux and Windows

Specifies whether the Message Post-processing exit is enabled. User exits are user-defined programs created using the CA Software Development Kit (SDK).

Valid Values: ON (enabled) or OFF (disabled)

Default: OFF

EMEVT_EXIT_MSGPRE (MessagePre Exit Control)

Applies to UNIX/Linux and Windows

Specifies whether the Message Preprocessing exit is enabled. User exits are user-defined programs created using the CA Software Development Kit (SDK).

Valid Values: ON (enabled) or OFF (disabled)

Default: OFF

EMEVT_EXIT_SYSINIT (SysInit Exit Control)

Applies to UNIX/Linux and Windows

Specifies whether the System Initiation exit is enabled. User exits are user-defined programs created using the CA Software Development Kit (SDK).

Valid Values: ON (enabled) or OFF (disabled)

Default: OFF

EMEVT_EXIT_SYSTERM (SysTerm Exit Control)

Applies to UNIX/Linux and Windows

Specifies whether the System Termination exit is enabled. User exits are user-defined programs created using the CA Software Development Kit (SDK).

Valid Values: ON (enabled) or OFF (disabled)

Default: OFF

Calendar Management

CA_CAL_DB_INTERVAL (# of Seconds Between Database Load Retries)

Applies to Windows only

The number of seconds to wait between database operation (dbload/select) retries for the calendar database.

Range: 0 to 1000

Default: 60

CA_CAL_DB_RETRIES (# of Database Load Retries)

Applies to Windows only

The number of times a database operation (dbload/select) is retried for the calendar database.

Range: 0 to 1000

Default: 60

CA_CAL_DSB (Copy of incore Calendars)

Applies to Windows only

Specifies the full path name of the file that contains the binary image (DSB) of the Calendar database.

Default: %CAIGLBL0000%\cal\ca_cal.dsb

CA_CAL_TRACE (Calendar Trace)

Applies to Windows only

For internal use by CA Technologies Support personnel only.

CA_CAL_USEDB (Load From Database)

Applies to Windows only

This variable determines whether calendar management should use the calendar database (YES) or the calendar binary image (DSB) file of the calendar database (NO).

Default: YES

CA_CALENDAR_NODE (Calendar Master Server Node Name)

Applies to Windows only

Identifies the machine (node) that will be considered the master calendar server within the Enterprise Management domain.

Default: Assigned during installation

CACAL_PATH (Calendar Data Path)

Applies to Windows only

Indicates the working directory for the calendar function. If you want to change this value, you must create the new directory and then change the value.

For all Enterprise Management servers running the calendar service subcomponent, this should refer to the same directory on a shared disk.

Default: %CAIGLBL0000%\CAL\DATA

CAI_CAL_REMOTEDB

Applies to UNIX/Linux and z/OS USS

Used by the calendar daemon in CA Common Services to get calendar information. $CAIGLBL0000/scripts/envset, which is generated dynamically by fwsetup, sets the value of this environment variable to the host name of the CA Common Services manager node.

This is reserved for internal use only.

CAICAL0000 (Calendar Debug)

Applies to UNIX/Linux only

Controls whether the built-in debugging options of the Calendar daemon are turned on or off. Intended for use by CA Technologies Support personnel only.

Default: Off

CAICAL0001 (Calendar Daemon CAICCI applid)

Applies to UNIX/Linux and Windows

Identifies the name of the CAICCI server machine that the calendar service provider is accessed through and identifies the calendar demon's CAICCI APPLID. CAICAL0001 is also used by all Enterprise Management functions requesting calendar validation to communicate with the calendar daemon. This value is also used by all Enterprise Management functions that request calendar validation to communicate with the calendar service.

Default: CA_Cal_Chk

CAICAL0002 (Default Database Name)

Applies to UNIX/Linux and Windows

Specifies the name of the database where calendar definitions reside.

Note: Do not alter this value unless instructed to do so by CA Technologies Support.

Default: CAICALDB (Windows)

None (UNIX)

CAICAL0003 (Calendar Lock File)

Applies to UNIX/Linux only

Contains the path name of the Calendar daemon lock file. The lock file, named CA_CalDmon_Lock, prevents concurrent execution of more than one copy of the Calendar daemon. The lock file contains the process ID (PID) of the currently executing Calendar daemon.

