CA SYSVIEW® Performance Management

Administration Guide

Version 14.0

CA Technologies Product References

This document references the following CA Technologies products:

- CA Application Performance Management (CA APM)
- CA Common Services for z/OS (CCS for z/OS)
- CA Datacom®/DB (CA Datacom/DB)
- CA Easytrieve® Report Generator (CA Easytrieve)
- CA Insight™ Database Performance Monitor for DB2 for z/OS (CA Insight DPM for DB2)
- CA MIA Tape Sharing (CA MIA)
- CA OPS/MVS® Event Management and Automation (CA OPS/MVS)
- CA Roscoe® Interactive Environment (CA Roscoe)
- CA SERVICE DESK
- CA SYSVIEW® Performance Management (CA SYSVIEW)
- CA SYSVIEW® Performance Management CA Datacom® Option (CA SYSVIEW CA Datacom Option)
- CA SYSVIEW® Performance Management Option for CICS (CA SYSVIEW Option for CICS)
- CA SYSVIEW® Performance Management Option for IMS (CA SYSVIEW Option for IMS)
- CA SYSVIEW® Performance Management Option for TCP/IP (CA SYSVIEW Option for TCP/IP)
- CA SYSVIEW® Performance Management Option for WebSphere MQ (CA SYSVIEW Option for WebSphere MQ)

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.

Documentation Changes

The following documentation updates have been made after the last release of this documentation:

- Removed the CICSGRPS parmlib member.
- Updated the Main Services Address Space Initialization (see page 32) topic: Removed the SVC information.
- Updated the Start the CICSLOGR Subtask (see page 52) topic: Added the Auxiliary Address Spaces information.
- Updated the How the Main Services Address Space Is Initialized (see page 72) topic: Removed the SVC information.
- Updated the Define CA Insight DPM for DB2 Connections (see page 92): Added a note.
- Updated the Run the System Information Utility (see page 107) topic: Removed the SVC information.
- Updated the CICSOPTS (see page 308) parmlib member: Removed reference to CICSGRPS.

Contents

Chapter 3: Starting the Product Server and Its Subtasks 71

Chapter 4: Customization Information 81

Chapter 19: Parmlib Members 225

Chapter 1: Before Starting the Product

This chapter provides administrative information that you should be aware of before you start CA SYSVIEW.

This section contains the following topics:

CA GSS of CCS

CA GSS is a service of the CA Common Services for z/OS product that contains modules shared by several other CA products. Thus, enhancing their features and letting them communicate easily, seamlessly, and reliably with one another. CA GSS provides quick access to information from various sources.

CA GSS is required to use these CA SYSVIEW capabilities:

- The System Condition Monitor (SCM)
- The CA SYSVIEW Console Interface
- The ability to run IMODs (REXX execs) when thresholds are exceeded

Install and Customize CA GSS

The GSS samplib member contains statements specific to CA SYSVIEW that are required in the GSS.PARMLIB(RUNPARM) member.

- For CA GSS installation instructions, see the CA Common Services for z/OS documentation. For an example of how CA GSS is defined to CA SYSVIEW, see the GSS member of *sysview*.CNM4BSAM.

  Note: The parameters *must* be from the latest release of this member.
- To customize CA GSS to interface with CA SYSVIEW, see the *sysview*.CNM4BSAM member GSS for all required parameter statements.

Startup Considerations

Ensure that the SYSVIEW main address started task (SYSVIEW) procedure is started before CA GSS.

What Are IMODs

CA GSS lets you execute IMODs or *intelligent modules*, which are written in the REXX language. Execute IMODS together with the additional functions and instructions supplied by CA.

Use IMODs to automate system monitoring and regulate resources, or to create your own online reports on system activities. CA products use IMODs to communicate among themselves and to provide additional product functionality. IMODs can be used for, but are not limited to, the following types of tasks:

- Automating system monitoring
- Regulating resources
- Reporting on system activities
- Developing batch reports

Types of IMODs Provided by CA GSS

CA GSS provides the following types of IMODs for your convenience:

- Service Routine

  IMODs that can be called to provide both services and information to client IMODs.
- Server

  A commonly used service IMOD provided as a subroutine.
- Special Purpose

  IMODs that are executed under certain circumstances.

SCM and IMODs

The SCM uses IMODs to monitor multiple subsystems and collect information from a single screen. If you are upgrading to a new release of CA SYSVIEW, be sure to update the CA GSS ISET statement in the GSS RUNPARMS to point to the latest *sysview*.CNM4BISR data set.

Note: For more information, see the CA Common Services for z/OS documentation.

More information:

Performance IMODs (see page 175)

Customize CAICCI

Enable the Cross-system Resource Monitoring feature by defining CAICCI cross-system connections. In the CCIPARM member of the CAIENF component, include the appropriate SYSID, PROTOCOL, NODE, and CONNECT statements.

Note: For more information, see the CA Common Services for z/OS documentation and the CA SYSVIEW online help topics for CAICCI setup suggestions.

Chapter 2: Starting the Product Subtasks

This chapter explains how to start CA SYSVIEW and describes the subtasks. The Main Services Address Space and the User Interface Address Space of CA SYSVIEW control the subtasks.

You can further customize these subtasks using the appropriate PARMLIB members.

Once the address spaces have been started, we recommend using the online address space Administration command, ASADMIN, to perform additional functions.

This section contains the following topics:

Address Space Organization

The CA SYSVIEW product is comprised of two integrated address spaces.

- Main services address space
- User interface address space

Each address space has a defined purpose.

Main Services Address Space

The Main Services Address Space is the primary address space for CA SYSVIEW. Its primary role is to provide services and data collection functions. This address space is separate from the user interface address space so it can perform authorized functions.

More information:

Address Space Organization Diagram (see page 31)

User Interface Address Space

The User Interface Address Space is the display facility interface. Multiple user interface address spaces can be used.

The following CA SYSVIEW user interface types connect to this address space:

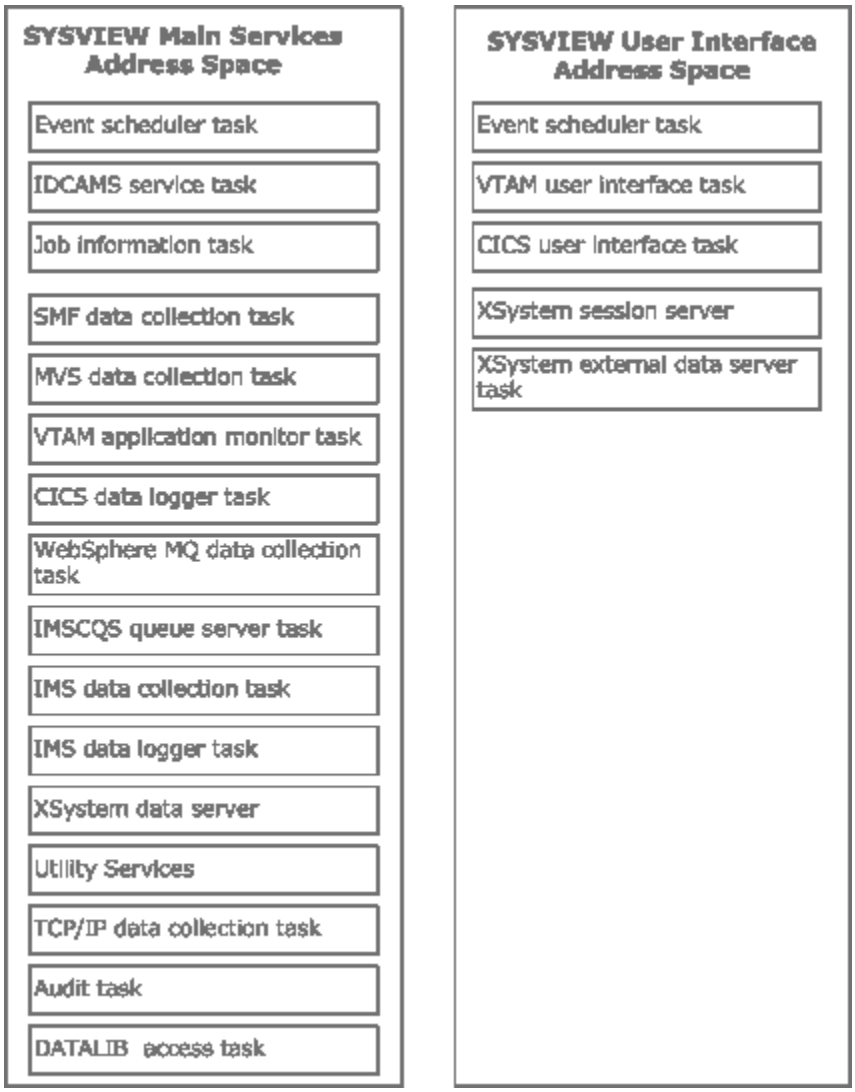
- VTAM
- CICS
- Cross system connections
- CA SYSVIEW CA Vantage GMI (CA Vantage GMI)

More information:

User Interface Address Space Initialization (see page 62)
Address Space Organization Diagram (see page 31)

Address Space Organization Diagram

The following diagram shows the organization of the main services address space and the user interface address space:



Main Services Address Space Initialization

The Main Services Address Space provides common services that other CA SYSVIEW tasks and users use. Optionally, data collection components can also be started in this task. The main task controls the starting and stopping of other subtasks. Start the Main Services Address Space as early as possible during an IPL.

The primary initialization information is obtained from the System Configuration Options member in the system parameter library. The System Configuration Options member contains information such as run-time data set names and features that are enabled. A System Configuration Options member is defined for each z/OS system.

You can run multiple copies of CA SYSVIEW on a single system.

If multiple copies are desired, maintain the following rules:

- Multiple copies of the same release must specify different subsystem IDs.
- Multiple copies of different releases can use the same subsystem ID.

PARMLIB Member SYSVIEW

The SYSVIEW parmlib member provides the list of functions and tasks that are started during initialization.

The MEM= parameter in the started task SYSVIEW specifies the parmlib member that is used. Detailed information regarding the options can be found in the parmlib member.

Default setting: MEM=SYSVIEW

The special keyword PRIMARY must be specified in this member. This keyword indicates that the current address space is to be the Main Services Address Space.

Main Services Address Space Subtasks

You can execute or start the following subtasks in the Main Services Address Space:

AMS

IDCAMS service subtask

APPLMON

VTAM application monitor subtask

AUDIT

Audit subtask

CICSLOGR

CICS data logger subtask

DATALIB

DATALIB access subtask

IMSDATA

IMS data collector subtask

IMSCQS

IMS common queue server subtask

IMSLOGR

IMS data logger subtask

JOBS

Job information collection

MQSDATA

WebSphere MQ data collector subtask

MVSDATA

MVS data collector subtask

SCHEDULR

Event scheduler subtask

SMFDATA

SMF data collector subtask

TCPDATA

TCP/IP data collector subtask

UTIL

Utility services subtask

XSDS

XSystem data services

PARMLIB Member OPTIONS Sets Subtask Configuration Defaults

The OPTIONS parmlib member allows default configuration to be set for all subtasks. Detailed information regarding the options can be found in the parmlib member.

PARMLIB Member STARTCMD

The STARTCMD parmlib member is processed immediately after the initialization has been completed. The member can contain a list of any valid z/OS operator commands. These commands are issued following initialization. Detailed information regarding the options can be found in the parmlib member.

Main Services Address Space Start Modes

You can specify the start mode for the CA SYSVIEW Main Services address space and associated tasks. The start mode for the Main Services address space and each associated task can be different. You can specify the following two start modes for the Main Services address space:

WARM

Retrieves configuration information from the persistent data store. Configuration information from the previous session is restored.

COLD

Retrieves configuration information from the PARMLIB member. Configuration information from the previous session is not restored.

Default: WARM start

Note: The User Interface address space does not support a start mode.

Main Services Tasks

Explicitly specifying a COLD start for the Main Services address space performs a COLD start for all associated tasks. A COLD start is performed even if the task has requested a WARM start.

The following main services tasks support WARM and COLD start modes:

APPLMON

Starts the VTAM application availability monitor task.

AUDIT

Starts the audit task.

IMSDATA

Starts the IMS performance data collection task.

MQSDATA

Starts the WebSphere MQ performance data collection task.

MVSDATA

Starts the z/OS performance data collection task.

SCHEDULR

Starts the event scheduler task.

SMFDATA

Starts the SMF data collection task.

TCPDATA

Starts the TCP/IP data collection task.

How to Specify the Start Mode on the Started Task

You can specify the start mode when you start the Main Services address space started task.

Specify the start mode using one of the following three methods:

- Method 1: Issue the following START command for the started task:

  ```
  S SYSVIEW,START=COLD
  S SYSVIEW,START=WARM
  ```

- Method 2: Use the following command START= parameter within the started task PROCLIB member:

  ```
  //SYSVIEW PROC MEM=SYSVIEW,START=WARM
  .
  .
  //SYSVIEW EXEC PGM=GSVXMAIN,REGION=0M,TIME=1440,
  //         PARM='&MEM,&START'
  .
  ```

- Method 3: Specify as an initialization option in the parmlib member specified by the MEM= parameter. The default member name is SYSVIEW:

  ```
  WARM
  or
  COLD
  ```

Start Mode Matrix Table

The following table shows how the start mode specified using the STC PARM= and PARMLIB MEM= affects the resulting start mode.

STC PARM=	PARMLIB MEM=	Resulting Start Mode
None	None	WARM
None	COLD	COLD
None	WARM	WARM
COLD	None	COLD
COLD	COLD	COLD
COLD	WARM	COLD
WARM	None	WARM
WARM	COLD	WARM
WARM	WARM	WARM

Note: When you explicitly specify the start mode on the START command or within the PROC, causes the parmlib setting to be ignored.

MODIFY Command—Communicate with the Started Task

Once the main services task has started, use the z/OS MODIFY (F) command to communicate additional requests to the task. You can also use the online Address Space Administration command, ASADMIN, to perform additional functions.

The available commands and functions are:

STATUS

Displays the status of all subtasks or a particular subtask in the Main Services Address Space. The STATUS command has the following syntax:

`MODIFY sysview,STATUS [subtask]`

subtask

Specifies the subtask name for the requested status display. The valid values for *subtask* are: VTAM and CICS.

START

Starts a subtask in the Main Services Address Space. The START command has the following syntax:

`MODIFY sysview,START subtask[.taskident,parameters]`

subtask

Specifies the name of the subtask to start.

taskident

Identifies a task, if you are starting multiple tasks in the same subtask.

parameters

Represents the parameters to pass to the subtask.

STOP

Stops a subtask in the Main Services Address Space. The STOP command has the following syntax:

`MODIFY sysview,STOP subtask|taskid`

subtask

Specifies the name of the subtask to terminate.

MODIFY

Modifies current parameter settings for a subtask in the Main Services Address Space. The MODIFY command has the following syntax:

`MODIFY keyword value <<COMMAND> cmdname<.scrname><.fmtname>>`

cmdname

Displays the option value for the specified command rather than the active command.

scrname

Displays the option value for the specified command and screen name rather than the active command and screen.

fmtname

Displays the option value for the specified command, screen name, and format name rather than the active command screen, and format.

FORCE

Forces the subtask in the Main Services Address Space to terminate abnormally. The FORCE command has the following syntax:

`MODIFY sysview,FORCE subtask|taskid`

subtask

Specifies the name of the subtask you want to terminate abnormally. The task is canceled with a dump.

Main Services Address Space Termination

The Main Services Address Space can run continuously for the duration of an IPL. The address space can be terminated and restarted at any time. All subtasks executing in the Main Address are stopped automatically.

PARMLIB Member STOPCMD

The STOPCMD parmlib member is processed immediately at the start of the termination process. The member can contain a list of any valid z/OS operator commands. These commands are issued before termination. Detailed information regarding the option can be found in the parmlib member.

Stop the SYSVIEW Started Task

The Main Services Address Space can be terminated using the z/OS STOP (P) command. The task can be stopped from a system console by entering the command P SYSVIEW.

Event Scheduler Subtask

Initiating processes and functions based on a predefined time schedule is an important part of operating a computer system. The CA SYSVIEW Event Scheduler can initiate a variety of functions at a specified time or recurring interval. You can initiate events in CA SYSVIEW as well as outside of its boundaries. The Event Scheduler subtask executes in the Main Services Address Space and starts automatically during initialization.

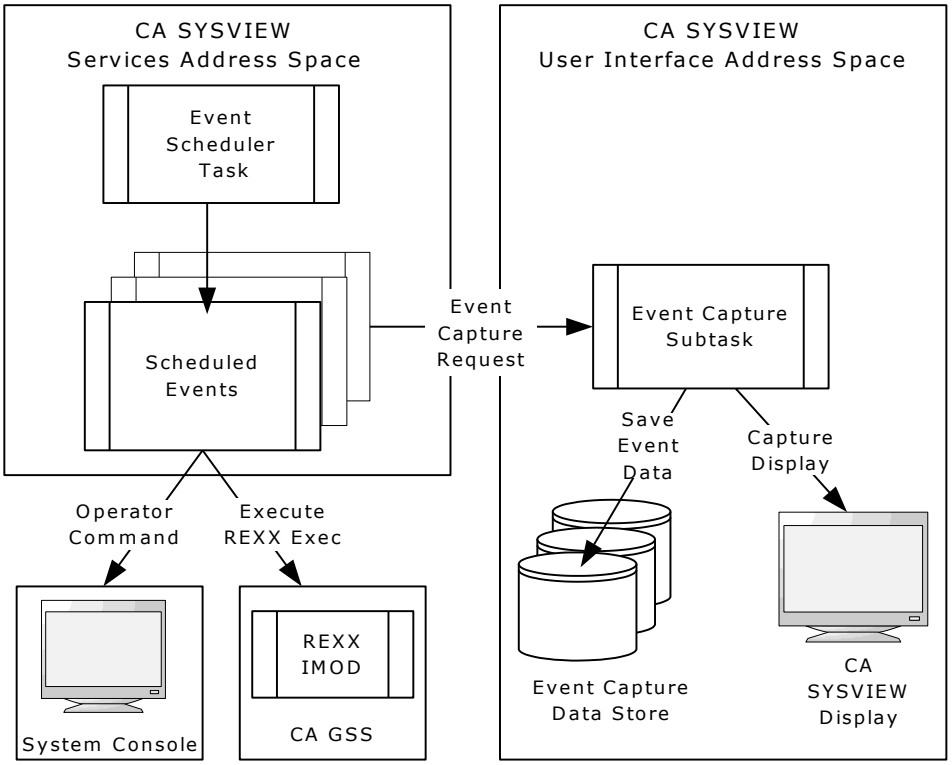
How Event Scheduler Subtask Works

When a scheduled event is due, the Event Scheduler Subtask performs the following:

- Issues a z/OS operator command.
- Initiates asynchronously an Event Capture request.
- Executes asynchronously a REXX exec.
- Displays a list containing the previously specified functions.

Note: The Event Scheduler lets you control or limit the data collection metrics.

The following diagram shows the Event Scheduler process:



More information:

Data Collection (see page 97)

PARMLIB Member SCHEDULE

The SCHEDULE parmlib member provides SCHEDULR configuration options and event definitions.

For detailed information regarding these options and sample event definitions, see the SCHEDULE parmlib member.

Start the SCHEDULR Subtask

This statement is only valid in the Main Services Address Space.

The Scheduler subtask starts automatically when the Main Services Address Space initializes. Configuration options and event definitions are defined in the sysview.CNM4BPRM member SCHEDULE. There is no need to include a START statement in the sysview.CNM4BPRM member SYSVIEW. If an entry is included, an error message is generated indicating that the task is already started.

To start a stopped subtask, issue the following z/OS MODIFY command:

```
MODIFY sysview,START SCHEDULR
```

AUDIT Subtask Setup

The AUDIT subtask records system resource events or changes occurring within CA SYSVIEW. These events can be initiated by a user or internal task.

The AUDIT subtask starts automatically when the Main Services Address Space initializes. Configuration options for the Audit Event component are defined in the AUDIT parmlib member. The AUDITDEF parmlib member contains a list of events that are to be audited.

Start the AUDIT Subtask

The AUDIT Event displays let you view and control the audit activities.

Follow these steps:

1. Enter the AUDITDEF command.

   The Audit Event Definitions menu displays showing the current settings for each defined audit event. With proper security authorization, you can alter the settings for this menu.

2. Enter the AUDITLOG command.

   The Audit Event Log menu displays the contents of the audit event log stream. This lets you view historical audit events.

3. Enter the AUDITMSG command.

   An audit event is logged for type USER_EVENT_MESSAGE. The content of the audit event is the text message entered with the command.

DATALIB Subtask Setup

The DATALIB subtask stores and maintains all information about the data objects.

Use the following guidelines to set up the DATALIB:

- You can have one or multiple DATALIB data sets.
- Use only one DATALIB for any one instance of CA SYSVIEW.
- The administrator may choose to have a unique DATALIB for each instance of SYSVIEW or share one of the DATALIBs.
- One CA SYSVIEW can use only one DATALIB.
- You can share a DATALIB within a sysplex but not across multiple sysplexes.
- Multiple releases of CA SYSVIEW can share the DATALIB, making migration to a new release easier.

Start the DATALIB Subtask

The persistent data store provides an environment called the DATALIB for maintaining data objects. You can save, retrieve, and copy data from the persistent data store. Some DATALIB monitoring may be required to watch for a full condition.

To start the DATALIB subtask, enter the DLLIST command. A menu displays showing the contents of the CA SYSVIEW DATALIB library.