Default: $CAIGLBL0000/cal/config/`uname -n`

CAICAL0210 (Calendar Work Days)

Applies to UNIX/Linux

Specifies the number of working days in a week. If not set, Saturday is considered a non-working day. The maximum is 7.

Set Saturday as a working day by issuing this command:

```
export CAICAL0210=6
```

Default: 5

Common

ACCESSDB (Microsoft Access Subdirectory)

Applies to UNIX/Linux and Windows

For UNIX/Linux: Location of report database CAIDB.MDB. Applies only if CAI_PRINT is set to ACCESS.

For Windows: Identifies the subdirectory where Microsoft Access resides.

CA_CAILANGUAGE

Applies to UNIX/Linux

Used by the CA NSM messaging system (CAmsgf) to determine the language. This value is set by $CAIGLBL0000/scripts/global_vars, which is called by $CAIGLBL0000/scripts/envset. Some of Its values are:

enu

English

chs

Simplified Chinese

jpn-sjis

Japanese shift jis

jpn-euc

Japanese euc

This is reserved for internal use only.

CA_CAIMESSAGE (Message Directory)

Applies to UNIX/Linux

Defines the directory where the Enterprise Management message file resides. If you want to change this value, you must create the new directory and then change the value.

Default: $CAIGLBL0000/messages/american (UNIX)

CA_CASE_MIXED (Case of Node for Communication)

Applies to Windows only

This variable is for CA NSM internal use only.

CA_DB (CA-DB Directory)

Applies to UNIX/Linux only

Contains the name of the UNIX file system directory that is the home directory of the cadb login ID.

Default: none

CA_DB_BINSUPPORT (Database binary support)

Applies to Windows only

Turns on binary support for calendar. If the data source supports binary, calendar uses it.

Note: This field is for internal use only. Do not change the value of this field unless so instructed by CA Technologies Support.

CA_DB_DRIVER (Database Driver)

Applies to Windows only

Specifies the type of database source, either TMSDB or ODBC.

Note: This field is for internal use only. Do not change the value of this field unless so instructed by CA Technologies Support.

CA_ENVSH_MAX (Maximum Environmental Names per Key)

Applies to Windows

Lets you choose the maximum number of environmental names per environmental key that can be used by the *ENVSH utilities. Valid values are from 50 to 5000.

Default: 100

CA_EOID2STR (Enterprise OID displayed as)

Applies to Windows only

SNMP trap messages include a field for the enterprise OID (object identifier). The setting controls how the OID is displayed in that field. The valid values are:

NUMBER

Displays the seventh element of the complete enterprise OID, for example, 791.

OID

Displays the entire enterprise OID, for example, 1.3.6.1.4.1.791.

STRING

Displays a MIB string translation of the complete enterprise OID.

NAME

Displays the translated name of the company or organization that owns this enterprise MIB number, for example, Computer.Associates. (The Internet Assigned Numbers Authority-IANA-assigns enterprise MIB numbers to a given organization and reserves those numbers for exclusive use by that organization from that point forward.)

The translation is governed by the enterprise.dat file in the %CAIGLBL0000%\DB directory. You can add your own entries to the file. The layout is as follows:

First column

The numeric enterprise ID (last element of the OID).

Second column

The name of the company that owns the enterprise ID. This string is used when a trap is sent to Event Manager.

For example:

791 Computer.Associates International Greg Jones greg10@usxxxxxx.x.x

Means:

791 Is the enterprise ID for CA

Computer.Associates Is the string used when the trap is sent to Event Manager.

Notes:

Only the first blank-separated word is used in column two. Use a period (.) to connect a company name that has more than one word.

Comment lines begin with a pound sign (#) and are ignored.

Default: NUMBER

CA_NETMSG_NODE (Network Messaging Node)

Applies to Windows only

Use this variable to direct messages to a node when timely intervention can occur on message dialogs that require a response from an operator.

CA_PROTEPINDX (Transport Protocol Endpoint Index)

Applies to Windows only

Identifies the transport protocol installed on the Windows machine. The three possible settings are:

Indicates NetBEUI

Indicates TCP/IP

Indicates IP-UDP

Default: 0 (for NetBEUI)

Note: All machines that will be exchanging data must be running the same transport protocol index.

CA_REPLY (Reply Mechanism)

Applies to Windows only

Specifies whether messages will be sent to the operator (OPR--the Event Management console) or to the machine on which the process is running (BOX).

When specified as OPR, messages are automatically routed to the console. BOX causes Event Management to pop up a message box on the particular machine when a message occurs. This may be useful if manual intervention is necessary when a program runs.

Valid Values: OPR and BOX

Default: OPR

CA_REPORT (Report Files directory)

Applies to UNIX/Linux and Windows

Indicates the working directory for the CAUTIL list command reports; these files have the extension .RET because they are produced by CA-RET. If you want to change this value, you must create the new directory and then change the value.

Default: $CAIGLBL0000/report (UNIX)

%CAIGLBL0000%\REPORT

CA_UNI_DB_INTERVAL (# of Seconds Between DB Load Retries)

Applies to Windows only

When the CA_UNI_USEDB variable is set to NO, this value is used to specify the number of seconds to wait between database operation (dbload/select) retries.

Range: 0 to 1000

Default: 60

CA_UNI_DB_RETRIES (Number of Retries Allowed for DBLOAD/SELECT)

Applies to Windows only

When the CA_UNI_USEDB variable is set to NO, this value is used to specify the number of times the database operation (dbload/select) is retried.

Range: 0 to 1000

Default: 60

CA_UNI_USEDB (Use database)

Applies to Windows only

When a successful load from a database has occurred, the database information is saved in a file. If subsequent attempts to load from the database fail, use this option to allow the database information to be loaded from the file. This gives you a relatively current set of data to work with until the database load can be completed on a retry. If you want to take advantage of this option, set the value to NO.

Default: YES

CA_UNICENTER_DB_ROOT (caiunidb Root Directory)

Applies to UNIX/Linux only

Contains the name of the UNIX file system directory where the caiunidb root file resides.

Default: none

CAI_DATEFMT (Date Format)

Applies to UNIX/Linux, Windows and z/OS USS

Defines the preferred date format to be used by Enterprise Management. The date format is comprised of three date elements, a day (dd or ddd), a month (mm or mmm) and a year (yy or yyyy). You can arrange the date elements in any order, but you must use each element once and only once.

Note: It is recommended that you use the same date format for both UNIX and Windows systems, so that they will be consistent. The date will always default to the format used on your Windows system.

A description of each date element follows.

dd Specifies a day of the month, from 01 to 31.

Note: The range of days in January is 01 to 31. The range of days in February is 01 to 28, except during leap year when the range is 01 to 29.

ddd Specifies a Julian day, from 001 to 366.

mm Specifies a numeric month, from 01 to 12.

mmm Specifies a character month. Valid specifications are: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, and DEC.

yy Specifies a valid year, from 00 to 99.

Note: Any year in the range 00 to 59 is translated internally as a twenty-first century date (2000-2059). Any year in the range 60 to 99 is translated internally as a twentieth century date (1960-1999).

yyyy Specifies a valid year, from 1960 to 2099.

Note: The valid date range is January 1, 1960 (01-JAN-1960) through December 31, 2099 (31-DEC-2099).

The date elements must be separated by a separator character. Valid separator characters are: - (hyphen), . (period), / (slash), , (comma), or space.

Size/Type: 1-11/alphanumeric bytes

Default: mm/dd/yyyy

CAI_DBSERVER (Database Server Name)

Applies to Windows only

The name of the machine running the SQL server database.

Size/Type: 1-15/alphanumeric bytes

Default: Assigned during installation

CAI_DBPWD (Database Password)

Applies to Windows only

The password assigned to the user ID used to connect to the SQL Server database. This value should not be changed.

Default: cadb

CAI_DBUID (Database User ID)

Applies to Windows only

The user ID used to connect to the SQL Server database. This value should not be changed.

Size/Type: 1-20/alphanumeric bytes

Default: cadb

CAI_MSG_EXIT

Applies to Windows only

This environment variable can be used to facilitate the execution of user-defined programs in response to a console message. Set the variable to any .exe, .bat, or other executable-like file either in the Windows Control Panel in Settings, System, User Environment or at the DOS prompt before launching CAUGUI CONLOG.

When that program or script is invoked, the name of a temporary file with the message in it and the name of the log file are appended as the last parameter. The user program should delete this temp file after completion.

Notes:

- If the program/script is not in the normal user PATH environment, a full path should be provided.

  The records in the temporary file are formatted according to the SDK documentation. See cauevt.h struct: EMEVT_LOG. The emlog1 program included in the SDK will convert the temporary file containing the log records (parm 2) to a text comma-separated file (parm 1).
- More than one record can be selected at a time. The temp file will contain all of them in selection order.