IDCAMS Service Subtask Setup

The IDCAMS Service subtask performs Access Method Services (AMS) requests. These requests must be performed in the main services task due to authorization requirements. The IDCAMS Service subtask starts automatically when the Main Services Address Space initializes.

Start the AMS Subtask

This statement is only valid in the Main Services Address Space.

The AMS subtask starts automatically when the Main Services Address Space initializes. There is no need to include a START statement in the *sysview*.CNM4BPRM member SYSVIEW. If an entry is included, an error message is generated indicating that the subtask is already started. If, for some reason, the subtask has stopped and needs to be restarted, use the following z/OS MODIFY command to restart the subtask:

```
MODIFY sysview,START AMS
```

JOB Information Subtask Setup

The JOBS subtask collects job summary information used by the JOBSUM, LISTINP, JJOBQUE, JHELDQUE, and JOUTQUE commands. The subtask starts automatically when the Main Services Address Space initializes. This subtask is optional and does not have to be started; although, the information collected by the JOBS subtask improves the performance of the JOBSUM, LISTINP, JJOBQUE, JHELDQUE, and JOUTQUE display commands.

If the JOBS subtask is not started, CPU usage for a user's session will increase because the data normally kept by the JOBS subtask will have to be acquired by the user's subtask.

JES Job Queue Updates

Periodically, the JOBS subtask checks the status of jobs on the JES queues. When the information for the jobs on the JES queues changes, the JOBS subtask updates the information.

Set the Update Interval

The range for wait-time interval is 1 to 9999 seconds. The JOBS subtask waits the default interval of 30 seconds and then checks to see if anything has changed. Also, if someone is using the JOBSUM command, the information is updated every time they enter the REFRESH command on the Job Summary Display.

You can set the JOBS task interval higher during times when fewer users are using the JES job-related commands. The response time of JES job-related commands might increase but the information provided will be up-to-date.

To dynamically change the wait-time interval, issue the MODIFY command.

Example: Change the JOBS Interval

The following command changes the interval to 60 seconds:

```
F SYSVIEW,MODIFY JOBS,INTERVAL=60
```

Default: INTERVAL=30

Start the JOBS Subtask

This statement is only valid in the Main Services Address Space.

The JOBS subtask executes as part of Main Service Address Space. Address space initialization statements are found in the sysview.CNM4BPRM member SYSVIEW.

To start the JOBS subtask, enter the following z/OS command:

```
START JOBS,parameters
```

The *parameters* value is:

INTERVAL=*nnn*

Specifies the update interval as the number of seconds from 1 to 9999.

Default: 30 seconds

SMF Event Capture Subtask Setup

The SMF Event Capture facility:

- Is enabled through z/OS Dynamic Exits.

  Many third-party products, such as z/OS, produce SMF records. These records typically contain statistics or status information about resources in use. This facility provides a convenient way to log or suppress specific record types or subtypes. Captured records that have been written to the SYSVIEW SMF log can be displayed online. Several record format routines have been provided.

- Uses z/OS Dynamic Exits IEFU83 and IEFU84 to capture records as they are being written to SMF.

  You must enable these dynamic exits in the SMFPRM*xx* member of SYS1.PARMLIB by adding IEFU83 and IEFU84 to the EXITS() parameter on the SYS entry, and all SUBSYS entries, if they are not already specified.

Example: SYS and SUBSYS definitions

This shows IEFU83 and IEFU84 included in the SYS and SUBSYS definitions.

```
SYS(NOTYPE(4,5,19,20,34,35,40,99,169),
      EXITS(IEFU83,IEFU84,IEFACTRT,IEFUJV,IEFU85,
      IEFUSI,IEFUJI,IEFUTL,IEFU29),
      INTERVAL(SMF,SYNC),NODETAIL)

SUBSYS(STC,EXITS(IEFUJI,IEFU29,IEFU83,IEFU84,IEFU85,
      IEFUTL,IEFACTRT))
```

PARMLIB Member SMFDATA

The SMFDATA parmlib member provides default configuration options for the SMF Event Capture Function.

Note: For detailed information regarding the options, see the SMFDATA parmlib member.

PARMLIB Member SMFTYPE

The SMFTYPE parmlib member provides information about individual SMF record types and subtypes. This member provides a description for each record type and subtype. Optionally, a program module can be associated with each record type and subtype. The associated program is called to format the record display.

Note: For detailed information regarding the options, see the SMFTYPE parmlib member.

Start the SMFDATA Subtask

This statement is only valid in the Main Services Address Space.

The SMFDATA Event Capture subtask executes as part of Main Services Address Space. Address space initialization statements are found in the *sysview*.CNM4BPRM member SMFDATA.

To start the SMFDATA SMF Event Capture subtask, enter:

```
START SMFDATA
```

MVS Data Collection Subtask Setup

The MVS Data Collection subtask performs monitoring and exception analysis activities for z/OS related resources. The term MVS is being used as a common name for the z/OS operating systems.

Communicate with the MVSDATA Subtask

You can dynamically add or modify the CA SYSVIEW threshold, state, and monitor definitions.

To add or modify these definitions, use online commands or a MODIFY command to the CA SYSVIEW Main Services address space sysview as follows:

```
MODIFY sysview,MODIFY MVSDATA,RELOAD member
```

Checkpoint Function—Request a Checkpoint from MVSDATA

You can write configuration data to the persistent data store using the CHECKPOINT function of the MODIFY command.

To request a checkpoint, add the CHECKPOINT function to the MODIFY command as follows:

```
MODIFY sysview,,MODIFY MVSDATA,CHECKPOINT <ALL|MONITOR|THRESHOLD|STATES>
```

PARMLIB Member MVSDATA

The MVSDATA parmlib member provides default configuration options for the MVS Data Collector Function.

Note: For detailed information regarding the options, see the MVSDATA parmlib member.

PARMLIB Member MVSMON

The MVSMON parmlib member provides monitoring definitions to the MVS Data Collector.

Note: For detailed information regarding the options, see the MVSMON parmlib member.

PARMLIB Member MVSSTATE

The MVSSTATE parmlib member provides resource state monitoring definitions to the MVS Data Collector.

Note: For detailed information regarding the options, see the MVSSTATE parmlib member.

PARMLIB Member MVSTHRSH

The MVSTHRSH parmlib member provides threshold monitoring definitions to the MVS Data Collector.

Note: For detailed information regarding the options, see the MVSTHRSH parmlib member.

Start the MVSDATA Subtask

This statement is only valid in the Main Services Address Space.

The MVSDATA subtask executes as part of the Main Services Address Space. You can find the address space initialization statements in the *sysview*.CNM4BPRM member SYSVIEW.

To start the MVSDATA subtask, enter:

```
START MVSDATA
```

TCP/IP Data Collection Subtask Setup

The TCP/IP Data Collection subtask helps you monitor and manage TCP/IP definitions through basic discovery and viewing of TCP/IP configuration data.

Communicate with the TCPDATA Subtask

You can dynamically add or modify the CA SYSVIEW threshold, state, and monitor definitions.

To add or modify these definitions, use online commands or a MODIFY command to the CA SYSVIEW Main Services address space sysview as follows:

```
MODIFY sysview,MODIFY TCPDATA,RELOAD member
```

Checkpoint Function—Request a Checkpoint from TCPDATA

You can write configuration data to the persistent data store using the CHECKPOINT function of the MODIFY command.

To request a checkpoint, add the CHECKPOINT function to the MODIFY command as follows:

```
MODIFY sysview,,MODIFY TCPDATA,CHECKPOINT <ALL|MONITOR|THRESHOLD|STATES>
```

PARMLIB Member TCPDATA

The TCPDATA parmlib member provides default configuration options for the TCP Data Collector Function.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member TCPMON

The TCPMON parmlib member provides monitoring definitions to the TCP Data Collector.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member TCPSTATE

The TCPSTATE parmlib member provides resource state monitoring definitions to the TCP Data Collector.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member TCPTHRSH

The TCPTHRSH parmlib member provides threshold monitoring definitions to the TCP Data Collector.

Note: For detailed information regarding the options, see the parmlib member.

Start the TCPDATA Subtask

This statement is only valid in the Main Services Address Space.

The TCPDATA subtask executes as part of the Main Services Address Space. You can find the address space initialization statements in the *sysview*.CNM4BPRM member SYSVIEW.

To start the TCPDATA subtask, enter:

```
START TCPDATA
```

Server Address Space Security Access

The TCP feature includes monitoring for Communications Server resources for TCP/IP and Communications Storage Manager (CSM). The server uses the Communications Server Network Management Interface (NMI) to gather CSM data.

Note: Set the VTAM start option SNAMGMT to yes so that the ISTMGCEH subtask is attached to open the NMI.

Security requires superuser authority or access to the following SAF resource:

- Class: SERVAUTH
- Profile or Entity Name: IST.NETMGMT.*sysname*.SNAMGMT
- Access: Read

Superuser authority is either a UID of 0 or READ access to the BPX.SUPERUSER entity of the FACILITY class.

Note: To specify security requirements for TCP/IP and Communication Storage Manager, see the chapter "Interfacing with External Security" in the *Security Guide*.

Note: For information on defining the TCP/IP job names for CA SYSVIEW Option for TCP/IP data collector monitoring, see the TCPMON parmlib member.

VTAM Application Monitor Subtask Setup

The VTAM Application Monitor checks the availability of specified VTAM application IDs.

PARMLIB Member APPLMON

The APPLMON parmlib member provides default configuration options for the VTAM Application Monitor. The task requires the use of a predefined APPLID.

Sample VTAM definitions have been provided in the SYSVAPPL *sysview*.CNM4BSAM member.

Note: For detailed information regarding the options, see the parmlib member.

Start the APPLMON Subtask

This statement is only valid in the Main Services Address Space.

The APPLMON subtask executes as part of Main Services Address Space. Address space initialization statements are found in the *sysview*.CNM4BPRM member SYSVIEW.

To start the APPLMON subtask, enter:

```
START APPLMON
```

CICS Data Logger Subtask Setup

The CICS Data Logger performs the SMF record logging function for the CICS Data Collector subtasks that are executing in each monitored CICS address space. Since the logging of records can require an I/O function to be performed, it is done outside the CICS address space so the CICS address space is not impacted. The records can be logged to SMF as well as internal logs available for online viewing.

The CICS data logger subtask must be active prior to the initialization of any CICS data collector in a CICS address space. If the CICS data collection process is started prior to logger initialization, the data collector will terminate. You can start the data collection manually once the logger initializes. You can use multiple CICS data loggers.

PARMLIB Member CICSLOGR

The CICSLOGR parmlib member provides default configuration options for the CICS Data Logger function.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member CICSCMDS

The CICSCMDS parmlib member is processed immediately after the data logger initialization completes. The member can contain a list of any valid z/OS operator commands. These commands are issued following initialization.

Since the CICS data collectors require the CICS Data Logger to be active prior to their initialization, this member can be used to automatically start your CICS address spaces by inserting the following statements in the CICSCMDS member:

```
)IF SYSNAME=system1
    )IF INACTIVE=cicsprod
        START cicsprod
    )ENDIF
    )IF INACTIVE=cicstest
        START cicstest
    )ENDIF
    )IF INACTIVE=cicsdev
        START cicsdev
    )ENDIF
)ENDIF
```

Note: For detailed information regarding the options, see the parmlib member.

Start the CICSLOGR Subtask

This statement is valid in the Main Services Address Space and the Auxiliary Address Spaces.

The CICSLOGR subtask executes as part of the Main Services Address Space, the Auxiliary Address Spaces, or both. Address space initialization statements are found in the *sysview*.CNM4BPRM members SYSVIEW and SYSVAUX.

When using multiple loggers, a unique task ID must be supplied.

The task ID for the CICSLOGR must be unique within the set SYSVIEW Main and Auxiliary Address Spaces for the specific CA SYSVIEW subsystem and release.

If a task ID is not specified, the default task ID equals the subtask name of CICSLOGR.

To start the CICSLOGR subtask, enter:

`START CICSLOGR.taskID,`*`parameters`*

The values for *parameters* are:

STARTcmds

Specifies that the commands defined in the CICSCMDS member are issued at the end of the subtask initialization. The option overrides the STARTUP-COMMANDS configuration option that is found in the parmlib member CICSLOGR.

NOSTARTcmds

Specifies that the commands defined in the CICSCMDS member *not* be issued at the end of the subtask initialization. The option overrides the STARTUP-COMMANDS configuration option that is found in the parmlib member CICSLOGR.

WebSphere MQ Data Collection Subtask Setup

The WebSphere MQ data collection subtask performs monitoring and exception analysis activities for WebSphere MQ related resources. This subtask can monitor multiple WebSphere MQ Queue Managers.

Communicate with the MQSDATA Subtask

You can dynamically add or modify the CA SYSVIEW threshold, state, and monitor definitions.

To add or modify these definitions, use online commands or a MODIFY command to the CA SYSVIEW Main Services address space sysview as follows:

```
MODIFY sysview,MODIFY MQSDATA,RELOAD member
```

Checkpoint Function—Request a Checkpoint from MQSDATA

You can write configuration data to the persistent data store using the CHECKPOINT function of the MODIFY command.

To request a checkpoint, add the CHECKPOINT function to the MODIFY command as follows:

```
MODIFY sysview,,MODIFY MQSDATA,CHECKPOINT <ALL|MONITOR|THRESHOLD|STATES>
```

PARMLIB Member MQSDATA

The MQSDATA parmlib member provides default configuration options for the WebSphere MQ Data Collector function.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member MQSMON

The MQSMON parmlib member provides resource monitoring definitions to the WebSphere MQ Data Collector.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member MQSSTATE

The MQSSTATE parmlib member provides resource state monitoring definitions to the WebSphere MQ Data Collector.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member MQSTHRSH

The MQSTHRSH parmlib member provides resource threshold definitions to the WebSphere MQ Data Collector.

Note: For detailed information regarding the options, see the parmlib member.

Start the MQSDATA Subtask

This statement is only valid in the Main Services Address Space.

The MQSDATA subtask executes as part of the Main Services Address Space. Address space initialization statements are found in the *sysview*.CNM4BPRM member SYSVIEW.

To start the MQSDATA subtask, enter:

```
START MQSDATA
```

IMS Common Queue Server Subtask Setup

The IMS Common Queue Server subtask (IMSCQS) is the interface between the z/OS Common Queue Server (CQS) and CA SYSVIEW.

The IMSCQS task:

- Issues CQS queries to obtain the information on the following:
  - IMS shared message queues
  - Structures that support a shared messaging environment
- Processes requests for shared message queue data from the CA SYSVIEW IMSQSTAT, IMSQTRAN and IMSTRANS, as well as the IMS Data Collector task.
- Automatically discovers IMS subsystems in a shared messaging environment and connects to the queue structures defined to the IMS.
- Queries the CQS based on the queue sharing group assigned to the current target IMS subsystem when the display command is issued. The target IMS subsystem can be set by selecting an IMS subsystem on the IMSLIST display, issuing the IMS SYSVIEW command or setting the IMSTGTDEF variable.

To use the IMSCQS, you need to have the following configured and implemented:

- CA SYSVIEW Option for IMS
- IMS Common Services Layer
- IMS Common Queue Server

Note: There is no need to start the IMSCQS task because it starts automatically when the SYSVIEW main task starts. The task may also be started or stopped from the ASADMIN display. If no IMS subsystems are participating in IMS queue sharing groups, then removing the task from the SYSVIEW PARMLIB member will prevent it from starting. For more information on configuring IMS Shared Message Queues, see the IBM documentation.

IMS Data Collection Subtask Setup

The IMS Data Collection subtask performs monitoring and exception logging activities for any active IMS subsystem. The task will also monitor the active and inactive status of IMS subsystems to dynamically activate or deactivate an IMS Data Logger subtask for each subsystem.

Communicate with the IMSDATA Subtask

You can dynamically add or modify the CA SYSVIEW threshold, state, and monitor definitions.

To add or modify these definitions, use online commands or a MODIFY command to the CA SYSVIEW Main Services address space sysview as follows:

```
MODIFY sysview,MODIFY IMSDATA,RELOAD member
```

Checkpoint Function—Request a Checkpoint from IMSDATA

You can write configuration data to the persistent data store using the CHECKPOINT function of the MODIFY command.

To request a checkpoint, add the CHECKPOINT function to the MODIFY command using the following format:

```
MODIFY sysview,,MODIFY IMSDATA,CHECKPOINT <ALL|MONITOR|THRESHOLD|STATES>
```

PARMLIB Member IMSDATA

The IMSDATA parmlib member provides default configuration options for the IMS Data Collection function.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member IMSMON

The IMSMON parmlib member provides monitoring definitions to the IMS Data Collector.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member IMSSTATE

The IMSSTATE parmlib member provides resource state monitoring definitions to the IMS Data Collector.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member IMSTHRSH

The IMSTHRSH parmlib member provides threshold monitoring definitions for the IMS Data Collection function.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member IMSTYPE

The IMSTYPE parmlib member provides information about individual IMS log record types and subtypes. This member provides a description for each record type and subtype. Optionally, a program module can be associated with each record type and subtype. The associated program is called to format the record for display.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member IMSCMDS

The IMSCMDS parmlib member is processed immediately after the data collection initialization has been completed. The member can contain a list of any valid z/OS operator commands. These commands are issued following initialization.

Note: For detailed information regarding the options, see the parmlib member.

Start the IMSDATA Subtask

This statement is only valid in the Main Services Address Space.

The IMSDATA subtask executes as part of the Main Services Address Space. Address space initialization statements are found in the *sysview*.CNM4BPRM member SYSVIEW.

To start the IMSDATA subtask, enter:

```
START IMSDATA
```

IMS Data Logger Subtask Setup

The IMS Data Logger performs the transaction data collection for each IMS subsystem as well as the logging of SMF records created by the collector. The records can be logged to SMF as well as to internal logs that are available for online viewing.

The IMS Data Collection subtask (IMSDATA) must be active prior to the initialization of any IMS data logger subtask. The IMS Data Collection subtask will normally start and stop the IMS Data Logger tasks dynamically based on the status of a monitored IMS subsystem. The IMS Data Logger subtasks can also be started and stopped manually.

PARMLIB Member IMSLOGR

The IMSLOGR parmlib member provides default configuration options for the IMS Data Logger function.

Note: For detailed information regarding the options, see the parmlib member.

Start the IMSLOGR Subtask

This statement is only valid in the Main Services Address Space.

The IMSLOGR subtask executes as part of the Main Services Address Space. Address space initialization statements are found in the *sysview*.CNM4BPRM member SYSVIEW.

To start the IMSLOGR subtask, enter:

```
START IMSLOGR.taskid
```

The logger subtask name is specific to the IMS subsystem that is being monitored. The taskid must be defined as IMS*ssid* where *ssid* is the IMS four character subsystem ID.

Example: IMS subsystem ID

The following examples uses PRD1 as the IMS subsystem ID.

```
START IMSLOGR.IMSPRD1
```

IMS SPOC Subtask Setup

The IMS single point of control (IMSSPOC) Server subtask is the interface between CA SYSVIEW and the IMS Common Service Layer (CSL).

The IMSSPOC task does the following:

- Registers CA SYSVIEW with the IMS Common Service Layer SCI as an AOP.
- Forwards IMS type-1 and type-2 commands to IMS.
- Formats command responses for display at the CA SYSVIEW user session.

To use the IMSSPOC, you need configure and implement the following:

- CA SYSVIEW Option for IMS
- IMS Common Services Layer

The IMSSPOC task starts automatically when the SYSVIEW main task starts. You may start and stop the task from the ASADMIN display. If no IMS subsystems are participating in an IMSPLEX or the CSL is not configured, then remove the task from the SYSVIEW PARMLIB member to prevent it from starting.

Note: For more information on configuring IMS Common Service Layer, see the IBM documentation.

XSystem Data Services Subtask Setup

This statement is only valid in the Main Services Address Space.

The XSystem data server provides internal cross-system data communication between CA SYSVIEW tasks operating on different systems. The XSystem data server uses CAICCI as a communications protocol. Cross-system CAICCI definitions must be setup for the XSystem data server to function properly.

PARMLIB Member XSYSTEM

The XSYSTEM parmlib member provides default configuration options for the following functions:

- XSystem Session Services function
- XSystem eXternal Services function
- XSystem data services function

Note: For detailed information regarding the options, see the parmlib member.

Start the XSDS Subtask

The XSDS subtask starts automatically during the Main Services Address Space initialization. There is no need to include a START statement in the *sysview*.CNM4BPRM member SYSVIEW. If an entry is included, an error message is generated indicating that the task has already started.

If the subtask has stopped and needs to be restarted, use the following z/OS MODFIY command to restart the subtask:

```
MODIFY sysview,START XSDS
```

XSystem eXternal Server Subtask Setup

The XSystem eXternal Server provides same-system and cross-system session communication between CA SYSVIEW and other CA products like CA Vantage GMI. Before products like CA Vantage GMI can create a session, the XSXS subtask must be started. The XSystem eXternal Server uses CAICCI as a communications protocol. CAICCI definitions must be setup based on the locations of CA SYSVIEW and the communicating external CA product for the XSystem eXternal Server to function properly.

PARMLIB Member XSYSTEM

The XSYSTEM parmlib member provides default configuration options for the following functions:

- XSystem Session Services function
- XSystem eXternal Services function
- XSystem data services function

Note: For detailed information regarding the options, see the parmlib member.

Start the XSXS Subtask

This statement is only valid in the User Interface Address Space.

The XSXS subtask executes as part of the User Interface Address Space. Address space initialization statements are found in the sysview.CNM4BPRM member SYSVUSER.