CAI_PRINT (Print Settings for Access or CA-RET)

Applies to UNIX/Linux only

Sets the print settings to either Access or Caret.

CAI_TNGREPOSITORY (CA NSM Repository Name)

Applies to Windows only

Identifies the server on which the Common Object Repository of the WorldView component of CA NSM resides. When a server is specified, Enterprise Management reports its status to the WorldView Maps.

Default: %CAI_DBSERVER%

CAIACTCOMSV (CA NSM Server Common Component Active?)

Applies to UNIX/Linux and Windows

Indicates whether the Enterprise Management common server components will be activated when Enterprise Management is started.

Default: No

CAIGLBL0000 (Enterprise Management Install Directory)

Applies to UNIX/Linux and Windows

Identifies the path where Enterprise Management is installed.

Default: None

CAIGLBL0002 (CAICCI Application ID for Criteria Profile Manager)

Applies to UNIX/Linux only

Indicates the CAICCI APPLID (application ID) of the Criteria Profile Manager.

Default: CACRITMON

CAIGLBL0003 (Server Node of Central Enterprise Management Server)

Applies to UNIX/Linux only

Identifies the host running the main Enterprise Management. Applicable only to remote hosts which are administered from a centralized copy of Enterprise Management.

Default: none

CAIGLBL0004 (Server Node)

Applies to UNIX/Linux

Identifies the name of the TCP/IP node (host name) where the tape server is installed. It must be set to the same value on all of the clients in the network.

Default: none

CAIGLBL0005 (CAICCI Application ID for SNMP Trap Manager)

Applies to UNIX/Linux only

Identifies the CAICCI application ID to be assigned to the Enterprise Management SNMP Trap Manager.

Default: none

CAIGLBL0006 (CAICCI Application ID for CATRAPD)

Applies to UNIX/Linux only

Identifies the CATRAPD CA-Trap Manager CAICCI ID.

Default: CASNMPMGR

CAIGUI0002 (Control GUI In Progress Windows)

Applies to UNIX/Linux only

Controls whether the GUI access in progress windows are displayed when you open a new folder window or access the Enterprise Management database from your GUI session. If this variable is set, the access in progress windows are displayed.

CAILOCALAPPMAP (Use Local UNIAPP.MAP)

Applies to Windows only

Specifies whether the UNIAPP.MAP file for the client will be merged with the global UNIAPP.MAP file on the server. By default, if a client UNIAPP.MAP file exists, Enterprise Management merges it with the global UNIAPP.MAP file to provide the client with a combined view of Enterprise Management. Enterprise Management looks for the client UNIAPP.MAP in the root directory of %CAILOCL0000%.

Valid Values: YES or NO

Default: YES

CAIMLICSRV (License Server Status)

Applies to UNIX/Linux only

Reserved for internal use. Identifies the status of the license server.

Default: none

CAIUNIDB (Logical Database Name of Enterprise Management)

Applies to UNIX/Linux

Specifies the logical database name of the Enterprise Management database.

Default: caiunidb

CAIUNIXDEBUG (Enterprise Management UNIX Debug)

Applies to UNIX/Linux only

If set in environment, this program will provide diagnostic information.

CAIUSER (User Working Directory)

Applies to Windows only

Specifies the path of the internal working directory used by Enterprise Management. If you want to change this value, you must create the new directory and then change the value.

Default: %CAIGLBL0000%\CAIUSER

CAUWVINI (Populate Repository from Local Node)

Applies to Windows

Lets you populate the repository with information from the local node only. Set this to LOCALONLY before running setup or CAUWVINI.

Default: None

DPATH (Data Path for GUI Data Files)

Applies to Windows only

Defines the full path name that the Enterprise Management GUI uses to search for data files such as .CDR and .FLD.

Default: %CAIGLBL0000%\DATA

Agent Technology

AGENTWORKS_DIR

Applies to Windows, UNIX/Linux, z/OS USS

Points to the root directory where Agent Technology is installed. This environment variable must be set for the Agent Technology common services to run.

Windows

```
%AGENTWORKS_DIR%
```

UNIX/Linux, z/OS USS

```
$AGENTWORKS_DIR
```

Chapter 13: Address Space

This chapter covers address space management. This includes management of permanent address space for specified purposes.

This section contains the following topics:

CAHCHECK Address Space

This section contains information about the CA Health Checker address space.