To start the XSXS subtask, enter the following:

```
MODIFY SYSVUSER,START XSXS
```

Utility Service Subtask Setup

The UTIL subtask is a utility service that provides system processes.

Start the UTIL Subtask

The UTIL subtask starts automatically during the Main Services Address Space initialization. There is no need to include a START statement in the *sysview*.CNM4BPRM member SYSVIEW. If an entry is included, an error message is generated indicating that the subtask has already started. If the subtask has stopped and needs to be restarted, use the following z/OS MODIFY command to restart the UTIL subtask:

```
MODIFY sysview,START UTIL
```

Communicate with the UTIL Subtask

You can communicate with the UTIL Subtask by modifying the library caching using the EMPTY, DELETE, COMPRESS, or SETCACHE subcommands with the UTIL subtask.

To empty, delete, compress, or set cache

1. Use the EMPTY subcommand to empty the library cache of all members as shown in the following command:

   `MODIFY sysview,MODIFY UTIL,LIBCACHE EMPTY`

2. Use the DELETE subcommand to delete a member from library cache as shown in the following command:

   `MODIFY sysview,MODIFY UTIL,LIBCACHE DELETE type *|member`

   type

   Library type

   member

   Member name

   ***

   Delete all members

3. Use the COMPRESS subcommand to compress the library cache member as shown in the following command:

   `MODIFY sysview,MODIFY UTIL,LIBCACHE COMPRESS type interval`

   type

   Library type or ALL

   Note: Specifying ALL will compress all of the library types.

   interval

   Time interval specified as *hh*:*mm*:*ss*

   ***

   Uses the default interval

   Default: 00:30:00

   Minimum: 00:00:01

   Maximum: 48:00:00

   Any member that has not been accessed within the specified time interval will be removed from the cache.

4. Use the SETCACHE subcommand to set the caching option by library type as shown in the following command:

   `MODIFY sysview,MODIFY UTIL,LIBCACHE SETCACHE type YES|NO`

   type

   Library type

   action

   NO - Do not cache members

   YES - Do cache members

   If the caching option is set to NO and members already exist in the cache, the members will not be retrieved from the cache.

User Interface Address Space Initialization

The User Interface Address Space provides the online display connection for specific interface types. Multiple user interface address spaces can be created if needed. Some users might want to create separate addresses for each interface type.

More information:

User Interface Address Space (see page 30)

PARMLIB Member SYSVUSER

The SYSVUSER parmlib member provides the list of SYSVIEW functions and tasks that are started during initialization. The MEM=parameter on the started task SYSVUSER specifies the parmlib member to be used. The default setting specifies MEM=SYSVUSER.

The special keyword SECONDARY must be specified in this member. This keyword indicates that the current address space is to be a User Interface Address Space.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member OPTIONS Sets Task Configuration Defaults

The OPTIONS parmlib member allows default configuration options be set for all tasks.

Note: For detailed information regarding the options, see the parmlib member.

PARMLIB Member STARTCMD—Start the User Interface Address Space

The STARTCMD parmlib member is processed immediately after the initialization completes. The member can contain a list of any valid z/OS operator commands. These commands are issued following initialization.

After the Main Services Address Space initializes, it can automatically start the User Interface Address Space by adding the following statements to the STARTCMD member:

```
)IF JOBNAME=SYSVIEW
    )IF INACTIVE=SYSVUSER
        START SYSVUSER
    )ENDIF
)ENDIF
```

Note: For detailed information regarding the options, see the parmlib member.

Start the SYSVUSER Started Address Space

The User Interface Address Space is started from a system console by entering the command:

```
START SYSVUSER
```

Communicate with the SYSVUSER Started Subtask

After starting the user interface subtask, communicate additional requests to the subtask using the z/OS MODIFY (F) command. You can also use the online Address Space Administration command, ASADMIN, to perform additional functions.

The following z/OS modify (F) commands and functions are available:

STATUS

Displays the status of all subtasks or a particular subtask in the user interface address space.

`MODIFY sysvuser,STATUS [`*`subtask`*`]`

subtask

Specifies the name of the subtask for the requested status display.

START

Starts a subtask in the user interface address space.

```
MODIFY sysvuser,START subtask [.taskident, parameters]
```

subtask

Specifies the subtask name to start.

taskident

Identifies a task, if you are starting multiple tasks with the same subtask name.

parameters

Represents the parameters to be passed to the subtask. If the parameters contain a comma or a space, it must be enclosed in single quotes.

STOP

Displays the status of all subtasks or a particular subtask in the user interface address space.

```
MODIFY sysvuser,STOP [subtask]
```

subtask

Specifies the name of the subtask for the requested status display.

MODIFY

Communicates with a subtask in the user interface address space.

```
MODIFY sysvuser,MODIFY subtask/taskid [, parameters]
```

subtask

Specifies the name of the subtask to modify.

parameters

Represents the parameters to be passed to the subtask. If the parameters contain a comma or a space, it must be enclosed in single quotes.

FORCE

Forces a subtask in the user interface address space to terminate abnormally.

```
MODIFY sysvuser,FORCE subtask/taskid
```

subtask

Specifies the name of the subtask that you wish to abnormally terminate. The task is canceled with a dump.

User Interface Address Space Termination

The User Interface Address Space can run continuously for the duration of an IPL. The address space can be terminated and restarted at any time if needed. All subtasks executing in the user interface address are stopped automatically.

PARMLIB Member STOPCMD

The STOPCMD parmlib member is processed immediately at the start of the termination process. The member can contain a list of any valid z/OS operator commands. These commands are issued before termination. Detailed information regarding the option can be found in the parmlib member.

Note: For detailed information regarding the options, see the parmlib member.

Stop the SYSVUSER Started Task

The User Interface Address Space can be terminated using the z/OS STOP (P) command. The task can be stopped from a system console by entering:

```
P SYSVUSER
```

Event Capture Subtask Setup

Analyzing historical and captured event data is an important part of maintaining your system. To analyze a problem, you must first have data to analyze. The Event Capture option gives you a set of tools that enables the capturing of the critical data that you need. The option provides out-of-the-box data gathering plus the capability to customize fully the process.

Configure the Event Capture option even when your site does not have license. CA technical support requests Event Capture data for support issues.

System Configuration Options Member

The following keywords are used in the System Configuration Options member to set up the Event Capture option:

OPTIONS-Event_Capture

Specify a value of Yes to enable the Event Capture option for use. It does not start the function.

Dsn-System-CAPINDEX

Specifies the capture index data set name. This data set can contain symbolic names, such as &SYSNAME or &SMFID.

Dsn-System-CAPLIB

Specifies the capture event definition library data set name.

Dsn-System-CAPDATA-HLQ

Specifies the capture data high-level qualifier. The high-level qualifier can contain a maximum of 29 characters to provide room to append the date and time qualifiers. The Event data set names are constructed as follows:

capdhlq.Dyymmdd.Thhmmss

Dsn-System-CAPINDEX-HLQ

Specifies the capture index high-level qualifier.

PARMLIB Member CAPTURE

The CAPTURE parmlib member provides default configuration options for the Event Capture function.

Note: For detailed information regarding the options, see the parmlib member.

Event Capture CAPLIB Members

Specify individual capture events in the Capture Library (CAPLIB) members found in the sysview.CNM4BCAP data set. Each CAPLIB member can override the default configuration options.

Start the Event Capture Subtask

This statement is only valid in the User Interface Address Space.

The following statement starts the CAPTURE controller subtask:

```
START CAPTURE
```

CICS User Interface Subtask Setup

The CICS User Interface subtask enables users to log on to CA SYSVIEW from a CICS session. Before a user can access CA SYSVIEW from a CICS session, the CICS User Interface subtask must be started.

Logging on Using CICS

To use CA SYSVIEW with CICS, enter the CICS transaction SYSV.

Cancel Users from the CICS Interface

Users can be canceled from the CICS interface by using the MODIFY command.

To cancel user ABC from the CICS interface, enter the following z/OS command:

```
F SYSVUSER,MODIFY CICS,'CANCEL ABC'
```

Be sure to enclose the CANCEL command in single quotes.

Start the CICS User Interface Subtask

This statement is only valid in the User Interface Address Space.

The CICS task executes as part of the User Interface Address Space. Address space initialization statements are found in the *sysview*.CNM4BPRM member SYSVUSER.

To start the CICS subtask, enter the following:

```
START CICS,parameters
```

The value for *parameters* is:

MAXUSERS=*nnn*

Controls the maximum number of users that can use the CICS interface. The MAXUSERS parameter must be a number from 1 to 500.

Default: 50

VTAM Interface Subtask Setup

The VTAM subtask executes as part of the User Interface Address Space. Before you can use CA SYSVIEW with VTAM, you must start the VTAM User Interface subtask.

Sample VTAM definitions are provided in the SYSVAPPL *sysview*.CNM4BSAM member.

Logging on from VTAM

To use CA SYSVIEW with VTAM, enter SYSVIEW. If an application ID other than SYSVIEW was specified for the APPLID parameter, this ID must be used when logging on to CA SYSVIEW.

Use the format for the SYSVIEW command for VTAM:

```
SYSVIEW [DATA(userid)]
```

DATA(*userid*)

Supplies the user ID when logging on to CA SYSVIEW. If you omit the DATA parameter, CA SYSVIEW prompts for your user ID.

Note: Because a user can enter anything for a user ID, including the user ID of an administrator, it is important that passwords be used to secure user IDs with special authority.

If you have a password, or are using the VTAM interface, a User Identification display appears when you type in SYSVIEW. You must enter your user ID and a password, if required. You have three chances to enter the correct password before the session is terminated.

Pressing any PF key from this display terminates the session.

Cancel Users from the VTAM Interface

Users can be canceled from the VTAM interface by using the MODIFY command. For example, to cancel user ABC from the VTAM interface, enter the following z/OS command:

```
F SYSVUSER,MODIFY VTAM,'CANCEL ABC'
```

Be sure to enclose the CANCEL command in single quotation marks.

Start the VTAM Subtask

This statement is only valid in the User Interface Address Space.

The VTAM task executes as part of the User Interface Address Space. Address space initialization statements are found in the *sysview*.CNM4BPRM member SYSVUSER.

To start the VTAM subtask, enter the following:

```
START VTAM,parameters
```

The values for *parameters*:

APPLID=*applid*

Controls the VTAM application ID. This is the name used when logging on through VTAM.

Default application ID: SYSVIEW

MAXUSERS=*nnn*

Controls the maximum number of users that can use the VTAM interface. The MAXUSERS parameter must be a number from 1 to 500.

Default: 50

XSystem Session Server Subtask Setup

The XSystem Session Server provides cross-system session communication between CA SYSVIEW tasks operating on different systems. Before a user can create a cross-system session, the XSSS subtask must be started. The XSystem Session Server uses CAICCI as a communications protocol. Cross-system CAICCI definitions must be setup for the XSystem Session Server to function properly.

PARMLIB Member XSYSTEM

The XSYSTEM parmlib member provides default configuration options for the following functions:

- XSystem Session Services function
- XSystem eXternal Services function
- XSystem data services function

Note: For detailed information regarding the options, see the parmlib member.

Start the XSSS Subtask

This statement is only valid in the User Interface Address Space.

The XSSS subtask executes as part of the User Interface Address Space. Address space initialization statements are found in the *sysview*.CNM4BPRM member SYSVUSER.

To start the XSSS subtask, enter the following:

```
MODIFY SYSVUSER,START XSSS
```

Chapter 3: Starting the Product Server and Its Subtasks

This chapter explains how to start CA SYSVIEW and describes the subtasks. The Main Services Address Space and the User Interface Address Space of CA SYSVIEW control the subtasks.

You can further customize these subtasks using the appropriate PARMLIB members.

Once the address spaces have been started, we recommend using the online address space Administration command, ASADMIN, to perform additional functions.

This section contains the following topics:

Main Services Address Space

The Main Services address space provides common services used by other CA SYSVIEW tasks and users. The main task controls the starting and stopping of other subtasks. Start the Main Services address space as early as possible during IPL.

How the Main Services Address Space Is Initialized

Before you start the main services address space, review how the initialization information is obtained.

Initialization performs the following actions:

- The primary initialization information is obtained from the System Configuration Options member in the system parameter library SYS1.PARMLIB.
- The System Configuration Options member contains the following information:
  - Run-time library names
  - Enabled features
- A System Configuration Options member is defined for each z/OS system.

To run multiple copies of CA SYSVIEW on a single system, review these rules:

- Multiple copies of the same release must specify different subsystem IDs.
- Multiple copies of different releases can use the same subsystem ID.

Main Services Address Space Subtasks

The Main Services address space starts the CA SYSVIEW Server subtasks.

You can start the following subtasks:

- Event Scheduler
- IDCAMS service
- Job information collection
- Syslog output file information
- CICS data logger

PARMLIB Member OPTIONS

The OPTIONS parmlib member allows default configuration to be set for all subtasks.

Note: For detailed information regarding the options, see the OPTIONS parmlib member.

PARMLIB Member STARTCMD

The STARTCMD parmlib member processes immediately after the initialization completes. The member contains a list of valid z/OS operator commands. These commands are issued following initialization.

Note: For detailed information regarding the options, see the STARTCMD parmlib member.

Communicate with the Started Task

Once the main services task has started, additional requests can be communicated to the task using the z/OS MODIFY (F) command.

The following commands and functions are available.

STATUS

Displays the status of all subtasks or a particular subtask in the Main Services Address Space. The format of the STATUS command is:

```
MODIFY SYSVSERV,STATUS [subtask]
```

subtask

Specifies the name of the subtask for the requested status display.

START

Starts a subtask in the Main Services Address Space. The format of the START command is:

```
MODIFY SYSVSERV,START subtask[.taskident,parameters]
```

subtask

Specifies the name of the subtask to start.

taskident

Identifies a task, if you are starting multiple tasks in the same subtask.

parameters

Represents the parameters to pass to the subtask.

STOP

Stops a subtask in the Main Services Address Space. The format of the STOP command is:

```
MODIFY SYSVSERV,STOP subtask|taskid
```

subtask

Specifies the name of the subtask you want to terminate.

taskid

Identifies a task, if you are starting multiple tasks in the same subtask.

MODIFY

Modifies current parameter settings for a subtask in the Main Services Address Space. The format of the MODIFY command is:

```
MODIFY SYSVSERV,MODIFY subtask|taskid [, parameters]
```

subtask

Specifies the name of the subtasks to modify.

taskid

Identifies a task, if you are starting multiple tasks in the same subtask.

parameters

Represents the parameters to pass to the subtask. If the parameters contain a comma or a space, it must be enclosed in single quotes.

FORCE

Forces the subtask in the Main Services Address Space to terminate abnormally. The format of the FORCE command is:

```
MODIFY SYSVSERV,FORCE subtask|taskid
```

subtask

Specifies the name of the subtask you want to terminate abnormally. The task is canceled with a dump.

taskid

Identifies a task, if you are starting multiple tasks in the same subtask.

Main Services Address Space Termination

The Main Services address space can run continuously for the duration of an IPL. You can terminate and start the address space at any time. When you terminate the address space, all subtasks executing in the address space are stopped automatically.

PARMLIB Member STOPCMD

The STOPCMD parmlib member contains a list of valid z/OS operator commands. The commands process at the start of termination.

Note: For detailed information, see the STOPCMD parmlib member.

Stop the SYSVSERV Started Task

You can terminate the Main Services address space without having to perform an IPL.

To terminate the Main Services address space, enter the following z/OS STOP (P) command from a system console:

```
P SYSVSERV
```

Event Scheduler Subtasks Setup

Initiating processes and functions based on a predefined time schedule is an important part of operating a computer system. Tasks are performed every morning, noon, and midnight. CA SYSVIEW provides an Event Scheduler that is able to initiate many types of functions at a specified time or recurring interval. Many events can be initiated in CA SYSVIEW and outside its boundaries. The Event Scheduler subtask executes in the Main Services Address Space and is started automatically during initialization.

PARMLIB Member SCHEDULE

The SCHEDULE parmlib member provides configuration options and defines scheduled events for the subtask SCHEDULR.

Note: For detailed information regarding these options and sample event definitions, see the SCHEDULE parmlib member.

Start the SCHEDULR Subtask

This statement is only valid in the Main Services Address Space.

The SCHEDULR subtask starts automatically when the Main Services Address Space initializes. Configuration options and event definitions are defined in the *sysview.*CNM4BPRM member SCHEDULE.

If the subtask has stopped and needs restarted, use the following z/OS MODIFY command to restart the subtask:

```
MODIFY SYSVSERV,START SCHEDULR
```

IDCAMS Service Subtask Setup

The IDCAMS Service subtask performs Access Method Services (AMS) requests. These requests need performed in the main services task due to authorization requirements. The IDCAMS Service subtask starts automatically when the Main Services address space initializes.

Start the AMS Subtask

This statement is only valid in the Main Services Address Space.

The AMS subtask starts automatically when the Main Services Address Space initializes.

If the subtask has stopped and needs restarted, use the following z/OS MODIFY command to restart the subtask:

```
MODIFY SYSVSERV,START AMS
```

JOBS Information Subtask Setup

The JOBS subtask collects job summary information displayed by the JOBSUM and LISTINP display commands. The task starts automatically when the Main Services Address Space initializes. This task must be active or the JOBSUM and LISTINP display commands do not display any information. The information collected by the JOBS task improves the performance of the JJOBQUE, JHELDQUE, and JOUTQUE display commands.

When the user issues a JOBSUM display command, the information is obtained from the Main Services Address Space. Because the JOBS subtask periodically saves the job summary information, the time it takes to collect the information is greatly reduced.

JES Job Queue Updates

Periodically, the JOBS subtask inspects the status of jobs on the JES queues. If the information for the jobs on the JES queues has changed, it updates the information.

Collect the Highest Completion Code Information

The JOBS subtask collects highest completion code information about jobs on the output queue. This information displays on the Job Summary display (produced by the JOBSUM command) in the CCODE field. To bypass the collection of the information, use the NOCCODE parameter when starting the JOBS subtasks.

To turn on the completion code function, enter the following z/OS command:

```
F SYSVSERV,MODIFY JOBS,CCODE
```

To turn off the completion code function, enter the following z/OS command:

```
F SYSVSERV,MODIFY JOBS,NOCCODE
```

Set the Update Interval

The range for wait-time interval is 1 - 9999 seconds. The JOBS subtask waits 10 seconds and then verifies if anything has changed. This wait time assures that the information displayed by the JOBSUM command is no more than 10 seconds old. Also, when using the JOBSUM command, the information is updated every time you enter the REFRESH command on the Job Summary display. Use the MODIFY command to make a dynamic change to the wait-time interval.

To change the interval to 60 seconds, enter the following z/OS command:

F SYSVSERV,MODIFY JOBS,INTERVAL=60

Default interval: 10 seconds

Start the JOBS Subtask

This statement is only valid in the Main Services Address Space.

The JOBS subtask executes as part of Main Service Address Space.

To start the JOBS subtask, enter the following command:

```
START JOBS,parameters
```

Where *parameters*:

INTERVAL=*nnn*

Specifies the update interval as the number of seconds from 1 to 9999.

Default interval:10 seconds

CCODE

Indicates to collect highest completion code information for jobs on the output queue.

Default: CCODE

NOCCODE

Indicates do not collect highest completion code information for jobs on the output queue.

Default: CCODE

CICS Data Logger Subtask Setup

The CICS data logger subtask must be active before the initialization of any CICS data collector in a CICS address space. When the CICS data collection process starts before logger initialization, the data collector terminates. You can start the data collection once the logger has been initialized. Multiple CICS data loggers can be used.

PARMLIB Member CICSLOGR

The CICSLOGR parmlib member provides default configuration options for the CICS Data Logger function. Detailed information regarding the options can be found in the parmlib member.

PARMLIB Member CICSCMDS

The CICSCMDS parmlib member processes immediately after the data logger initialization completes. The member contains a list of valid z/OS operator commands that are issued following initialization. Detailed information regarding the options can be found in the parmlib member.

The CICS Data Logger must be active before the CICS data collectors can initialize. Once initialized, start your CICS address spaces automatically by inserting the following statements in the CICSCMDS member:

```
)IF SYSNAME=system1
    ) IF INACTIVE=cicsprod
         START cicsprod
    ) ENDIF
    ) IF INACTIVE=cicstest
         START cicstest
    ) ENDIF
    ) IF INACTIVE=cicsdev
         START cicsdev
    ) ENDIF
) ENDIF
```

Start the CICSLOGR Subtask

This statement is only valid in the Main Services Address Space.

The CICSLOGR subtask executes as part of the Main Services Address Space.

When using multiple loggers, a unique task ID must be supplied. If a task ID is not specified, the default task ID equals the subtask name of CICSLOGR.

To start the CICSLOGR subtask, enter:

`START CICSLOGR.taskID,`*`parameters`*

Where *parameters*:

STARTcmds

Specifies to issue the commands defined in the CICSCMDS member at the end of the subtask initialization. The option overrides the STARTUP-COMMANDS configuration option that is found in the parmlib member CICSLOGR.

NOSTARTcmds

Specifies *not* to issue the commands defined in the CICSCMDS member at the end of the subtask initialization. The option overrides the STARTUP-COMMANDS configuration option that is found in the parmlib member CICSLOGR.

Chapter 4: Customization Information

This chapter contains product customization information.

This section contains the following topics:

GENERAL Section of the DEFAULT Profile

The GENERAL section in the DEFAULT profile contains allocation parameters that are used when CA SYSVIEW allocates a data set. You can change these allocation parameters to conform to the installation standards.

For more information about changing the DEFAULT profile, see the *Security Guide*.