Purpose

CAHCHECK is a CA common service system address space that hosts health checks on behalf of CA products and components.

Many CA product health checks are hosted directly by the owning CA product address space. Some CA products do not have a permanent address space in which to host the health checks. Also, some CA products that have a permanent address space want some or all of the health checks to be hosted in a more permanent address space. The CAHCHECK address space was created to host CA product health checks that have these requirements.

CA product health checks provide the following value for your site:

- Improve product availability by eliminating outages due to configuration option errors
- Point out product features that should be activated to get maximum benefit from a product
- Point out settings that can optimize a product's performance. This is similar to having a product expert constantly reviewing the product settings in your environment.

The CAHCHECK address space is optional, but we strongly recommend that you use it to receive the full benefit of CA product health checks. It is important to note that the CAHCHECK address space requires the CAMASTER common service system address space.

Operational Overview

The CAHCHECK address space is started automatically by the CAMASTER address space at IPL time and is intended to remain active for the duration of IPL. However, it can be shut down or recycled using operator commands.

You do not need to shut down the CAHCHECK address space before a scheduled IPL. The CAHCHECK address space will not prevent any other system component address space from shutting down.

No action is required to activate the CAHCHECK address space. That is, no JCL PROC, no security access rules, and no WLM policy requirements are required to run the CAHCHECK address space. The CAHCHECK address space does not require a security system user ID, unless your installation's default started task user ID is defined to be suspended. If this is the case, then a started task user ID of CAHCHECK minimally must be defined to permit the CAHCHECK address space to start. The CAHCHECK address space executes like any other system component as a "Trusted User" authority. This means that CAHCHECK is permitted to access the libraries that contain the CA product health checks. It also means that any access to those libraries is logged. The CAHCHECK address space executes in the SYSSTC service class.

After the CAHCHECK address space is active, its services are available to CA products. When CA products register their health check routines in the CAHCHECK address space, those checks remain active until they are terminated either by the owning product or by an operator command. If the CAHCHECK address space is recycled, any CA Health Checks that were hosted before shutdown are re-registered automatically when the CAHCHECK address space is restarted.

CA Health Check Owners

Any CA product that registers CA product health checks in the CAHCHECK address space is referred to as a "check owner", or simply "owner". Various commands use the OWNER parameter to refer to the CA product health checks hosted by a specific health check owner.

Communicate with the CAHCHECK Address Space

Use the MVS MODIFY (F) operator command to communicate with the CAHCHECK address space. For your convenience, the "CAHC" is dynamically set up as the short form, so communication with CAHCHECK is accomplished with MVS operator MODIFY commands in the following format:

F CAHC,*command*

Several operator commands are available for controlling the CAHCHECK address space, controlling the hosted CA Health Checks, and displaying information related to both.

Recycle the CAHCHECK Address Space

To recycle the CAHCHECK address space, issue the SHUTDOWN command with the RESTART option:

```
F CAHC,SHUTDOWN,RESTART
```

```
F CAHC,SHUT,R
```

Start the CAHCHECK Address Space

If you have shut down the CAHCHECK address space without the restart option, then you must manually restart it by issuing the following MVS console command:

```
S IEESYSAS,PROG=CAHCHECK,JOBNAME=CAHCHECK,TIME=1440,SUB=MSTR,REUSASID=YES
```

Handling Data Set Contention

Hosted CA Health Checks may have data sets allocated to them. If other processes require exclusive control of those data sets, they will be in contention with the CAHCHECK address space. You do not need to shut down the CAHCHECK address space to resolve the contention. Use the DISPLAY ALLOC command to determine which check owner has the data set:

`F CAHC,DISPLAY ALLOC`

Displays all data sets allocated to all checks

`F CAHC,D A,DSN=`*`data.set.name`*

Displays a specific data set

`F CAHC,D A DSN=`*`generic.data.set`*`*`

Displays all data sets matching partial name up to the asterisk

The check owner task name is displayed in the response to this command. You can then terminate the checks belonging to the check owner with the STOP command:

`F CAHC,STOP `*`ownertask`*

Stop the checks and cause deallocation of data sets

CAHCHECK Command Reference

The following table contains an alphabetical list of CAHCHECK commands that you can pass to the CAHCHECK address space using the MVS MODIFY (F) operator command.

Command	Description
CANCEL *taskname*	Terminates a task that is not responding to a STOP or DELETE command. Use the CANCEL command only as a last resort to end a health check owner task. Use the STOP command to stop the check or use the DELETE command to totally end the check owner's task and all of its health checks. While most CA Health Checks complete their function very quickly, some CA Health Check routines run longer. The health check task is normally allowed to complete, so it may take longer to respond to the STOP or DELETE than you might expect. If the health check task does not respond, you can issue the CANCEL command to completely end the task and free up any associated storage. This may leave data sets allocated which you need to free manually using the DEALLOC command.
DEALLOC DSN=*data.set* DEALLOC DDN=*ddname*	Deallocates data sets that are currently allocated to the CAHCHECK address space. If DDN= is specified, only the one data set identified by the DD name is freed. If DSN= is specified, every occurrence of an allocation for that data set name is freed.
DELETE *ownertask*	Use the DELETE command to terminate a hosted health check owner. Unlike the STOP command, which suspends the task, the DELETE command causes the hosted health check owner to go through final termination and it cannot be restarted. Consider using the STOP command to allow the health check owner to be started again later.
DISPLAY ALLOC D A,DSN=*data.set* D A,DSN=*data.set.qual**	Displays the data sets allocated to the CAHCHECK address space. If DSN= is specified, all allocations that match that fully qualified data set name are displayed. If the trailing asterisk (*) is specified, then all data sets that match the name up to the asterisk are displayed.
DISPLAY EVENTS	Displays any events that the CAHCHECK address space is monitoring.
DISPLAY INIT	Displays the initialization settings that are in effect.

Command	Description
DISPLAY MODULES	Displays the list of modules that comprise the CA Health Checker infrastructure (CASHCCMN and CASHCLPA) followed by the modules that comprise the CAHCHECK address space (CASHCPVT). The CSECT components are displayed with their assembly date and time and maintenance level.
DISPLAY OPTIONS	Displays the current options in effect.
DISPLAY OWNERS D OWNER,NAME=*owner* D OWNER DETAIL	Displays the health check owners being hosted in the CAHCHECK address space. You can specify the DETAIL parameter to produce much more detailed display of health check owners. You can also display the information on a single check owner using the NAME= operand.
DISPLAY TASKS	Displays the various tasks being managed within the CAHCHECK address space. Some of these tasks are required by the CA Health Checker infrastructure to manage the CAHCHECK address space itself, while others are used to manage the hosted health check owners.
HELP HELP *command*	Displays help about the commands that CAHCHECK supports. If issued with no operands, the help summary is displayed. You can get more detailed help on a given command using the command name as the HELP operand.
SET OPTIONS DEBUGMSG=YES|NO	Sets the control settings for the CAHCHECK address space. DEBUGMSG= can be set to YES activate debug messages or NO to suppress them.
SET OWNER *owner* DEBUG=YES|NO	Alters control settings for a given health check owner. You can turn debugging and tracing on and off to cause the health checks to produce additional diagnostic information when they run.
SHUTDOWN, RESTART SHUTDOWN	Shuts down and automatically restart the CAHCHECK address space when the RESTART parameter is specified. This recycles the CAHCHECK address space in one command. When the RESTART parameter is omitted, this command shuts down the CAHCHECK address space without restarting. You can also use the MVS STOP (P) command to shut down the CAHCHECK address space without restarting.

Command	Description
START *taskname*	Use the START command to restart a task that was stopped with the STOP command. Use the DISPLAY TASKS command to display the tasks being managed in the CAHCHECK address space. Any task whose status is "Stopped" may be started with the START command.
STOP *taskname*	Use the STOP command to suspend a task. Any task that is considered crucial to the CA Health Checker infrastructure is not eligible to be stopped. Any hosted health check task can be stopped. Stopping a hosted health check frees any data set that was allocated by the health check owner and any of its health check routines. This is useful when there is contention with a data set allocated to a given health check owner task.

Maintenance Considerations

Both CA product health checks and CA Health Check services modules can be refreshed without an IPL

Maintenance Applied to a Hosted Check

If you apply maintenance to a CA Health Check module that is currently being hosted by the CAHCHECK address space, you can refresh the module by issuing the IBM Health Checker REFRESH command:

F *hzsproc*,REFRESH,CHECK(*CA_owner*,*chkname*)

hzsproc

Specifies the IBM Health Checker started task address space name

CA_owner

Specifies the CA product owner name

chkname

Specifies the specific CA Health Check name

The IBM Health Checker sends a refresh directive to the CA Health Checker infrastructure, which responds by shutting down the hosted health check, deleting the modules used by that check from memory, and then re-registering the CA Health Check using the newly refreshed CA Health Check module.