Default Security Group

The security data set on the installation tape contains a security group that defines the default security. This security group is known as the DEFAULT security group.

The DEFAULT security group defines which commands anyone can use when they are not defined as a member of a general security group.

The administrator reviews the DEFAULT security group before using CA SYSVIEW to determine whether the default security meets the approval of your installation.

Note: For detailed information, see the *Security Guide*.

The Command Exit

The command exit can modify commands entered on the command line of the CA SYSVIEW product. Commands you enter with a PF key can also be modified.

CA SYSVIEW determines whether the command being entered is one of the following types before it passes the command to the command exit:

- CA SYSVIEW command
- subcommand
- unknown command

The command exit can alter the command before CA SYSVIEW acts on it or discards it. The command exit lets you:

- Change the command format of a CA SYSVIEW command.
- Change a command from another software package to a command for CA SYSVIEW.

Location of the Command Exit

The source for the default command exit module is located in the GSVXCMDX member in the *sysview*.CNM4BSAM data set.

How the Command Exit Works

The following list explains how the command exit works:

- A user enters a command on the command line in CA SYSVIEW.
- CA SYSVIEW determines the type of command entered.
- CA SYSVIEW sets register 0 to the command type, sets register 1 to the parameter list, and calls the command exit.
- The command exit looks at and changes the command entered by the user, if necessary. The exit then returns to CA SYSVIEW.
- CA SYSVIEW receives and executes the command or subcommand.

More information:

Register Settings for Indicating Type of Command (see page 83)

Assemble and Link the Exit

The member ASMCMDX in *sysview*.CNM4BSAM contains the JCL necessary to assemble and link the GSVXCMDX module.

Register Settings for Indicating Type of Command

CA SYSVIEW uses register 0 to indicate the type of command it has passed.

The possible settings of register 0 are described in the following table:

Register Setting	Type of Command
0	CA SYSVIEW primary command
4	CA SYSVIEW subcommand
8	Unknown command

Register 1 Settings

Register 1 points to a list of addresses. The following list describes what is contained in each address.

- First Address

  The first address in the parameter list points to the reply buffer.

  The format of the reply buffer is a halfword containing the length of the reply, followed by the reply. The reply can be 1 - 256 bytes long.

  If the command exit updates the reply buffer, the length of the reply is placed in the halfword pointed to by the reply buffer address. The reply is placed after the halfword containing the length and is limited to 256 bytes.

- Second Address

  The second address in the parameter list points to a 256-byte temporary work area.

- Third Address

  The third address in the parameter list points to the full eight-character command name if the command was one of the CA SYSVIEW primary commands.

  For example: A user has a synonym of LOG for the SYSLOG command and issues the LOG command, then this parameter contains the characters *SYSLOG*.

  If the command entered is a subcommand of the active command, the parameter contains the full eight-character subcommand name.

  If the command entered is an unknown command, the contents of this parameter are unpredictable.

- Fourth Address

  The fourth address in the parameter list points to the parameters entered with the command.

  The first halfword contains the length of the parameters. The parameters follow the halfword.

- Fifth Address

  The fifth address in the parameter list points to the default parameters for a command when the following conditions are true:

  - If the command was one of the CA SYSVIEW primary commands
  - If there are any default parameters

  The format of the default parameters is a halfword containing the length of the parameters, followed by the parameters. If there are no default parameters, the length field contains 0.

  Users define default parameters in their profiles.

- Sixth Address

  The sixth address in the parameter list points to the 8-byte user ID of the user who issued the command.

- Seventh Address

  The address in the seventh parameter points to a byte describing the application where CA SYSVIEW is running. The bits in the byte have the meanings shown in the following table:

Setting	CA SYSVIEW Running
X'80'	Under TSO
X'40'	Under CICS
X'20'	As a VTAM application
X'08'	Under ISPF

X'04'	As a batch program
X'02'	Under ROSCOE/ETSO
X'01'	Through the application program interface (API)

- Eighth Address

  The eighth address in the parameter list points to a 4-byte area. This area can be used to save the address of storage acquired by the command exit. This 4-byte area is cleared to binary zeros before calling the exit for the first-time.

- Ninth Address

  The ninth address in the parameter list points to a fixed-length mask character.

- Tenth Address

  The tenth address in the parameter list points to a variable-length mask character.

Settings of Additional Registers

The additional registers used with the command exit are explained in the following table.

Register	Description
13	Points to a save area to store the registers on entry to the command exit
14	Contains the return address
15	Contains the entry point address of the command exit

Cache System Owned Libraries

This optional product feature can cache members that reside in system owned libraries. System owned libraries are defined in the System Configuration Options member.

Caching library members provides the following benefits:

- Faster read operation.

  When a library member is cached, subsequent read requests for the same member do not require I/O or data set allocation. When the read request is satisfied from the cache, the read operation is faster because the I/O is eliminated.

- Shorter session initialization time.
- Enhances the use of the REXX API interface.

How Library Cache is Maintained

The library cache basically maintains itself.

- When a read request is received for a member, the library cache is reviewed to determine whether it currently contains the member.
  - If the member is found in the cache, its contents are returned from the cache.
  - If the member is not found in the cache, the member is read from the data set and stored into the cache.
- The library cache is emptied each time the CA SYSVIEW Main Services Address Space is started.

View List of Eligible Libraries

Issue the LIBS command to view the list of libraries.

The following library types are eligible to be cached:

Library	Description
CAPLIB	Contains capture scripts
CLISTLIB	Contains command lists
HELPLIB	Contains product online help system
MAPLIB	Contains control block maps
MIBLIB	Contains management information block (MIB) structures
PANELLIB	Contains menu panels
PARMLIB	Contains parameter definitions
PLOTLIB	Contains predefined plots

View Library Cache Contents

You can view the library cache by issuing the following command:

```
LIBCACHE
```

If you do not want to participate in library caching, set your profile option accordingly:

```
SET LIBCACHE YES|NO
```

Compress Members from Library Cache

Compress the library cache to remove those members that are not actively being used.

To compress unused members, use any of the following methods

- Use the LIBCACHE COMPRESS subcommand as follows:

  `COMPRESS type interval`

- Issue a console operator command as follows:

  `MODIFY sysview,MODIFY UTIL,LIBCACHE COMPRESS type interval`

- Use a CA SYSVIEW Scheduler event to compress the library cache automatically. See the example provided in *sysview*.CNM4BPRM(SCHDLIBC).

Remove All Member from Cache

To remove all members from cache, use any of the following methods:

- From the LIBCACHE command, enter the DELETE line command on the desired members. Use the FILL command or block line commands to enter the DELETE line command on multiple lines.
- Use the LIBCACHE EMPTY subcommand as follows:

  `EMPTY`

- Issue a console operator command as follows:

  `MODIFY sysview,MODIFY UTIL,LIBCACHE EMPTY`

Delete a Member from Cache

When caching is active, a read request is satisfied from the cache. After you update a library member, delete the member from the active cache so you can use the new or updated member.

To delete a member from cache, use any of the following methods

- From the LIBCACHE command, enter the DELETE line command on the desired member.
- Use the LIBCACHE DELETE subcommand as follows:

  `DELETE type member`
- Issue a console operator command as follows:

  `MODIFY sysview,MODIFY UTIL,LIBCACHE DELETE type member`

Note: The member will be reloaded into the cache the next time the member is requested.

Set System Owned Libraries for Caching

The parmlib member *sysview*.CNM4BPRM(OPTIONS) contains the initial configuration settings for library caching.

To set which library types to cached, use any of the following methods

- Use the following configuration options to control the caching of each library type:

LibCache-Caplib	Yes
LibCache-Clistlib	Yes
LibCache-Helplib	Yes
LibCache-Maplib	No
LibCache-Miblib	No
LibCache-Panellib	Yes
LibCache-Parmlib	Yes
LibCache-Plotlib	Yes

- Use the LIBCACHE SETCACHE subcommand as follows:

  `SETCACHE type YES|NO`
- Issue a console operator command as follows:

  `MODIFY sysview,MODIFY UTIL,LIBCACHE SETCACHE YES|NO`

SDSFMIGRATE Option Environment

The SDSFMIGRATE option lets users work in an environment similar to SDSF while they are becoming accustomed to the CA SYSVIEW environment and command structure. After they work with CA SYSVIEW, users can turn off SDSFMIGRATE option and experience the full potential and flexibility of the environment.

Activate SDSFMIGRATE to Migrate from SDSF

CA SYSVIEW provides an SDSFMIGRATE option to help you migrate from the IBM SDSF product to CA SYSVIEW.

The SDSFMIGRATE option does the following:

- Defines a Primary Option Menu that contains a menu option that shows SDSF commands
- Allows users to enter most SDSF primary commands
- Allows users to enter most SDSF line commands

The SDSF Primary Option Menu is located in the MENUSDSF member of the *sysview*.CNM4BPNL data set.

To activate the SDSFMIGRATE option

- A user can enter the SET SDSFMIGRATE ON command.
- A CA SYSVIEW administrator can change the DEFAULT profile when the product is first installed.

Change the DEFAULT Profile Member

Only CA SYSVIEW administrators can change the DEFAULT profile member.

To change the SDSFMIGRATE option in the DEFAULT profile member, enter the following commands:

- PROFILE CHANGE DEFAULT
- SET SDSFMIGRATE ON

Masking Characters for the SDSFMIGRATE Option

When the SDSFMIGRATE option is turned on:

- The fixed-length masking character is set to a percent sign (%)
- The variable-length masking character is set to an asterisk (*)

When you turn the option off, these settings remain the same; they are not reset to the CA SYSVIEW defaults.

The SDSF Command Exit

The SDSF command exit:

- Translates SDSF commands to the corresponding commands in this product
- Lets users enter most SDSF primary commands

The source for the default exit is located in the GSVXSDSX member of the *sysview*.CNM4BSAM data set. The JCL in member ASMSDSFX in the *sysview*.CNM4BSAM data set can be used to assemble the exit when changes are necessary.

Note: Modifications to the MENUSDSF panel and the GSVXSDSX module are the responsibility of the site.

Chapter 5: Customize the Components, Options, and Monitoring

You can perform several customization tasks after installation to let the components, options, and monitoring perform to your specific instructions.

This section contains the following topics:

Define CA Datacom Address Spaces for Monitoring

The list of CA Datacom/DB address spaces that CA SYSVIEW uses for monitoring is obtained using the following methods:

- Dynamically from a list of active jobs
- From a list of jobs, if any, that are defined in the DATACOM PARMLIB member of CA SYSVIEW

To define address spaces in the DATACOM Names Table, follow the instructions in the DATACOM PARMLIB member.

Defining CA Datacom jobs in the DATACOM PARMLIB member is not required. When you do not define CA Datacom jobs in this member, you can view only the status of active CA Datacom address spaces. To view the status of active and inactive CA Datacom jobs, define those jobs in the DATACOM Names Table in the DATACOM PARMLIB member. CA SYSVIEW indicates the status of all CA Datacom jobs defined in the DATACOM PARMLIB member, even if they are inactive.

Note: The DATACOM PARMLIB member contains other relevant information about the CA SYSVIEW CA Datacom Option.

Define CA Insight DPM for DB2 Connections

A site CA Insight DPM for DB2 license permits you to define the component CA SYSVIEW for CA Insight DPM for DB2. This connection lets you display data for monitored DB2 subsystems.

Follow these steps:

1. Access the DB2 parmlib member.
2. Review the configuration instructions in DB2 parmlib.
3. Update the following two parameters and base your updates on the settings in Xnet:
   - XNET-PassTicketApplId
   - XNET-Port

   These parameters are maintained in DB2 parmlib.

   Notes:

   - Be sure that CA Insight DPM for DB2, CA DB2 Tools Xmanager, and CA DB2 Tools Xnet are installed and configured and CA DB2 Tools Xnet is started.
   - Ensure you configured CA DB2 Tools Xnet to use PassTickets and the required PassTicket setup has occurred with the External Security package. For more information, see the *Security Guide*.

Your CA Insight DPM for DB2 connections are defined.

Install and Start the CA Roscoe Monitor

After you install and start the CA Roscoe monitor, you can display information about executing its jobs.

Follow these steps:

1. Add the following line to the CA Roscoe startup parameters for each CA Roscoe job to monitor:

   ```
   RUN=GSV
   ```

2. Add the CA SYSVIEW load library as a STEPLIB and a TASKLIB to the CA Roscoe jobs to monitor.

   The monitor routine becomes accessible to CA Roscoe.

   This step makes the GSVRRMIR module and its alias RSSCGSV0 accessible to these jobs.

3. Execute the GSV monitor by entering **GSV** from the CA Roscoe job to monitor using the following parameters for the GSV monitor:

   START

   (Default) Starts the monitor.

   STOP

   Stops the monitor.

   STATUS

   Displays the status of the monitor.

   Note: The GSV monitor routine must be executed once after CA Roscoe startup to start the monitoring process.

   Alternatively, you can also execute the GSV monitor by adding an entry to the ROSCOE ROSXINIT module that automatically starts the GSV monitor. The format of the entry is:

   ```
   Label ROSX GSVRGSVI,MON
   ```

   Label

   Any label or it can be omitted. Similarly, an entry needs added to ROSCOE ROSXTERM module that automatically stops the GSV monitor when ROSCOE is shut down. The format of the entry is:

   ```
   Label ROSX GSVRGSVT,MON
   ```

   Label

   Any label or it can be omitted.

   Note: For more information, see the *CA Roscoe Interactive Environment Extended Facilities System Programmer's Guide*.

   Note: Member ROSXINIT must be assembled into the CA Roscoe load library using USERMOD MRO601B that is in the CA Roscoe SAMPJCL.

   The monitor executes.

4. Add the following line to the ROSCOE startup parameters for each job to monitor:

   ```
   RTM=YES
   ```

   The CA Roscoe response times can be viewed.

After monitoring is started, you can use the ROSLIST command to display the status of all monitored CA Roscoe jobs.

Set Up the VTAM Application Monitor

The VTAM Application Monitor task APPLMON monitors VTAM applications from the main product address space. You can use this feature after setting up the VTAM application monitor.

Follow these steps:

1. Set startup parameters for APPLMON in the CA SYSVIEW VTAM Application Monitor Table in the APPLMON parameter library member.
2. Review the instructions and details are provided there for defining the following startup parameters:

   APPLID *name*

   Defines the VTAM APPL name that the APPLMON task uses to communicate with VTAM.

   Note: This statement must be supplied.

   INTERVAL *secs*

   Defines the monitor interval.

   MONITOR *name descr*

   Defines the name of a VTAM application to monitor.

   Note: Supply one MONITOR statement.
3. Issue the APPLMON command to view a summary of VTAM applications monitored by APPLMON.
4. On the VTAM Application Monitor display, add or delete applications from the table of monitored applications.

System Overview Information Area in Command and Menu Displays

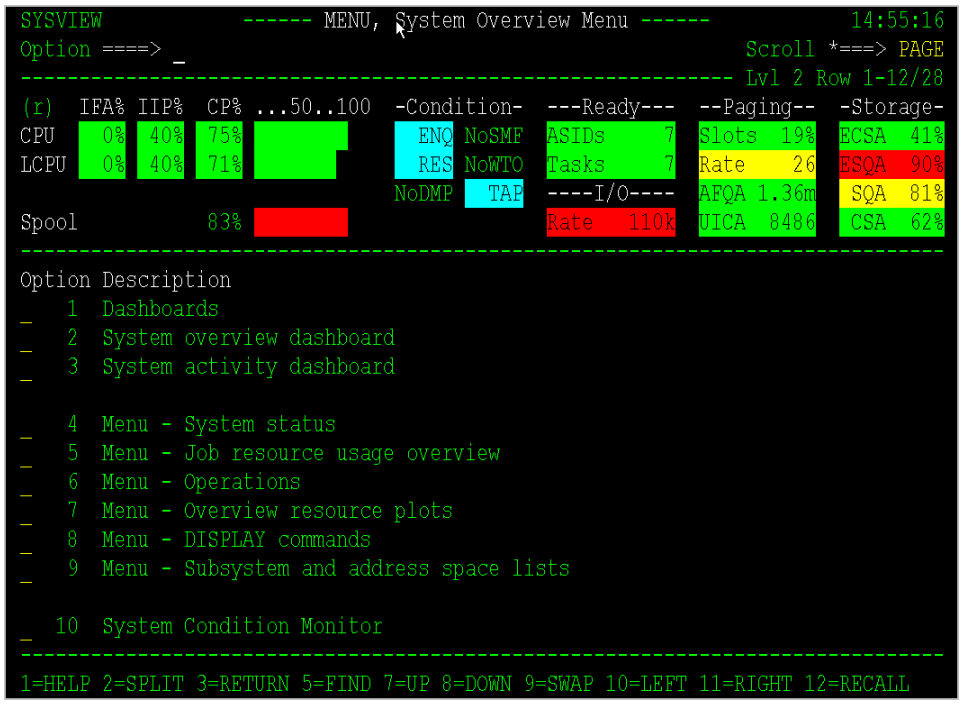
The System Overview component lets you display and modify a select group of metrics and conditions on the present status of your z/OS system.

You can display this package of information as follows:

- In the information area of CA SYSVIEW command and menu displays that are defined as overview capable
- In a cross system view showing multiple z/OS images by issuing the SYSTEMS command

Sample Overview Information Area

The following sample screen demonstrates how the System Overview information displays with easy drill-down capabilities.



Each menu optionally includes an information section keyword that requests the display of the System Overview information.

CNM4BPNL(OVERVIEW)—Define the Overview Menu

The *sysview*.CNM4BPNL(OVERVIEW) data set defines the overview menu system.

Example: *sysview*.CNM4BPNL(OVERVIEW) data set

```
....+....1....+....2....+....3....+....4....+
TSystem Overview Menu
IOverviewRealtimeLong
HOption Description
 _      Menu - System status
 _      Menu - Job resource usage overview
 _      Menu - Overview resource plots
```

The format of the data is as follows:

Column 01	Panel Line Type
T	Menu title - Only one title record is allowed and must precede all other record types.
I	Info line (col 2:12 is an info keyword) - Up to six information records are allowed and they must follow any title record and precede any header or data records.
H	Header line - Up to four header records are allowed and they must follow any title record and info records and precede any data records.
F	Data header (floating header line) - Data header records are scrollable data records that are displayed with the screen header line attribute.
A	Highlighted data line with input fields
blank	Data line with input fields
E	Highlighted data line without input fields
D	Data line without input fields
*	Comments

Control the Display of Realtime and Interval Data

The following information line keywords control the display of real-time and interval data:

- OverviewRealtime
- OverviewInterval
- OverviewRealtimeLong (same as OverviewRealTime)
- OverviewIntervalLong (same as OverviewInterval)
- OverviewRealtimeShort
- OverviewIntervalShort

Chapter 6: Data Collection

The CA SYSVIEW data collector components collect, monitor, and provide exception processing for many resources and metrics.

In some sites, users decide they do not want to collect information about all possible resources and metrics. In this situation, the collection of unwanted or unneeded metrics is a waste of important system resources and CPU cycles.

You can control what is collected and how often it is collected using the Event Scheduler.

Controlling the data collection metrics can:

- Reduce CPU cycles
- Reduce the amount of storage used by the data collection data spaces

The reduction in data space storage also reduces the amount of real storage used by the SYSVAAST data anchor address space.

This section contains the following topics:

How to Control Data Collection Using the Event Scheduler

The CA SYSVIEW data collector components collect, monitor and provide exception processing for many of resources and metrics.

The Event Scheduler controls data collection as follows:

1. The scheduler provides the ability to schedule events on an interval basis.
2. The scheduled event definitions can be customized to control the collection interval, time of day, and day of week.
3. The data collection process is defined as two separate but integrated components:

   Data Collection

   The process of gathering data metrics over a defined time. The data that is collected is stored for future analysis and reporting. CA SYSVIEW primarily uses the following two types of data collection:

   - Interval-based data gathering
   - Event-based data gathering

   Exception Processing

   Exception processing analyzes collected data and provides alerts and notifications based on user-defined criteria. This function is also commonly referred to as threshold processing.

PARMLIB Members Containing Data Collection Definitions

The data collection component is design to work out-of-the-box. The administration of the data collection processes is controlled through the CA SYSVIEW Event Scheduler.

The following parmlib members contain initial data collection event definitions:

SCHEDULE

Defines scheduled events.

SCHDMVS

z/OS data collection events

SCHDMQS

WebSphere MQ data collection events

SCHDIMS

IMS data collection events

SCHDTCP

TCP/IP data collection events

CICSSCHD

Defines CICS scheduled events.

SCHDCICS

CICS data collection events

The previous parmlib members are only used to create the initial list of events or during a COLD start of the Event Scheduler.

If the Event Scheduler is being WARM started, the recommended start method, then the list of events is maintained in the persistent data store.

Control Data Collection Events

You can make ongoing changes to the data collection events using the online displays.

Follow these steps:

- Use the CA SYSVIEW online command SCHEDULE to control the following data collection events:
  - z/OS
  - WebSphere MQ
  - IMS
  - TCP/IP
- Use the CA SYSVIEW online command CSCHEDUL to control the following data collection events:
  - CICS

Your changes are maintained and stored in the CA SYSVIEW persistent data store.

For COLD starts, manually change the parmlib members.

VARIABLE-SET—Enable and Disable Data Collection Metrics

The VARIABLE-SET statements can be directly coded in the *xxx*DATA parmlib members if desired.

The *xxx*VARS members are dynamically included into the corresponding *xxx*DATA parmlib member through an include statement such as:

```
)INCLUDE MVSVARS
```

To limit the data collection, enable or disable the collection metrics using the following CA SYSVIEW parmlib members:

MVSVARS

Defines the z/OS metrics.

MQSVARS

Defines the WebSphere MQ metrics.

IMSVARS

Defines the IMS metrics.

TCPVARS

Defines the TCP/IP metrics.

CICSSTAT

Defines the CICS metrics.

Note: The data collection tasks do not directly read the parmlib members MVSVARS, MQSVARS, IMSVARS and TCPVARS.

The VARIABLE-SET statement has the following syntax:

```
VARIABLE-SET  source:variable:action
```

source

Specifies the resources to enable or disable for collection.

Valid values are:

- MVS
- WEBMQ
- IMS
- TCPIP

variable

Specifies the name of the data collection metric, which you can specify generically.

- Variable-length mask character: = (equal)

- Fixed-length mask character: * (asterisk)

action

Specifies the action to apply. Valid values are:

- ENAbled
- DISabled

Examples: Enable and Disable Data Collection

- Enable collection for all MVS metrics:

  ```
  VARIABLE-SET      MVS:=:ENABLE
  ```

- Disable USS related collection for JOBS/ASIDs:

  ```
  VARIABLE-SET      MVS:JOBU=:DISABLE
  ```

- Disable zAAP related collection for JOBS/ASIDs:

  ```
  VARIABLE-SET      MVS:JOBIFA=:DISABLE
  ```

  Note: Use this setting if no zAAP processors are available.

- Disable zIIP related collection for JOBS/ASIDs:

  ```
  VARIABLE-SET      MVS:JOBIIP=:DISABLE
  ```

  Note: Use this setting if no zIIP processors are available.

- Disable TCP/IP IP version 4 variables:

  ```
  VARIABLE-SET      TCPIP:I4=:DISABLE
  ```

  Note: Use this setting if IP version 4 is not used.

- Disable TCP/IP IP version 6 variables:

  ```
  VARIABLE-SET      TCPIP:I6=:DISABLE
  ```

  Note: Use this setting if IP version 6 is not used.

Chapter 7: zIIP Processor

IBM offers an optional specialty processor type know as a z/architecture Integrated Information Processor (zIIP processor). The zIIP processor lets you offload specific types of work from a general processor or CP.

The type of work that is capable or allowed to execute on a zIIP processor is limited to SRB-mode work executing in an Enclave.

SRB-mode has requirements that can limit the type and amount of code that is eligible to run in SRB-mode and therefore on a zIIP processor. Because of the limitations, not all code can run on a zIIP processor.

Example requirements or limitations:

- Requires an authorized environment
- Cannot issue any SVCs
- Cannot perform I/O

Note: Branch entry WTOs may impact your automation and you should consult with you automation team.

This section contains the following topics:

Determine Work Volume

CA SYSVIEW provides several methods to determine how well the zIIP processors are being exploited or utilized.

Note: Only SRB-mode work executing in an Enclave is eligible to run on a zIIP processor. The important or keyword here is eligible. Although SRB-mode work could be executing on an Enclave, it still not always executes on a zIIP processor.

WLM determines the work to dispatch on a zIIP processor. If a zIIP processor is not available, the work is still dispatched on a general process (CP). This process is commonly known as zIIP on CP work.

Follow these steps:

- To display the current CPU configuration and usage, issue the following command:

  ```
  CPU
  ```

- To plot the overall use of the zIIP processors, issue the following command:

  ```
  PLOT MVS CPU% IIP
  ```

- To display the WLM enclaves utilizing zIIP processors, issue the following command:

  ```
  WMENCLAV
  ```

- To display address spaces that are using a zIIP processor, issue the following command:

  ```
  ACTSUM
  ```

Enable Exploitation of zIIP Processing

This topic provides the requirements that must be met for CA SYSVIEW to exploit zIIP processing.

Follow these steps:

1. Define at least one zIIP processor to the LPAR.
2. Install the CA Common Services component to enable the zIIP environment.
3. Set the CA SYSVIEW configuration options to request zIIP processing to be enabled. These options are enabled by default:

```
SYSVIEW Service Address Spaces
            Parmlib member: OPTIONS
            Configuration Option         Value
            -------------------------    -------
            zIIPEnable                   Yes

SYSVIEW for CICS Data Collection
            Parmlib member: CICSOPTS
            Configuration Option         Value
            ------------------------     -------
            ZIIP-ENABLE                  Yes
```

The requirements are met and CA SYSVIEW can exploit zIIP processing.

Note: For more CA Common Services component requirements information to enable the zIIP environment, see the *Installation Guide*.

Determine Exploitation of zIIP Processors

CA SYSVIEW provides in-depth views into its usage and exploitation of zIIP processors.

Views are provided at the following three levels:

- Address space
- Task
- Data collection component

Follow these steps:

- Display views for the CA SYSVIEW Service Address Spaces
  - To display the CA SYSVIEW address space administration summary and CA SYSVIEW zIIP usage by task, issue the following command:

    ```
    ASADMIN
    ```

  - To display the CA SYSVIEW zIIP usage by the data collection component, issue the following command:

    ```
    SYSDMON
    ```

  - To display the CA SYSVIEW zIIP usage dashboard, issue the following command:

    ```
    SYSVZIIP
    ```

    This view combines all of the information into an integrated graphical dashboard view.
- Display views for the CA SYSVIEW for CICS Data Collection
  - To display the CA SYSVIEW for CICS address space summary, issue the following command:

    ```
    CICSLIST
    ```

  - To display the CA SYSVIEW for CICS zIIP usage by task and zIIP usage by the data collection component, issue the following command:

    ```
    CSYSDMON
    ```

  - To display the CA SYSVIEW for CICS zIIP usage dashboard. This view combines all of the information into an integrated graphical dashboard view, issue the following command:

    ```
    CICSDATA
    ```

  - To display the CA SYSVIEW for CICS zIIP usage task summary. This view shows a single data row for each CICS, issue the following command:

    ```
    CDATAMON
    ```

Chapter 8: Utilities

This chapter describes the following utilities:

- The MIB compiler
- The system information utility

This section contains the following topics:

Compile a MIB to Use with MIB Browsing Commands

CA SYSVIEW provides a set of MIB browsing commands that let you communicate with any SNMP-compatible device or product.

Follow these steps:

1. Use the provided sample job to compile a user-supplied MIB. The sample JCL is in the data set *sysview*.SAMPJCL(MIBCOMP).
2. Save its output in the *sysview*.CNM4BMIB data set.

The compiled MIB definitions are used as input by the set of MIB browsing commands.

Run the System Information Utility

The system information utility provides useful information about how your system is defined. The functions available include the following:

- z/OS system information
- Subsystem information
- APF list information

A sample job has been provided to run the system information utility. The sample JCL is in the data set *sysview*.CNM4BSAM(GSVCUTIL).

Chapter 9: Using the Application Programming Interface

The application programming interface (API) for CA SYSVIEW can be used to obtain information from CA SYSVIEW displays for use in other programs. You can use TSO/E REXX to access the API.

Commands are passed to the API by using the REXX ADDRESS function. The information from the display is passed back to the caller in a line-oriented format. Change the information passed back by using the API.

A CA SYSVIEW environment is built as a subtask within the address space of the requester. The subtask and environment are freed when the session terminates.

This section contains the following topics:

Address Command—Issue Commands to the API

Use the Address command to issue a command to the API.

This command has the following format:

```
ADDRESS 'SYSVIEWE' 'keyword(value)' 'keyword(value)'...
```

Keywords can be abbreviated by using only their first character. For example, the following syntax causes the MTT command to execute:

```
ADDRESS 'SYSVIEWE' 'C(MTT)'
```

Keywords used with ADDRESS SYSVIEWE:

BLANKLINES(YES|NO)

Controls whether blank lines are returned on the stack.

Default: NO

COMMAND(*cmd*)

cmd is a CA SYSVIEW command, which has a maximum length of 256.

DELIMITER(*dlm*)

dlm is the delimiter that is used to separate fields on CA SYSVIEW displays that have defined formats.

Possible *dlm* values:

| @ # $ % ¢ & * ! < > : . / + = _ - ' “ ? ; } { \ ~

Once a delimiter is set, it remains in effect until another DELIMITER keyword is entered or the API interface is terminated.

Default: The vertical line (|)

EXTRACT(YES|NO)

Controls whether the title, message, and info line data are automatically extracted when the SET APISTACK or API STACK option is set to NO.

Default: YES

INPUT(*fld,val*)

fld

The name or relative number of an input field to receive input data.

val

The new value for the field.

If the command has a defined format, the name of a field can be used for the *fld* parameter. Otherwise, a number must be used to describe the relative input field position. You assign a 1 to the first input field and then add 1 for each input field after the first input field. The INPUT keyword only affects the first data line returned. The DOWN and FIND commands can be used to position a line so the INPUT keyword can be used.

The INPUT keyword is processed against the output of the last command executed by the COMMAND(*cmd*) keyword of the previous ADDRESS 'SYSVIEWE' statement. The INPUT keyword is not processes against the output of any COMMAND(*cmd*) keyword on the same ADDRESS 'SYSVIEWE' statement.

When you include both the INPUT and COMMAND keywords on the same ADDRESS 'SYSVIEWE' statement results in both keywords being accepted. No error condition generates, which probably does not produce the desired result. The INPUT and COMMAND keywords process in the following sequence:

1. The INPUT keyword, in effect, types in a data row on a virtual screen
2. The COMMAND keyword then types a command in the primary command input field

When all parameters on the ADDRESS 'SYSVIEWE' statement have been processed, the result is submitted to CA SYSVIEW. CA SYSVIEW execution of the command in the primary command input field takes precedence over the data row input.

LIMIT(*nnnnnn*)

nnnnnn is the maximum limit of the number of lines returned. *nnnnnn* must be a number from 4 through 999999. When the amount of data returned exceeds the LIMIT value, the GSVX360W message is returned stating the limit was exceeded and the data is incomplete. Once a limit is set, it remains in effect until another LIMIT keyword is entered or the API interface is terminated.

Default: 100000

PLACEHOLDER(*char*)

Specifies the character used by the API interface as a placeholder for blank fields and to fill embedded blanks in formatted fields. Specify any character in the range X'40' to X'FE'. You can use hex string notation to specify a character that cannot be directly typed or displayed.

Default: A blank

The API interface uses the placeholder character for the following line types returned on the stack:

- **T** - All title lines
- **H** - Header lines when field formats are used
- **D** - Data lines when field formats are used

The current placeholder character is always the last character in the title line.

SIZE(rows,cols)

Controls the virtual screen size used by the API. The rows value must be a number from 16 to 255 and the cols value must be a number from 79 to 255. The SIZE parameter is only valid on the first ADDRESS SYSVIEWE call.

Default: SIZE(100,132).

STACK(YES|NO)

Controls whether the REXX API interface places information in the stack.

Default: YES

A value of NO is in effect only for the duration of the product session. When APISTACK is set to NO, the REXX API interface:

- Does not place any information in the stack
- Does perform the following functions on every user interaction:
  - XVEXTRAC TITLE
  - XVEXTRAC MESSAGE
  - XVEXTRAC INFO

The REXX EXEC can then use the XVEXTRAC DATA function to extract whatever data fields are necessary for its processing.

For more information, see the XVEXTRAC command in the CA SYSVIEW online help system.

ZEROIFBLANK(YES|NO)

Controls whether any numeric field that is all blanks is:

- Left alone (NO)
- Filled with a single right-justified character; that is, a zero (YES).

Note: This keyword can only change blanks to a zero.

Default: NO, unless your site changed this setting with the OPTIONS PARMLIB member ZeroIfBlankForNonInterActAppl.

Setting this keyword to YES has no effect in the following instances:

- When the field already displays a zero
- When the screen is unformatted

Data Returned from the API

After you send a command to the API, the data from the display is returned up to the limit specified by the LIMIT keyword. The data is returned in the external data queue or stack.

The following types of lines are returned when a command is executed:

Message

The first position of a message line contains the letter M. This line is always first and is always present even if there is no message. The message text starts in the second position of the line. Only one message line is returned for each call to the API.

Title

The first position of a title line contains the letter T. The following fields are returned in this order:

- Product name
- Product release
- System ID where the information is obtained
- Command that produces the information
- Current date
- Current time

The second position of the title line contains a default delimiter that marks the start of each field. Specify the default delimiter with the DELIMITER keyword. Only one title line is returned for each call to the API.

Info

The first position of an information line contains the letter I. They contain the information line fields from the command display. Multiple information lines can be returned depending on the number of information lines on the command display.

Header

The first position of a header line contains the letter H. The header line contains the header line fields from the command display. If the second position of the line contains the default delimiter, the delimiter marks the start of each field. If the second position is blank, the fields are not delimited.

Data

The first position of a data line contains the letter D. They contain the data line fields from the command display. If the second position of the line contains the default delimiter, the delimiter marks the start of each field. If the second position is blank, the fields are not delimited. Multiple data lines can be returned.

Sample Display of Returned Data

The following example shows the data returned after issuing the command ADDRESS 'SYSVIEWE' 'C(ACTIVITY)':

```
M
T|SYSVIEW |     |C2        |ACTIVITY|03/05/08|06:40:06
I CPU  38%  LCPU  36%  Paging  5  SIO  138
H|Cmd|Jobname |Stepname|Procstep|Type|Jobnr |Cl|STATUS |CPU-Time |Limit
D|   |GLNREP  |GLNREP  |GLNREP  |STC |12988 |$ |OUT LW |    0.49 |86400
D|   |*MASTER*|        |        |SYS |12770 |$ |NS     |   41.89 |86400
D|   |PCAUTH  |PCAUTH  |        |SYS |      |  |NS     |    0.03 |
D|   |RASP    |RASP    |        |SYS |      |  |NS     |    0.01 |
D|   |TRACE   |TRACE   |        |SYS |      |  |NS     |    0.01 |
D|   |XCFAS   |XCFAS   |IEFPROC |SYS |      |  |NS     |  177.98 |86400
D|   |GRS     |GRS     |        |SYS |      |  |NS     |    7.07 |
D|   |SMXC    |SMXC    |        |SYS |      |  |NS     |    0.00 |
D|   |SYSBMAS |SYSBMAS |        |SYS |      |  |NS     |    0.00 |
D|   |DUMPSRV |DUMPSRV |DUMPSRV |SYS |      |  |NS     |    7.06 |86400
D|   |CONSOLE |CONSOLE |        |SYS |      |  |NS     |   55.94 |
D|   |ALLOCAS |ALLOCAS |        |SYS |      |  |NS     |    0.01 |
D|   |SMF     |SMF     |IEFPROC |SYS |      |  |NS     |    1.02 |86400
D|   |LLA     |LLA     |LLA     |STC |      |  |NS     |    4.17 |86400
D|   |VLF     |VLF     |VLF     |STC |      |  |NS     |    5.96 |86400
D|   |INIT    |INIT    |IEFPROC |INIT|12934 |$ |OUT DW |    0.03 |
```

The data would extend to the right for the full length of the display. The number of lines returned can be determined by using the REXX QUEUED() function.

Assume that the last call returned the previous data. The following call would simulate entering the **S** line command in the Cmd field next to the job GLNREP:

```
ADDRESS 'SYSVIEWE' 'INPUT(CMD,S)'
```

The output for job GLNREP would then be returned.

Return Codes from Each API Call

The following return codes are passed back after each API call:

0

Command executed, no messages are returned.

4

Informational message returned. You always receive return code 4 the first-time you call the API. The copyright message is always displayed when you first invoke CA SYSVIEW.

8

Action messages are returned.

12

Warning messages are returned.

16

Error messages are returned.

20

Termination return code. The API has terminated.

C(END)—End the API Session

To terminate the API session, use the following command:

```
ADDRESS 'SYSVIEWE' 'C(END)'
```

The C(END) is a requirement when using TSO/E REXX; if you do not use this command, an abend occurs.

API Called from a REXX Internal Subroutine

When a REXX EXEC calls an internal subroutine to issue ADDRESS SYSVIEWE statements, and the subroutine includes the PROCEDURE statement to hide subroutine variables from the calling routine:

- The subroutine must use the PROCEDURE EXPOSE option
- Include a variable named GSVXAPIE_APIC in the list of exposed variables

Use the API with TSO/E REXX

To use the CA SYSVIEW API from TSO/E REXX, update the REXX host command environment table to include the SYSVIEW entry. One method is to code the following statement at the beginning of the REXX routine that invokes the API:

```
ADDRESS 'LINK' 'GSVXRXAA'
```

The GSVXRXAA module adds the SYSVIEW entry to the table when it does not exist.

To avoid the overhead of invoking the GSVXRXAA module multiple times, add the SYSVIEW entry to the IRXPARMS, IRXTSPRM, and IRXISPRM modules. Verify that the routine name is GSVXAPIE and the token value is blank.

For more information about updating these modules, see your IBM documentation.

Sample REXX Procedure

A sample REXX procedure that invokes the API follows. This sample is provided in the APISAMP member of the *sysview*.CNM4BSAM data set. This procedure displays all jobs using a resource like a data set name.

Important! This procedure uses the REXX POS function to get the offsets of the fields in the data. This procedure avoids problems with fields that change position on the display. Do not assume that the fields are always in a particular order or that they are always at the same offset.

```
/* REXX ============================================================*/
/*                                                                  */
/* This sample REXX procedure will display jobs which have issued   */
/* an enqueue for a specific resource like a data set name.         */
/*                                                                  */
/* Input parameter: res - The resource name to test                 */
/*                                                                  */
/*==================================================================*/
/*------------------------------------------------------------------*/
/* Change to TRACE I for testing                                    */
/*------------------------------------------------------------------*/
TRACE N
/*------------------------------------------------------------------*/
/* Check for required parameter                                     */
/*------------------------------------------------------------------*/
```

```
PARSE UPPER ARG res
IF res = '' THEN
  DO
    SAY 'Required resource parameter missing'
    EXIT(-1)
  END
/*--------------------------------------------------------------------*/
/* Initialization                                                     */
/*------------------------------------------------------------------- */
accum = 0
msg. = ''
ADDRESS 'LINK' 'GSVXRXAA'             /* Init SYSVIEW address env    */
/*--------------------------------------------------------------------*/
/* Get enqueues for specified resource                                */
/*--------------------------------------------------------------------*/
ADDRESS 'SYSVIEWE' "C(ENQUEUE "res")"
IF QUEUED() = 0 THEN                  /* Error occurred if no data    */
  DO
    SAY 'Error - No data returned from ADDRESS SYSVIEWE'
    SIGNAL EXIT
  END
/*--------------------------------------------------------------------*/
/* Extract 1st line. Should be a 'M' (Msg line).  Msg follows...      */
/*--------------------------------------------------------------------*/
PARSE PULL msgline
/*--------------------------------------------------------------------*/
/* Find header line                                                   */
/*--------------------------------------------------------------------*/
ltype = ''
DO WHILE QUEUED() > 0
  PARSE UPPER PULL ltype 2 ldelim 3 ldata
  IF ltype = 'H' THEN LEAVE
END
IF ltype ¬= 'H' THEN                  /*Error occurred if no header   */
  DO
    SAY 'Error - Header line not returned from ADDRESS SYSVIEWE'
    SIGNAL EXIT
  END
/*--------------------------------------------------------------------*/
/* Get the offsets of required fields                                 */
/*--------------------------------------------------------------------*/
rnameo    = POS('RNAME',ldata)          /*Get offset of Rname field   */
jobnameo  = POS('JOBNAME',ldata)        /*Get offset of Jobname field */
typeo     = POS('TYPE',ldata)           /*Get offset of Type field    */
/*--------------------------------------------------------------------*/
/* Process rest of returned data looking for data lines               */
/*--------------------------------------------------------------------*/
DO WHILE QUEUED() > 0
  PARSE PULL ltype 2 ldelim 3 ldata
```

```
  IF ltype 'D' THEN                      /* Process if data line      */
    DO
      PARSE VAR ldata =(rnameo) rname =(jobnameo) jobname =(typeo) type
      PARSE VAR rname rname (ldelim)    /* Parse rname to delimiter   */
      IF rname = res THEN               /* Process if rname matches   */
        DO
          PARSE VAR jobname jobname (ldelim) /*Parse jobname field    */
          PARSE VAR type    type    (ldelim) /*Parse type field       */
          accum = accum + 1                  /* Increment accumulator */
          msg.accum = jobname'    'type      /* Save jobname and type */
        END
    END
END
/*--------------------------------------------------------------------*/
/* If any jobs found, display them                                    */
/*--------------------------------------------------------------------*/
IF accum > 0 THEN
  DO
    SAY '*** Number of jobs using resource 'res' is 'accum' ***'
    SAY 'Jobname     Type'               /* Display header            */
    SAY '-------     ----'
    DO i = 1 TO accum                   /* Display each job          */
       SAY msg.i
    END
    SAY '-------     ----'              /* Display end line          */
  END
ELSE SAY '*** Resource 'res' not in use ***'
/*--------------------------------------------------------------------*/
/* Exit procedure                                                     */
/*--------------------------------------------------------------------*/
EXIT:
/*--------------------------------------------------------------------*/
/* Formal termination of the API is required.                         */
/* Abends will occur if this step is omitted.                         */
/*--------------------------------------------------------------------*/
ADDRESS 'SYSVIEWE' "C(END)"            /* TERMINATE SYSVIEW SESSION   */
/*--------------------------------------------------------------------*/
/* Discard any residual output, so TSO won't try to execute it.
/*--------------------------------------------------------------------*/
CALL CLEAREDQ                          /* Discard any residual output */
EXIT 0
/* ================================================================*/
/* CLEAREDQ Subroutine:  Clean out the external data queue            */
/* ================================================================*/
CLEAREDQ: PROCEDURE
TRACE N
   DO WHILE QUEUED () > 0
   PULL line
   END
```

```
RETURN
```

Chapter 10: Use the Local 3270 Device Interface

Use the local 3270 device interface to run the product in a dedicated mode from any locally attached 3270 device.