Maintenance Applied to CA Health Checker Dynamic LPA-Resident Module

The CA Health Checker infrastructure modules are loaded into LPA during initialization. The addresses of the various service routines are anchored into a control structure that does not move once it has been created. If you apply maintenance to any of the modules that are LPA resident, issue the following MVS console command after LLA has been refreshed:

```
SETPROG LPA,ADD,MODNAME=CASHCLPA,DSNAME=LNKLST,FIXED,PAGEPROTALL
```

The CA health Checker infrastructure automatically detects the new copy and updates the addresses that are anchored in the control structure. Subsequently, it initiates an automatic refresh of all hosted health checks in the CAHCHECK address space to ensure that only the freshest copy of the infrastructure is being used.

Normally, if any maintenance requires this action, the instructions accompanying the PTF will advise you to take this action.

Maintenance Applied to CAHCHECK Address Space-Resident Modules

If you apply maintenance to the CAHCHECK address space service and control modules, you must recycle CAHCHECK address space to obtain the most current copy. After applying the modules to the LINKLIST and refreshing LLA, you can use the following command to recycle the CAHCHECK address space:

```
F CAHC,SHUT,RESTART
```

This command terminates all the hosted checks, shuts down the address space, and starts up a new CAHCHECK address space with the refreshed modules. Any hosted CA health checks that were active before the shutdown are re-registered with the IBM Health Checker and operations resume as they were, except the newly refreshed CAHCHECK address space-resident modules are now in control.

Again, if any maintenance requires this action, the instructions accompanying the PTF will advise you to take this action.

CAMASTER Address Space

This section contains information about the CAMASTER address space.

Purpose

CAMASTER is a non-cancelable started task that provides system services used by various CA products and CA Common Services such as the CA Health Checker address space. It also provides storage resources to CA components that are used instead of CSA and ECSA, thus reducing the demand on your system's common storage. After initialization, CAMASTER uses minimal CPU, and cannot be stopped or restarted.

Operation

The CAMASTER address space starts up automatically at IPL time and remains active for the duration of IPL in a way consistent with the IBM Master Scheduler address space. Likewise, there is no need to shut down the CAMASTER address space before a scheduled IPL. The CAMASTER address space will not prevent any other system component address space from shutting down.

No action is required to activate the CAMASTER address space. That is, no JCL PROC, no security access rules, no security user ID, and no WLM policy requirements are required to run the CAMASTER address space. The CAMASTER address space executes like any other system component as a "Trusted User" authority. This means that any CAMASTER resource access is logged. The CAMASTER address space executes in the SYSTEM service class. After CAMASTER completes its initialization processing, no further processing takes place.

Optional CAIMST00 Logical Parmlib Member

You can optionally add a CAIMST00 member to logical PARMLIB to override the default values and behavior of the CAMASTER address space.

The CAIMST00 member can contain a CAMSINIT control statement and comments. Place CAIMST00 in your installation logical PARMLIB concatenation.

The following example control statement illustrates all the defaults that would be in effect if the CAIMST00 member is omitted:

```
CAMSINIT  START,             /* Start up at IPL time       */
          JOBNAME(CAMASTER)  /* Job name to use            */
```

To maximize the value of your investment in CA technology, we strongly recommend that you use the CAMASTER address space. However, if you do not want to have the CAMASTER address space running in your system, use the "NOSTART" parameter instead of "START". By not starting the CAMASTER address space, other CA Common Services may not be fully operational. Most notably, the CA Health Checker system address space, CAHCHECK, cannot be started without CAMASTER.

The CAMASTER system address space is currently optional. However, at some future date, it may be required by specific CA products.

Maintenance Applied to a CAMASTER Dynamic LPA-Resident Module

CAMASTER modules are loaded into dynamic LPA during the CAMASTER system address space initialization processing. If you apply maintenance to any of the modules that are dynamic LPA resident, issue the following MVS console command after LLA has been refreshed:

```
SETPROG LPA,ADD,MODNAME=module,DSNAME=LNKLST,FIXED,PAGEPROTALL
```

Normally, if any maintenance requires this action, the instructions accompanying the PTF will advise you to take this action.

Index

A

C

D

E

G

I

L

M

R

S

T