This section contains the following topics:

Start a Session when TSO and VTAM are Inactive

Use the local 3270 device interface of CA SYSVIEW to run the product in a dedicated mode from any locally attached 3270 device.

Use this interface to start a session with CA SYSVIEW even when TSO and VTAM are not active. Also use this interface when JES2 is not active.

Follow these steps:

1. Copy the SYSVLCL procedure from the *sysview*.SAMPJCL data set into SYS1.PROCLIB.
2. Specify SUB=MSTR on the START command as described in the next section.

The TSO session starts.

START SYSVLCL—Invoke SYSVLCL PROC

Invoke the SYSVLCL PROC by entering the following z/OS command from a system console:

```
START SYSVLCL[.identifier],devn[,SUB=subsystem]
```

identifier

Specifies the name identifying the task to start. This name can be as many as eight characters long. The first character must be alphabetical. If no identifier is specified, the device number specified for *devn* becomes the identifier.

devn

Specifies the device number of the local 3270 display device with which the CA SYSVIEW session is to start.

subsystem

Specifies the name of the subsystem that selects the task for processing. The name must be one to four characters long, defined in the IEFSSNxx member of SYS1.PARMLIB, and the subsystem must be active. If SUB= is not specified, the primary subsystem as specified in IEFSSNxx selects the task.

Chapter 11: Event Capture Option

The CA SYSVIEW Event Capture option enables you to capture the critical data to analyze historical events to assist you maintain your system.

With the Event Capture option, you can:

- Set collection methods by schedules, intervals, threshold, or exception events
- Use SMF collection data
- Gather data on multiple systems simultaneously
- Redisplay any data that can be viewed using a CA SYSVIEW Display command
- Set retention periods for storing data

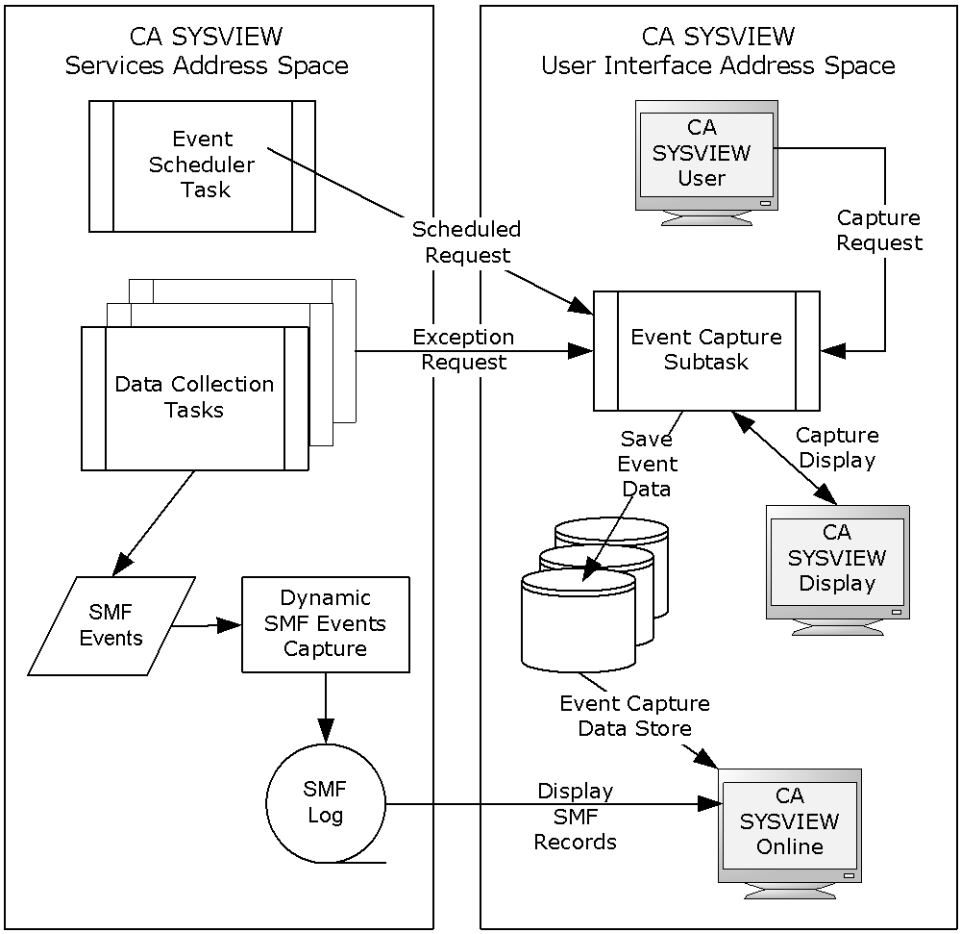
The Event Capture option uses two methods to capture and redisplay historical data:

- The Event Capture function
- The SMF Capture function

This section contains the following topics:

How Event Capture Option Works with Your System

The following illustration shows how the Event Capture option fits into your system:



Event Capture Function Collection Methods

The Event Capture function takes snapshots of a specific system at a given time. Use this snapshot to diagnosing current or potential problems on that system.

The Event Capture function collects, groups, and saves the captured CA SYSVIEW command displays by event name in chronological order. This information is then indexed for easy retrieval. A single Capture index can include data for multiple systems or for only one system. Multiple Event Capture indexes can be in use across all of your systems. However, a single system can only use one Event Capture index at any given time.

This information is not restricted to those systems participating in a sysplex. Regardless of the index method, you can display the list of indexed events from a single index or from a group of indexes. Remember that any index can represent more than one physical system.

You can collect data with Event Capture using the following methods:

- On Demand by User Request

  You can request the capture of an event at any time. To do so, first determine if any existing CAPLIB event member contains the desired list of commands you want to capture. If so, capture the event by issuing the following command:

  `CAPTURE eventname`

  You can also initiate a capture event from the system console by issuing the following command:

  `MODIFY sysview,CAPTURE eventname`

- Scheduled Time Events

  CA SYSVIEW contains an event scheduler that can execute CAPTURE events at predefined times or intervals. With the scheduler, you can have one or more schedule events, each of which can perform one or more CAPTURE events.

- Threshold- or Exception-Driven Events

  CA SYSVIEW contains hundreds of monitored metrics that can have associated threshold or state definitions. These definitions provide a mechanism to alert you of current or potential problems. You can assign problem and warning levels to each definition, and then take proactive measures upon receiving early alerts of a problem. When a threshold definition is triggered, you can specify one or more actions to take, including capturing an event. You can set thresholds in the threshold definition members in the parmlib, or with the online definition commands. Capturing the status or look of the system at the time of an alert helps you to debug or solve a problem at another time.

- Automated Operations Events

  Similar to threshold-driven events, automated operation events track and monitor the state of hundreds of resources. The automated operations rules or definitions can be used to trigger event captures.

Data Source for Event Capture

You can capture any CA SYSVIEW command display for redisplay at another time. You can redisplay the data either on the same system on which it was captured or on a different system. The redisplayed data is presented using the profile settings of the current user and can be manipulated with the following commands:

- SORT
- SELECT
- PRINT

Capture a Group of Commands

Though a captured event can be a single command display, it is more useful to capture a group or list of commands as one event. The list of commands to execute are defined and stored in a member of the CAPLIB data set. This member is a simple text member containing a list of commands.

Follow these steps:

The following steps let Event Capture monitor common storage information.

1. Create a member named CSA in the CAPLIB data set.

   The CSA member is referred to as the CSA event.

2. Enter a list of CA SYSVIEW command displays that you want to have captured in the CSA CAPLIB member.

   For example, the CSA member can contain the following commands:

   - COMMON SUMMARY

     Displays overall summary of common storage usage for the system.

   - VSMTRACK SUMMARY

     Displays orphaned common storage summary by address space.

   - ACTIVITY

     Displays list of active address spaces.

   - SORT CSA D LIMIT 25

     Sort address space list by CSA usage in a descending order. Limit the display to the top 25 biggest users of CSA.

3. Trigger the CSA event using any of the collection methods described in the previous section.

The group of commands are captured.

Event Capture Data Organization

The following sections discuss Event Capture organization, including data structures and data retention.

Data Structures

The Event Capture data sets can be thought of as data stores. These data sets can be shared among multiple systems, or unique data sets can be created for each system.

The Event Capture function uses the following three different types of data sets:

- The Capture Library data set

  The Capture library data set (CAPLIB) contains members whose content describes the data the specific event is to capture. Each event member is a simple list of SYSVIEW display commands and each member in the data set is a separate event. Individual event members can be combined to make larger sets of events using the following parmlib keyword statements:

  "`)INCLUDE member`"

  Sample event members are supplied in the *sysview*.CNM4BCAP data set that comes with CA SYSVIEW.

- The Capture Index data set

  One or more index data sets are used to catalog a list of captured events. A common or shared index data set can be used for all systems when the data set is allocated on shared DASD. A separate index data set can also be used for each system. Index data sets are defined as a VSAM KSDS data set. If separate index data sets are used, the high-level qualifier specified for the index data sets should contain the system name. You can easily specify the high-level qualifier using the &SYSNAME symbolic parameter.

- The Capture Event data set

  The Event data sets are used to store the captured data for each event. One event is stored per data set. Specify a high-level qualifier for the Event data sets. Verify that the high-level qualifier contains a maximum of 29 characters to provide enough room to append the date and time qualifiers.

  Event data set names have the format:

  `sysview.capture.hlq.Dyymmdd.Thhmmss`

  sysview.capture.*hlq*

  Is specified in the System Configuration Options member using the Dsn-System-CAPDATA-HLQ keyword.

  D

  Indicates Date, and *yymmdd* specifies the year, month, and day.

  T

  Indicates Time, and *hhmmss* specifies hour, minute, and second.

  Important: Understand that a large amount of Event data sets can be created. Set the default space allocation parameters for the Event data sets in the parmlib member CAPTURE, which is found in the *sysview*.CNM4BPRM data set.

Data Retention Methods

Because a large amount of Event data can be captured, it is important to understand how to maintain this data either automatically or manually. Because of the potentially short-term lifespan of Capture Event data sets, it is a good practice to allocate the Event data sets on WORK volumes.

We recommend the following methods for specifying a retention period for captured events to control how long those events are saved:

- The Capture function

  Lets you specify and associate a retention period with each captured event in the CAPTURE parmlib member, or in each CAPLIB member.

- SMS storage class definitions

  Contains data set retention information. If you use this feature for setting retention periods, the process continues to function properly.

- CA SYSVIEW built-in maintenance and cleanup function

  Deletes and uncatalogs expired data sets on an interval basis. Use this facility to control the amount of data retained. If a data set is manually deleted, the supplied Index Maintenance function CAPMAINT cleans up any outstanding data.

If you do not use a retention period to remove Event data sets automatically, then those data sets exist forever unless manually deleted.

More information:

Event Capture Data Maintenance (see page 131)

Display Event Capture Data Using the CAPLIST Command

The system programmer received notification that the system experienced a common storage problem. The Event Capture facility saved the information to analyze the problem. You can display the data using the CAPLIST command.

Follow these steps:

1. Log on to a CA SYSVIEW session running on any available computer and enter the CAPLIST command.

   Captured data can be accessed from any session, even if the system to analyze is not running.

   The CAPLIST command displays a list of all available captured events. The displayed list is sorted by date to make finding the timeframe in question easier to locate. The following screen indicates a storage problem at 12:30 on system XE44:

```
SYSVIEW CAPLIST -------- Captured Event List --------------- ddmmmyyyy 13:59:33
Command ===>                                                  Scroll *===> HALF
------------------------------------------------ Lvl 4 Row 1-16/812 Col 1-79/383
Index Hlqual SYSVIEW.BASE.CAPINDEX                                      Count    1
Total Recs   812 Blks 43047 Expd   334  Select Recs   812 Blks 43047 Expd   334
--------------------------------------------------------------------------------
Cmd      Date     Time  SysName  Member   Descr                          Status
________ 01/10/08 12:30 XE44     COMMON   Common storage usage
________ 01/10/08 12:30 XE44     STORAGE  Storage usage                  PROBLEM
________ 01/10/08 12:30 XE44     OVERVIEW Overview status                PROBLEM
________ 01/10/08 12:30 XE44     JESSPOOL JES2 spool usage
________ 01/10/08 12:30 XE44     CPU      CPU usage
```

2. Type **S** in the Cmd field for the Storage usage entry. This entry has the keyword Problem in the Status column and you want to view details about possible cause of the problem.

   The following display appears showing a list of individual command displays that were captured for the selected event. Each command display is marked CMD in the Type field. The Norm, Warn, and Prob columns in the display show the respective number of incidents for the listed commands.

```
SYSVIEW CAPEVENT --------- Captured Event ------------------ ddmmmyyyy 14:01:19
Command ===>                                                  Scroll *===> HALF
------------------------------------------------- Lvl 5 Row 1-9/9 Col 1-79/146
Dsname   SYSVIEW.BASE.CAPDATA.D011008.T123051             Volser WORK03
Descr    Storage usage                                    Member STORAGE
System   JobName  Asid UserId   TaskId   Created                 Expires
XE44     SYSVDEV  015E GSVXSCHT SCHEDULR 12:30:51 08Oct2001      11Oct2001
-------------------------------------------------------------------------------
Cmd      Type Name     Screen   Title                          Norm Warn Prob
________ ID                     Capture dataset identification
________ MBR                    CAPLIB member
________ CMD  ACTIVITY          System Activity
________ CMD  COMMON   SUMMARY  Common Storage Summary
________ CMD  PAGEDS            Page Datasets
________ CMD  PAGING            MVS System Paging                1
________ CMD  STORAGE           MVS System Storage               4    2    4
________ CMD  SWAPPING          MVS System Swapping
________ CMD  VSMTRACK SUMMARY  Common Storage Tracking Summary
```

3. Drill down another detail level by typing **S** in the Cmd column to select the STORAGE command, which has two warnings and four problems.

   Note: Any of the commands listed on the screen can be selected for redisplay.

   The redisplayed information is a virtual look into the past. You can manipulate the data on the screen using the standard set of CA SYSVIEW function commands such as: SORT, SELECT, and FIND.

Event Capture Data Maintenance

Perform periodic maintenance on the Event Capture data structures to verify that the Capture Index and Event data sets are synchronized. Event Capture data can be stored and made available for as long as you want, but could eventually consume considerable amounts of DASD space. For this reason, we recommend using automated maintenance, as described in the following sections.

More information:

Data Retention Methods (see page 129)

Index Maintenance Utility Functions

Over time, Captured Event data sets are deleted automatically, based on their retention period and expiration date, or they are deleted manually. When an Event data set is deleted, remove the corresponding entry from the index data set. The CA SYSVIEW Index Maintenance utility (CAPMAINT) can perform the required functions to solve these synchronization problems.

You can schedule CAPMAINT to run on an interval basis, using the SYSVIEW Event Scheduler. For details, see the chapter “Start the Product Subtasks.” By default, the CAPMAINT utility is triggered in the supplied scheduled events parmlib member SCHEDULE.

We recommend running the CAPMAINT utility periodically to synchronize your Capture Index and Event data structures. The same functions can also be performed manually as subcommands and line commands of the CAPMAINT command display, as described in the following list:

- Cleanup Function

  This function attempts to synchronize the specified Capture Index data set and its corresponding Capture Event data sets. The cleanup function is both a subcommand and line command of the CAPMAINT command:

  - Subcommand

    If the cleanup function is entered as a subcommand, the Event data sets listed in the index are processed one at a time.

  - Line command

    The cleanup function only attempts cleanup of the selected event entry. The following cases illustrate the results of using the cleanup function based on the state of the Event data set.

    Case 1: The event data set does not exist.

    Action: The index entry for the event data set is erased.

    Case 2: The event data set does exist and is expired.

    Action: The event data can be optionally deleted based on the specified options of DELETE or NODELETE. If the data set is successfully deleted, the index record is erased.

    Case 3: The event data set does exist and is not expired.

    Action: None

- Delete Function

  The delete function is a line command of the CAPMAINT command. The delete line command is entered in the line command field of a selected event. The function deletes and uncatalogs the event data set. The corresponding index entry is not changed.

- Erase Function

  The erase function is a line command of the CAPMAINT command. The erase line command is entered in the line command field of a selected event. The function erases the selected entry in the index data set. The corresponding event data set is unchanged, when one exists.

- Purge Function

  The purge function is a line command of the CAPMAINT command. The purge line command is entered in the line command field of a selected event. The function deletes and uncatalogs the event data set. The corresponding index entry is erased. This function is equivalent to issuing DELETE and ERASE. This function is similar to the cleanup function.

  Important! Purge deletes the Event data set, regardless of the retention period set, that is, the expiration date.

- Import Function

  The import function is a subcommand of the CAPMAINT command. The import function lets you associate manually an event data set with an index.

SMF Event Capture Function

The Systems Management Facility (SMF) provides an additional source of historical data. Any data that is viewable through a SYSVIEW command display can be captured for later redisplay. The data that has been captured is maintained in an event data store or event log for SMF records.

More information:

Starting the Product Subtasks (see page 29)

Chapter 12: Using the Batch Interface

This chapter explains how to run the product as a batch job.

This section contains the following topics:

How to Use the Batch Interface

The following example illustrates the use of the batch interface.

In this example, the simulated screen size is 80 columns by 66 rows. The screen line size is 80 characters because RECFM contains the ASA specification.

```
//EXECBAT   JOB ,pgmrname,USER=userid
 //STEP0001  EXEC PGM=GSVXBAT,PARM='LINECNT=66,SHOWINP=NO'
 //STEPLIB    DD  DISP=SHR,DSN=SYSVIEW.CNM4BLOD
 //SYSPRINT DD  SYSOUT=*
 //SYSIN       DD  *
 *
 *   Sample input
 *   User ID for batch session is taken from JOB card
 *
 COMMAND=(VTOC MVSRES)      1. Issue VTOC command
 DATA=(1,DSI),              2. Select volume information line and...       X
 DATA=(2,DSI),DATA=(3,DSI)  3. Select first two data sets
 COMMAND=RETURN             4. Return for volume DSINFO screen
 COMMAND=RETURN             5. Return for first data set DSINFO screen
 COMMAND=RETURN             6. Return for second data set DSINFO screen
 SCROLL                     7. Scroll through all of the data sets
 COMMAND=END                8. End the batch session
 /*
```

The following describes each line of the control card:

1. Issues the VTOC command for volume MVSRES.
2. Places the DSI line command in the command field for the first data set on the VTOC display. Because this card has a nonblank character in column 72, processing proceeds to the next card.
3. Places the DSI line command in the command field for the second and third data sets on the VTOC display. Because there are no COMMAND keywords and the card is not continued, the response is terminated with this card. DSINFO is displayed for the first data set.

4. The DSINFO display for the first data set is ended and DSINFO is issued for the second data set.
5. The DSINFO display for the second data set is ended and DSINFO is issued for the third data set.
6. The DSINFO display for the third data set is ended, returning back to the VTOC display.
7. Issues multiple DOWN commands until all of the data sets are displayed.
8. Ends the CA SYSVIEW batch session.

EXECBAT Member—Start the Batch Interface

The EXECBAT member in the SAMPLIB data set can be used to run the product as a batch job. The following example shows the JCL necessary to start the batch interface:

```
//EXECBAT JOB ,pgmrname,USER=userid
//STEP1 EXEC PGM=GSVXBAT,PARM='LINECNT=66,SHOWINP=NO'
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
input control cards
//
```

The user ID field on the JOB statement controls the user ID for the batch session. The actual field used for the user ID comes from the ACEE security control block. If there is no security system installed, the batch interface uses a default user ID of ++++++++.

Initialization Parameters

The following initialization parameters can be specified with the PARM parameter on the EXEC statement:

LINECNT

Simulates the number of lines available on a screen and controls how many lines are written on a page. The LINECNT parameter must be at least 20 and must not exceed 100. The record length of the SYSPRINT DD statement controls the line size. The default is 66.

MENUON and MENUOFF

Controls the printing of any MENU command output to the SYSPRINT data set.

Default: MENUON

PRINTON and PRINTOFF

Controls the printing of all command output to the SYSPRINT data set. PRINTOFF also suppresses the printing of the internal COMMAND=DOWN statements generated by the SCROLL keyword.

Default: PRINTON

SHOWINP

Controls whether input control cards are displayed in the output. If the control cards are displayed, they are shown after the screen they would have been entered on. Specify YES to show the input control cards. Specify NO to exclude the control cards from the output.

GSVXBAT DD Statements

The GSVXBAT module requires two DD statements:

SYSPRINT

Used for screen output and any messages. If DCB parameters are not coded, the default DCB parameters are LRECL=133, BLKSIZE=1330, RECFM=FBA. The record format cannot be variable. The LRECL parameter controls the line size of the output screen. If RECFM includes the ASA specification, the line size is LRECL minus one. The LRECL parameter must be at least 80 and cannot exceed 255. The SYSPRINT DD can be either a SYSOUT or disk data set.

SYSIN

Used for input control cards. Control cards are limited to 80 columns and the record format must be fixed.

Input Card Keywords

Use the following keywords on an input control card:

COMMAND

Used to enter commands typically entered on the command line.

DATA

Used to enter input data that is in the body of a display. The parameter of the DATA keyword has the following format:

```
(subscript,input)
```

To determine the subscript parameter:

1. Assign a 1 to the first input field in the body of a display screen.
2. Add 1 for each input field after the first input field.

The order of the input fields is from left to right, top to bottom. The input parameter is the input data that would typically be entered in the input field.

MENUON and MENUOFF

Controls the printing of any MENU command output to the SYSPRINT data set.

Default: MENUON

PRINTON and PRINTOFF

Controls the printing of all command output to the SYSPRINT data set. PRINTOFF also suppresses the printing of the internal COMMAND=DOWN statements generated by the SCROLL keyword.

Default: PRINTON

SCROLL

Simulates a scrolling action until the end of the data is reached. This keyword issues successive DOWN commands until all of the data is displayed. For example, if you issued the VTOC command and you wanted to display all of the data sets, use the SCROLL keyword. The SCROLL keyword is not valid when the DUMP command is active.

Rules for Coding Batch Input Control Cards

Use the following rules when coding the batch input control cards:

- An asterisk (*) in column 1 indicates a comment card.
- Commas must be used to separate keyword statements.
- COMMAND parameters that contain embedded blanks or commas must be enclosed in parentheses.

- A blank terminates the processing of a control card unless the blank is part of a keyword parameter that is enclosed in parentheses. Any text following a blank is considered a comment.
- A response is considered complete when a COMMAND keyword is encountered.
- To continue input control cards, code a nonblank character in column 72 of the control card.
- If a control card is not continued and a COMMAND keyword has not been encountered, the response is considered complete.
- Any keywords after the first COMMAND keyword on a control card are ignored.
- Keyword statements cannot be split across control cards. They must fit on one control card.
- Keyword statements that result in an error condition are ignored and the next keyword statement is processed.

Return Codes from GSVXBAT

The following return codes are received from the GSVXBAT module:

0

Processing completed with no errors.

4

An error was detected during GSVXBAT initialization. Look for messages either in the job log or the SYSPRINT output.

8

An error was detected while processing the input control cards. Look for messages in the SYSPRINT output.

12

Errors were detected during initialization and while processing the input control cards. Look for messages in the job log and the SYSPRINT output.

16

An abend occurred during GSVXBAT initialization. Check for messages in either the job log or the SYSPRINT output.

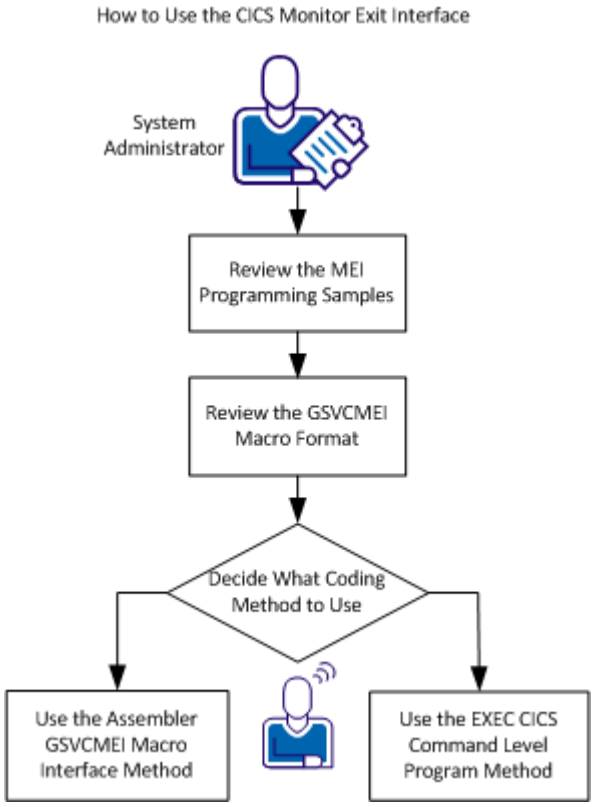
Chapter 13: Using the CICS Monitor Exit Interface

This section contains the following topics:

How to Use the CICS Monitor Exit Interface

The CA SYSVIEW for CICS Monitor Exit Interface (MEI) provides CICS programs or transactions an API into the CA SYSVIEW for CICS data collection process.

This process shows how to code MEI functions to communicate with the MEI. This interface provides valuable insight and status information.



The following process provides the information necessary to communicate with the CICS MEI.

- Review the Monitor Exit Interface Programming Samples (see page 143)
- Review the GSVCMEI Macro Format (see page 144)
- Decide What Coding Method to Use (see page 146)
- Use the Assembler GSVCMEI Macro Interface Coding Method (see page 147)
- Use the EXEC CICS Command Level Program Coding Method (see page 157)

Review the Monitor Exit Interface Programming Samples

Review the following programming samples to help you understand how to code the CICS MEI:

- Monitor File Requests (see page 143)
- Correlate Transaction Information (see page 143)

Monitor File Requests to Non CICS Resources

You can monitor file or database requests to non-CICS resources. Many applications use files or databases that are not under the control of CICS. The MEI can be used to monitor non-CICS file requests.

Follow these steps:

1. Insert an MEI Start Clock request before the file request.
2. Make the file request.
3. Insert an MEI Stop Clock request after the file request.

Correlate Transaction Information

A transaction can start one or more associated transactions that logically work together to complete work. You can use various methods to start additional transactions. The goal is to be able to associate all the transactions into a logical group using a correlation ID.

This procedure assumes that a started transaction contains a correlation ID inserted by the CA APM product.

The first transaction starts one or more transactions. The goal is to be able to correlate all associated transactions after completion.

Follow these steps:

1. Insert an MEI Get Field request to obtain the current correlation ID in the first transaction.
2. Pass the correlation ID to the additional transactions when starting. The method to pass the information to the additional transaction varies based on how the transactions are being started.
3. Insert an MEI Set Field request to set the correlation ID in the newly started transactions.

   The transaction information is correlated for transactions that run under different units of work.

GSVCMEI Macro—Interface to the Monitor Exit

The macro GSVCMEI is provided with CA SYSVIEW and provides an interface to its CICS monitor exit. Add the GSVCMEI macro to your existing packages to perform its functions.

Review the GSVCMEI macro format for the resources available for use when writing programs to communicate with the Monitor Exit Interface.

The GSVCMEI macro has the following format:

```
....+....1....+....2....+....3....+....4....+....5....+....6
MEIC                GSVCMEI GEN_COMMAREA,DSECT=NO
           EXEC CICS LINK PROGRAM('GSVCMEI')
                COMMAREA(MEIC)
                LENGTH(=AL2(MEIC_L))

MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO
           GSVCMEI function,                    Call MEI
                NAME=,
                TYPE=,
                FIELD=,
                DATA=,
                PL=,
                OKRET=,
                WNRET=,
                ERRET=,
```

MEIC

The MEIC portion of the GSVCMEI macro demonstrates the command-level interface:

- Generate MEI COMMAREA to use with the CICS command-level interface:

  ```
  ....+....1....+....2....+....3....+....4....+....5....+....6
  MEIC                GSVCMEI GEN_COMMAREA,DSECT=NO
  ```

- CICS command-level interface to MEI:

  ```
  ....+....1....+....2....+....3....+....4....+....5....+....6
             EXEC CICS LINK PROGRAM('GSVCMEI')
                  COMMAREA(MEIC)
                  LENGTH(=AL2(MEIC_L))
  ```

MEIPL

The MEIPL portion of the GSVCMEI macro demonstrates the assembler macro interface:

- Generate MEI parameter list to use with the assembler macro interface:

  ```
  ....+....1....+....2....+....3....+....4....+....5....+....6
  MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO
  ```

- Generate MEI assembler macro interface calls:

```
....+....1....+....2....+....3....+....4....+....5....+....6
          GSVCMEI function,                Call MEI
                NAME=,
                TYPE=,
                FIELD=,
                DATA=,
                DATALEN=,
                PL=,
                OKRET=,
                WNRET=,
                ERRET=,
```

function

Note: For usage details on these functions, see the programming examples in topics Use the Assembler Macro Interface Coding Method (see page 147) or Use the EXEC CICS Command Level Program Coding Method (see page 157).

The following functions are available:

SET_UMBRELLA_NAME - Provides an umbrella transaction name and optionally an umbrella type to associate with the transaction.

PROGRAM_USAGE - Provides the name of a program that is being used. This function tracks program usage within other products, such as fourth-generation products.

START_CLOCK - Starts the timing of an event or function. After the timed event or function completes, issue a STOP_CLOCK function call. The START_CLOCK function can be nested.

STOP_CLOCK - Stops the timing of an event or function. Precede each STOP_CLOCK function with a START_CLOCK function call.

START_EVENT - Starts the timing of an event or function. After the timed event or function completes, issue a STOP_EVENT function call. The START_EVENT function can be nested. Each event can be assigned a 16-character event name and a 16-character event type.

STOP_EVENT - Stops the timing of an event or function. Precede each STOP_EVENT function with a START_EVENT function call.

SET_FIELD - Provides data to store in the requested collection field.

GET_FIELD - Retrieves data from the requested collection field.

SET_USERDATA - Provides a user-defined data value to be associated with the executing transaction.

GET_USERDATA - Retrieves a previously stored user-defined data value for the executing transaction.

NAME=

Specifies the resource name for the function being called.

TYPE=

Specifies a type value, relative to the function being called and the NAME= value provided.

FIELD=

Specifies the field code

DATA=

Specifies the data value for function calls SET_FIELD, GET_FIELD, SET_USERDATA, GET_USERDATA.

DATALEN=

Specifies the length of the DATA= value for the SET_USERDATA and GET_USERDATA function calls.

PL=

Specifies the parameter list name. The default is MEIPL.

OKRET=

Specifies a branch label to use when the return code equals zero (0).

WNRET=

Specifies a branch label to use when the return code equals four.

ERRET=

Specifies a branch label to use when the return code is not equal to zero or four.

Note: The MEISAMP member in the sample library contains sample MEI calls.

More information:

Decide What Coding Method to Use (see page 146)

Decide What Coding Method to Use

Use one of the following two coding methods provided in the GSVCMEI macro within a single transaction or program:

- EXEC CICS command level program and a standard CICS COMMAREA
- Assembler macro interface

More information:

GSVCMEI Macro—Interface to the Monitor Exit (see page 144)

Use the Assembler Macro Interface Coding Method

The functions that follow are coded using the assembler macro interface method.

Set Umbrella Name and Type

Many CICS transactions execute using the same transaction ID but perform different functions. This type of transaction is referred to as an umbrella transaction or menu transaction.

Setting the umbrella name and or type provides additional information about the actual purpose of the transaction.

Use the following assembler GSVCMEI macro SET_UMBRELLA_NAME example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG          DSECT ,
MEIPL             GSVCMEI GEN_PARMLIST,DSECT=NO

module   DFHEIENT

         GSVCMEI SET_UMBRELLA_NAME,            Set umbrella name  X
              NAME=CL08'name',                 ..umbrella name    X
              TYPE=CL08'type'                  ..umbrella type

         EXEC CICS RETURN
```

UMBRELLA_NAME parameters:

name

Specify an eight-character umbrella transaction name.

type

(Optional) Specify an eight-character umbrella transaction type.

Record Non CICS Program Usage

Many CICS transactions execute programs that are not defined to CICS. Knowing the names of non-CICS or external programs that are being used lets you track their usage.

Use the following assembler GSVCMEI macro PROGRAM_USAGE example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG            DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO

module   DFHEIENT

                GSVCMEI PROGRAM_USAGE,          Program usage     X
                NAME==CL08'program'             ..program name

         EXEC CICS RETURN
```

Program usage parameters:

program

Specifies an eight-character program name.

Start Timing a File or Database Request

Many CICS transactions make file or database requests that are not performed using CICS functions.

The start clock function request does the following:

- Lets the transaction communicate the non-CICS file requests to the MEI
- Records the elapsed time of the request
- Optionally, records the request type

Pair this start clock MEI request with the stop clock request to stop the timing of the request.

Use the following assembler GSVCMEI macro START_CLOCK example to code this function:

```
reqtype - Request Type Keywords
           Valid values are:
           NONE
           ADD
           BROWSE
           DELETE
           READ
           READUPD
           UPDATE

....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO

module   DFHEIENT

        GSVCMEI START_CLOCK,                    Start clock       X
               NAME==CL08'filename'             ..filename        X
               TYPE=reqtype                     ..request type

        EXEC CICS RETURN
```

Stop Timing a File or Database Request

Many CICS transactions make file or database requests that are not performed using CICS functions.

The stop clock function request does the following:

- Lets the transaction communicate the non-CICS file requests to the MEI
- Records the elapsed time of the request
- Optionally, records the request type

Pair this stop clock MEI request with the start clock request to start the timing of the request.

Use the following assembler GSVCMEI macro STOP_CLOCK example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO
module   DFHEIENT
         GSVCMEI STOP_CLOCK              Stop clock
         EXEC CICS RETURN
```

Start Timing an Event

The START_EVENT function starts the recording of elapsed time for any generic event.

Pair the start event MEI request with the request to stop the timing of the event.

Use the following assembler GSVCMEI macro START_EVENT example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO

module   DFHEIENT

        GSVCMEI START_EVENT,                    Start event       X
                NAME==CL16'eventname',          ..event name      X
                TYPE==CL16'eventtype'           ..event type

        EXEC CICS RETURN
```

START_EVENT parameters:

eventname

Specify a 16-character event name. The event name must be the same as the event name used in the STOP_EVENT function call corresponding to the START_EVENT function call.

eventtype

Specify a 16-character event type.

Stop Timing an Event

The STOP_EVENT function stops the recording of elapsed time for any generic event.

Pair this STOP_EVENT MEI request with the START_EVENT request to stop the timing of the event.

Use the following assembler GSVCMEI macro STOP_EVENT example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO

module  DFHEIENT

        GSVCMEI STOP_EVENT,                     Stop event        X
                NAME==CL16'eventname',          ..event name      X
                TYPE==CL16'eventtype'           ..event type

        EXEC CICS RETURN
```

STOP_EVENT parameters:

eventname

Specify a 16-character event name. The event name must be the same as the event name used in the START_EVENT function call corresponding to the STOP_EVENT function call.

eventtype

Specify a 16-character event type.

Set Field Data Value for User ID

The SET_FIELD function sets or overrides the values for specific collected fields.

Use the following assembler GSVCMEI macro SET_FIELD example to code this function:

```
field   - Field Type Keywords
            Valid values are:
             CORID
             LUNAME
             OPERID
             TERMINAL
             UMBNAME
             UMBTYPE
             USERID
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO

module   DFHEIENT

        GSVCMEI SET_FIELD,                      Set field         X
        FIELD=USERID,                           ..field name      X
        DATA==CL08'userid'                      ..data value

        EXEC CICS RETURN
```

Get Field Data Value for User ID

The GET_FIELD function retrieves the values for specific collected fields.

Use the following assembler GSVCMEI macro GET_FIELD example to code this function:

```
field   - Field Type Keywords
          Valid values are:
           CORID
           LUNAME
           OPERID
           TERMINAL
           UMBNAME
           UMBTYPE
           USERID

....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG   DSECT ,
MEIPL              GSVCMEI GEN_PARMLIST,DSECT=NO
DSA_UserID         DS    CL8

module   DFHEIENT

        GSVCMEI GET_FIELD,                      Get field         X
              FIELD=USERID,                     ..field name      X
              DATA=DSA_UserID                   ..output area

        EXEC CICS RETURN
```

Set Field Data Correlation ID

This function sets a correlation ID for the transaction.

Use the following assembler GSVCMEI macro SET_FIELD example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO
DSA_CorID           DS    0CL130
DSA_CorID_Type      DS    AL1
DSA_CorID_Len       DS    AL1
DSA_CorID_Data      DS    CL128

module   DFHEIENT

         MVI   DSA_CorID_Type,MEIC_F005_CorID_User
         MVI   DSA_CorID_Len,16
         MVC   DSA_CorID_Data(16),=CL16'1234567890123456'

         GSVCMEI SET_FIELD,                    Set field         X
                FIELD=CORID,                   ..field name      X
                DATA=DSA_CorID                 ..data value

         EXEC CICS RETURN
```

Get Field Data Correlation ID

The get field data correlation ID function retrieves a correlation ID for the transaction.

Use the following assembler GSVCMEI macro GET_FIELD example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO
DSA_CorID           DS    0CL130
DSA_CorID_Type      DS    AL1
DSA_CorID_Len       DS    AL1
DSA_CorID_Data      DS    CL128

module   DFHEIENT

        GSVCMEI GET_FIELD,                     Get field         X
                FIELD=CORID,                   ..field name      X
                DATA=DSA_CorID                 ..data value

        EXEC CICS RETURN
```

Set User Data

The SET_USERDATA function sets or saves data values associated with an executing transaction.

Use the following assembler GSVCMEI macro SET_USERDATA example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO

module   DFHEIENT

        GSVCMEI SET_USERDATA,                   Set user data value X
                NAME==CL32'dataelementname',    ..data element name X
                TYPE=EBCDIC,                    ..data type         X
                DATA=CL256'datavalue',          ..data value        X
                DATALEN==Y(256)                 ..data value length

        EXEC CICS RETURN
```

Get User Data

The GET_USERDATA function retrieves data values associated with an executing transaction.

Use the following assembler GSVCMEI macro GET_USERDATA example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIPL               GSVCMEI GEN_PARMLIST,DSECT=NO
DSA_Data_Type       DS    AL1
DSA_Data_Len        DS    AL2
DSA_Data_Value      DS    CL256

module   DFHEIENT

        GSVCMEI GET_USERDATA,                   Get user data value X
                NAME==CL32'dataelementname',    ..data element name X
                TYPE=DSA_Data_Type,             ..data type         X
                DATA=DSA_Data_Value,            ..data value        X
                DATALEN=DSA_Data_Len            ..data value length

        EXEC CICS RETURN
```

Use the EXEC CICS Command Level Program Coding Method

The functions that follow are coded using the CICS command level program method.

A CICS program definition is required to implement the CICS command-level interface.

The following definition is included during the installation process:

```
DEFINE PROGRAM(GSVCMEI)
        LANGUAGE(ASSEMBLER)
        DATALOCATION(ANY)
        EXECKEY(CICS)
        CONCURRENCY(THREADSAFE)
        GROUP(SYSVIEW)
```

Set Umbrella Name and Type

Many CICS transactions execute using the same transaction ID but perform different functions. This type of transaction is referred to as an umbrella transaction or menu transaction.

Setting the umbrella name and or type provides additional information about the actual purpose of the transaction.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG            DSECT ,
MEIC                GSVCMEI GEN_COMMAREA,DSECT=NO
module  DFHEIENT
        MVC   MEIC_Function,=AL4(MEIC_SetUmbrella)       Function
        MVC   MEIC_F001_Name,=CL08'name    '             Name
        MVC   MEIC_F001_Type,=CL08'type    '             Type

        EXEC CICS LINK PROGRAM('GSVCMEI')                          X
                COMMAREA(MEIC)                                     X
                LENGTH(=AL2(MEIC_L))

        EXEC CICS RETURN
```

UMBRELLA_NAME parameters:

name

Specify an eight-character umbrella transaction name.

type

(Optional) Specify an eight-character umbrella transaction type.

Record Non CICS Program Usage

Many CICS transactions execute programs that are not defined to CICS. Knowing the names of non-CICS or external programs that are being used lets you track their usage.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIC                GSVCMEI GEN_COMMAREA,DSECT=NO

module   DFHEIENT

         MVS   MEIC_Function,=AL4(MEIC_ProgramUse)        Function
         MVC   MEIC_F002_Program,=CL08'program'           Program

         EXEC CICS LINK PROGRAM('GSVCMEI')                          X
              COMMAREA(MEIC)                                        X
              LENGTH(=AL2(MEIC_L))

         EXEC CICS RETURN
```

Program usage parameters:

program

Specifies an eight-character program name.

Start Timing a File or Database Request

Many CICS transactions make file or database requests that are not performed using CICS functions.

The start clock function request does the following:

- Lets the transaction communicate the non-CICS file requests to the MEI
- Records the elapsed time of the request
- Optionally, records the request type

Pair this start clock MEI request with the stop clock request to stop the timing of the request.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIC                GSVCMEI GEN_COMMAREA,DSECT=NO

module   DFHEIENT

        MVC   MEIC_Function,=AL4(MEIC_StartClock)        Function
        MVC   MEIC_F003_Request,=AL4(MEIC_F003_Read)     Request
        MVC   MEIC_F003_File,=CL08'filename'             File name

        EXEC CICS LINK PROGRAM('GSVCMEI')                          X
             COMMAREA(MEIC)                                        X
             LENGTH(=AL2(MEIC_L))

        EXEC CICS RETURN
```

Stop Timing a File or Database Request

Many CICS transactions make file or database requests that are not performed using CICS functions.

The stop clock function request does the following:

- Lets the transaction communicate the non-CICS file requests to the MEI.
- Records the elapsed time of the request
- Optionally, records the request type

Pair this stop clock MEI request with the start clock request to start the timing of the request.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG   DSECT ,
MEIC               GSVCMEI GEN_COMMAREA,DSECT=NO

module   DFHEIENT

        MVC   MEIC_Function,=AL4(MEIC_StopClock) Function

        EXEC CICS LINK PROGRAM('GSVCMEI')                          X
             COMMAREA(MEIC)                                        X
             LENGTH(=AL2(MEIC_L))

        EXEC CICS RETURN
```

Start Timing an Event

The START_EVENT function starts the recording of elapsed time for any generic event.

Pair the start event MEI request with the request to stop the timing of the event.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG           DSECT ,
MEIC               GSVCMEI GEN_COMMAREA,DSECT=NO

module   DFHEIENT

         MVC   MEIC_Function,=AL4(MEIC_StartEvent)       Function
         MVC   MEIC_F006_Name,=CL16'eventname'           name
         MVC   MEIC_F006_Type,=CL16'eventtype'           type

         EXEC CICS LINK PROGRAM('GSVCMEI')                        X
              COMMAREA(MEIC)                                      X
              LENGTH(=AL2(MEIC_L))

         EXEC CICS RETURN
```

START_EVENT parameters:

eventname

Specify a 16-character event name. The event name must be the same as the event name used in the STOP_EVENT function call corresponding to the START_EVENT function call.

eventtype

Specify a 16-character event type.

STOP_EVENT Function—Stop Timing the Event

The STOP_EVENT function stops the recording of elapsed time for any generic event.

Pair this STOP_EVENT request with the START_EVENT request to stop the timing of the event.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG   DSECT ,
MEIC               GSVCMEI GEN_COMMAREA,DSECT=NO

module   DFHEIENT

        MVC   MEIC_Function,=AL4(MEIC_StopEvent) Function
        MVC   MEIC_F007_Name,=CL16'eventname'           name
        MVC   MEIC_F007_Type,=CL16'eventtype'           type

        EXEC CICS LINK PROGRAM('GSVCMEI')                        X
             COMMAREA(MEIC)                                      X
             LENGTH(=AL2(MEIC_L))

        EXEC CICS RETURN
```

STOP_EVENT parameters:

eventname

Specify a 16-character event name. The event name must be the same as the event name used in the START_EVENT function call corresponding to the STOP_EVENT function call.

eventtype

Specify a 16-character event type.

Set Field Data Value for User ID

The SET_FIELD function sets or overrides the values for specific collected fields.

Use the following CICS command level program example to code this function:

```
field   - Field Type Keywords
            Valid values are:
             CORID
             LUNAME
             OPERID
             TERMINAL
             UMBNAME
             UMBTYPE
             USERID

....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIC                GSVCMEI GEN_COMMAREA,DSECT=NO

module   DFHEIENT

        MVC   MEIC_Function,=AL4(MEIC_SetField)      Function
        MVC   MEIC_F005_FldCode,=AL4(MEIC_F005_FC_UserID)
        MVC   MEIC_F005_UserID,=CL08'userid'         UserID

        EXEC CICS LINK PROGRAM('GSVCMEI')                        X
             COMMAREA(MEIC)                                      X
             LENGTH(=AL2(MEIC_L))

        EXEC CICS RETURN
```

Get Field Data Values for User ID

The GET_FIELD function retrieves the values for specific collected fields.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG   DSECT ,
MEIC               GSVCMEI GEN_COMMAREA,DSECT=NO
DSA_UserID         DS    CL8

module   DFHEIENT

        MVC   MEIC_Function,=AL4(MEIC_GetField)      Function
        MVC   MEIC_F008_FldCode,=AL4(MEIC_F008_FC_UserID)

        EXEC CICS LINK PROGRAM('GSVCMEI')                          X
             COMMAREA(MEIC)                                        X
             LENGTH(=AL2(MEIC_L))

        MVC   DSA_UserID,MEIC_F008_UserID               UserID

        EXEC CICS RETURN
```

Set Field Data Correlation ID

This function sets a correlation ID for the transaction.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIC               GSVCMEI GEN_COMMAREA,DSECT=NO

module   DFHEIENT

        MVC   MEIC_Function,=AL4(MEIC_SetField)       Function
        MVC   MEIC_F005_FldCode,=AL4(MEIC_F005_FC_CorID)
        MVI   MEIC_F005_CorID_Type,MEIC_F005_CorID_User
        MVI   MEIC_F005_CorID_Len,16

        MVC   MEIC_F005_CorID_Data(16),=CL16'1234567890123456'

        EXEC CICS LINK PROGRAM('GSVCMEI')                          X
              COMMAREA(MEIC)                                       X
              LENGTH(=AL2(MEIC_L))

        EXEC CICS RETURN
```

Get Field Data Correlation ID

The get field data correlation ID function retrieves a correlation ID for the transaction.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG    DSECT ,
MEIC              GSVCMEI GEN_COMMAREA,DSECT=NO
DSA_CorID         DS   0CL130
DSA_CorID_Type    DS   AL1
DSA_CorID_Len     DS   AL1
DSA_CorID_Data    DS   CL128

module   DFHEIENT

        MVC   MEIC_Function,=AL4(MEIC_GetField)  Function
        MVC   MEIC_F008_FldCode,=AL4(MEIC_F008_FC_CorID)

        EXEC CICS LINK PROGRAM('GSVCMEI')                       X
             COMMAREA(MEIC)                                     X
             LENGTH(=AL2(MEIC_L))

        MVC   DSA_CorID,MEIC_F008_CorID

        EXEC CICS RETURN
```

Set User Data

The SET_USERDATA function sets or saves data values associated with an executing transaction.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG           DSECT ,
MEIC               GSVCMEI GEN_COMMAREA,DSECT=NO

module   DFHEIENT

         MVC   MEIC_Function,=AL4(MEIC_SetUserData)   Function
         MVC   MEIC_F009_Name,=CL32'dataelementname'
         MVC   MEIC_F009_Type,=AL1(MEIC_F009_EBCDIC)
         MVC   MEIC_F009_DataLen,=Y(256)
         MVC   MEIC_F009_Data(256),=CL256'datavalue'

         EXEC CICS LINK PROGRAM('GSVCMEI')                      X
                COMMAREA(MEIC)                                  X
                LENGTH(=AL2(MEIC_L))

         EXEC CICS RETURN
```

Get User Data

The GET_USERDATA function retrieves data values associated with an executing transaction.

Use the following CICS command level program example to code this function:

```
....+....1....+....2....+....3....+....4....+....5....+....6||7..
DFHEISTG   DSECT ,
MEIC               GSVCMEI GEN_COMMAREA,DSECT=NO

module   DFHEIENT

         MVC   MEIC_Function,=AL4(MEIC_GetUserData)   Function
         MVC   MEIC_F010_Name,=CL32'dataelementname'

         EXEC CICS LINK PROGRAM('GSVCMEI')                      X
               COMMAREA(MEIC)                                   X
               LENGTH(=AL2(MEIC_L))

         EXEC CICS RETURN
```

Return Codes from MEI

The CA SYSVIEW for CICS Monitor Exit Interface will return the resulting return code from the MEI call differently depending on the method used to invoke the MEI interface.

- CICS command level interface

  The return code will be returned in the COMMAREA in the following field:

  `MEIC_RC`

- Assembler macro interface

  The return code will be returned through register 15.

The following are the return codes and descriptions:

00 x00

Okay

04 x04

Warning

08 x08

No commarea

12 x0C

Unable to locate anchor

16 x10

Unable to obtain workarea

20 x14

Workarea initialization failed

24 x18

Abend

28 x1C

Invalid function code

32 x20

Unable to locate task pointers

36 x24

Unable to locate monitor block

40 x28

Invalid commarea length

44 x2C

Error allocating segment

48 x30

Requested segment not found

52 x34

Requested event not started

56 x38

Invalid field type

60 x3C

Invalid parameter list

64 x40

Cannot add segment, full

68 x44

Invalid data type

72 x48

Invalid data length

76 x4C

Data type mismatch

80 x50

Data length mismatch

Chapter 14: Persistent Data Store Interface

CA SYSVIEW maintains various types of data collection, statistics, and configuration options that control the operation of the product. In most cases, these data objects are unique to a single iteration of a CA SYSVIEW execution.

The persistent data store environment requires low maintenance and enhances CA SYSVIEW by providing the following benefits:

- Maintains data objects across multiple executions of CA SYSVIEW
- Maintains data objects across an operating system IPL
- Shares the data store data set across multiple instances of CA SYSVIEW running in the same system or within the same sysplex

This section contains the following topics:

How Persistent Data Works

The CA SYSVIEW persistent data store interface lets you save and retrieve data from the persistent data store.

The data becomes persistent through the following process:

1. The original CA SYSVIEW configuration definitions are stored in various PARMLIB members.
2. The configuration information is saved during termination.
3. The saved configuration information is then persistent.
4. CA SYSVIEW retrieves the persistent data during initialization.

Persistent Data Store Components

The persistent data store contains the following components:

DATALIB

Stores and maintains all information about the data objects. You can have one or multiple DATALIB data sets. Any one instance of SYSVIEW can use only one DATALIB.

The administrator can determine whether to have a unique DATALIB for each instance of SYSVIEW or share one of the DATALIBs.

One CA SYSVIEW can use only one DATALIB. You can share a DATALIB within a sysplex but not across multiple sysplexes. Multiple releases of CA SYSVIEW can share the DATALIB, making migration to a new release easier.

Data Store

Serves as the repository for data objects. The data store is a standard PDSE data set allocated during installation. At completion of the installation process, the data set is empty. No initial data is required nor is any data shipped with the product.

Data Object

Contains any logical piece of data.

DataId

Specifies the internal data identifier of the data object maintained by the data store.

DataKey

Specifies the data object key that the data store maintains. The DataKey includes the DataId plus additional information to help ensure a unique key across multiple instances of CA SYSVIEW.

Logical Components

Lets CA SYSVIEW components read and write to and from the data store through programs or callable subroutines.

DATALIB Members

The DATALIB contains the following two types of members:

- A single index member named $$$INDEX

  Contains all of the information or knowledge about all data objects being stored in the DATALIB. A meaningful member name for a data object cannot be created in only eight characters. For this reason, the index member is used to maintain an index of the Data Object Members. The creator of a data object assigns a unique data ID to the data object.

  The following is the internal key for a data object:

  `dataid.sysname.ssid.build`

- Multiple data object members named from D0000001 to D9999999

  Contains the persistent data. Additional internal information is added to the object. The data object cannot be accessed directly. The member name of the data object is dynamically assigned through the index member.

Configure the DATALIB Data Set

The persistent data store DATALIB must be configured for your installation.

Follow these steps:

1. Determine how many DATALIBs you need.
2. Allocate the DATALIB as a PDSE using the following naming conventions:

   sysview.CNM4BDAT
3. Specify the name of the DATALIB in the System Configuration Options member using the Dsn-System-DATALIB keyword.

   At completion of the installation process, the data set is empty.

The configured DATALIB data set contains all information about data being maintained in the data store. No other associations are required.

Security Requirements

A universal read access to the DATALIB data set is recommended because multiple users, address spaces, and jobs read the data store.

The DATALIB subtask of the SYSVIEW Main Services address space performs all write functions to the DATALIB. Therefore, only the user ID associated with the SYSVIEW Main Services address space is required to have update access to the DATALIB data set.

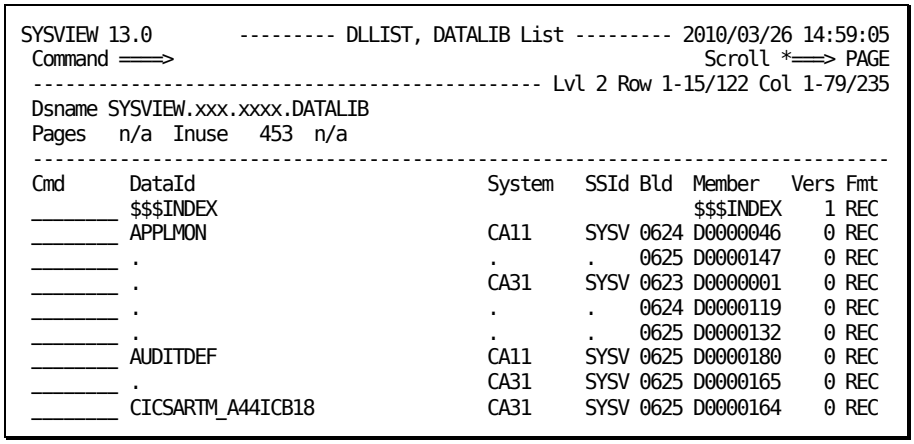
Access and Use the DATALIB Menu

The persistent data store provides an environment named the DATALIB for maintaining data objects. Data can be saved, retrieved, and copied from the persistent data store. Some DATALIB monitoring can be required to watch for a full condition.

Follow these steps:

1. Enter DLLIST command

   A menu similar to the following displays the contents of the CA SYSVIEW DATALIB library:



```
SYSVIEW 13.0        --------- DLLIST, DATALIB List --------- 2010/03/26 14:59:05
 Command ===>                                                  Scroll *===> PAGE
 ------------------------------------------------ Lvl 2 Row 1-15/122 Col 1-79/235
 Dsname SYSVIEW.xxx.xxxx.DATALIB
 Pages   n/a  Inuse   453  n/a
 ------------------------------------------------------------------------------
 Cmd      DataId                          System   SSId Bld  Member   Vers Fmt
 _______  $$$INDEX                                             $$$INDEX    1 REC
 _______  APPLMON                         CA11     SYSV 0624 D0000046    0 REC
 _______  .                               .        .    0625 D0000147    0 REC
 _______  .                               CA31     SYSV 0623 D0000001    0 REC
 _______  .                               .        .    0624 D0000119    0 REC
 _______  .                               .        .    0625 D0000132    0 REC
 _______  AUDITDEF                        CA11     SYSV 0625 D0000180    0 REC
 _______  .                               CA31     SYSV 0625 D0000165    0 REC
 _______  CICSARTM_A44ICB18               CA31     SYSV 0625 D0000164    0 REC
```

2. Use the scroll function commands (top, bottom, right, left, up, and down) to navigate the displays.

3. Enter the line commands in the Cmd input field. The following are a few valid line commands:

   - Question mark (?)

     Displays a list of available line commands.

   - = (Equal)

     Repeats any previously executed line command.

   - V (View) or S (Select)

     Retrieves and displays the contents of the data object member.

   - C (Copy)

     Copies a data object using the DATALIB Copy menu.

   - R (Rename)

     Renames a data object using the DATALIB Rename menu.

4. Press PF3 to back out of the displays.

The tasks have been completed and the DATALIB menu closes.

Checkpoint Function—Request a Checkpoint from Product Tasks

You can write configuration data to the persistent data store using the CHECKPOINT function of the MODIFY command.

Use the following format to request a checkpoint on the MVSDATA task:

```
MODIFY sysview,,MODIFY MVSDATA,CHECKPOINT <ALL|MONITOR|THRESHOLD|STATES>
```

You can request a checkpoint from the following CA SYSVIEW tasks:

- MVSDATA
- MQSDATA
- IMSDATA
- TCPDATA
- SCHEDULER

Chapter 15: Performance IMODs

CA GSS lets you execute intelligent modules (IMODs). IMODs are written in the REXX language, together with additional functions and instructions supplied by CA.

For complete information about IMODs, see the CA Common Services for z/OS documentation.

Important! When upgrading to a new release of CA SYSVIEW, the new release and the IMOD libraries shipped with the new release match. The new IMOD libraries overlay and therefore replace the IMOD libraries from the previous release. If you decide to modify the IMOD libraries, CA suggests creating a site-specific IMOD library for those locally written IMODs.

This section contains the following topics:

Purpose of IMODs

Use IMODs to automate system monitoring and regulate resources or to create your own online reports on system activities.

The SCM uses IMODs to monitor multiple subsystems and collect information from a single screen. The IMODs determine:

- What alerts to create
- The level of alert: Normal, Warning, Problem, Action
- The color coding, based on comparing the resource status against thresholds

Display Available IMODs

The IMOD Selection Panel lets you see a list of all available IMODs.

Follow these steps:

1. From the Primary Options Menu, enter **PRODUCTS** on the command line.

   The PRODUCTS External Applications menu displays.

2. From the PRODUCTS External Applications menu, select the IMOD option by entering an **S** in the Cmd area.

3. Select SYSVIEWE by entering an **S** in the Cmd area.
4. Locate the SCM IMODs by entering **L SCM** from the command line.

   The SCM IMODS are displayed.

Execute an IMOD from the IMOD Editor

An ISPF error message referring to ISPPROF when you select the IMOD editor could indicate insufficient space in the ISPPROF data set.

When you cannot reload or execute an IMOD from the IMOD editor, reload or execute the IMOD from the IMOD editor.

Follow these steps:

1. Press PF1 (or enter HELP) to display the long error message that identifies the cause of the failure.
2. Verify that CA-GSS is executing.

   This step requires that the GSSMAIN program is running.
3. Use the sample RUNPARM, ISETS, and EDITPARM members.

   These members verify that the ISERVE and DSNAME references are identical to those appearing on the ISET statements defined to the CA-GSS address space.

Display Sample Parameters for CA GSS

CA SYSVIEW provides SAMPLIB member GSS in sysview.CNM4BSAM. This member contains all required parameter statements to customize CA GSS for use with CA SYSVIEW. You can display the list of SAMPLIB parameter statements.

Follow these steps:

1. Enter **LIBS** on the command line.

   The LIBS, Product Libraries menu displays.
2. Select the SAMPLIB option by entering a **B** (Browse) in the command area and then press enter.
3. Select the statement you want to view by entering **S** on the line and then press enter.

   The parameter statement is displayed.

System Condition Monitor IMODs

The SCM monitors the IMODs that are defined and executed in the SCM parmlib member. The IMOD Selection Panel shows you a list of all IMODs that you can access. To view the SCM IMODS, see Display Available IMODs (see page 175).

- You can use the following IMODs for the System Condition Monitor:

  $CANCEL

  Issues the Cancel command based on address space type.

  Example:

  ```
  CALL $CANCEL jobname
  ```

  $NOTIFY

  Sends a notify message to a user or group. The $NOTIFY IMOD contains a table defining groups of users. A message can also be sent to the console by defining a user ID of WTO (write-to-operator).

  Example:

  ```
  CALL $NOTIFY userid 'message text'
  ```

  $SYSVIEWE_INIT

  Specifies an internal IMOD that is executed during GSS initialization.

  $SYSVIEWE_TERM

  Specifies an internal IMOD that is executed during GSS termination.

  PARMS_CLOCKTIME

  Specifies a parameter member used by the IMOD SCM_CLOCKTIME. Do not compile this member.

  PARMS_DEVICES

  Specifies a parameter member used by the IMODs SCM_NODES and SCM_DASD. Do not compile this member.

  SCM

  Specifies the System Condition Monitor Command Interface.

  Example:

  ```
  SCM [START|STOP|RESTART]
  ```

- You can use the following sample IMODs to create detailed entry information. For more information about the summary and detailed entry information, see the *sysview*.CNM4BPRM member SCM.

  SCM_ACTIVITY

  Displays excessive resource usage

  SCM_ASID

  Provides SCM detail set ASID

  SCM_CALERTS

  Provides CICS exception alerts

  SCM_CDB2CONN

  Provides CICS DB2 connections

  SCM_CENQUEUE

  Provides CICS enqueue conflict monitoring

  SCM_CHANPATH

  Provides channel path status

  SCM_CICS

  Sets the target address space to a CICS jobname

  SCM_CICSALERTS

  Provides CICS exception alerts

  SCM_CICSINACTIVE

  Provides CICS inactive status

  SCM_CICSPLEX

  Provides CICSPLEX exception alerts

  SCM_CMQCONN

  Provides CICS MQ connection monitoring

  SCM_COMMON

  Provides Common storage usage

  SCM_CONFLICTS

  Provides Enqueue conflicts

  SCM_CREMOTE

  Provides CICS remote connections status

SCM_CSTATUS

Provides CICS exception alerts

SCM_CTDATA

Provides CICS transient data queues monitoring

SCM_CUOW

Provides CICS units of work monitoring

SCM_DASD

Provides DASD devices

SCM_DCLOG

Provides Datacom logging and status monitoring

SCM_DCSYSTEM

Provides Datacom system statistics

SCM_ECONFLICTS

Provides enhanced enqueue conflicts monitoring

SCM_IMSALERTS

Provides IMS exception alerts

SCM_JOBSUM

Provides JES2 job summary

SCM_JOBTYPE

Displays the job type

SCM_JRESOURC

Provides JES2 resources

SCM_LISTCONS

Provides information about your consoles

SCM_LISTINP

Processes jobs on input queues

SCM_MIATAPES

Processes tape mounts pending from CA MIA

SCM_MQLIST

Processes MQ queue managers

SCM_MQSALERTS

Processes MQ exception alerts

SCM_MVSALERTS

Processes MVS exception alerts

SCM_NODES

Processes JES2 nodes

SCM_PAGEDS

Processes page data sets and auxiliary storage

SCM_PRINTERS

Processes printers

SCM_SMF

Processes SMF data sets

SCM_SPACE

Displays DASD space

SCM_SPOOLS

Displays JES spool volumes

SCM_STORGRP

Displays SMS storage groups

SCM_TAPE

Displays tape devices

SCM_TCPALERTS

Displays TCP/IP exception alerts

SCM_UFILESYS

Displays USS mounted file system

SCM_WMSYSSUM

Displays workload manager system summary

SCM_WTOR

Displays WTOR messages that require replies

SCM_XSLIST

Displays cross system connections

Chapter 16: Using the Logger

The Logger services component can have some CA SYSVIEW components do the following:

1. Record information pertaining to system and subsystem statuses and events
2. Retain that information, perhaps for years

The Logger services component accomplishes this recording and retention by providing a consistent way to define, create, and manage log streams.

This section contains the following topics:

LOGR Couple Data Set

The LOGR Couple data set performs the following functions:

- Records all log stream definitions
- Tracks other resources that are in use by these log streams, such as offload data sets

The installation must consider both of the following:

- The number of log stream definitions
- The number of offload data sets

Both can concurrently exist at the time the IXCL1DSU utility formats the LOGR Couple data set.

A defined log stream definition record within the LOGR Couple data set can track 168 log stream offload data sets. Create a directory extent record pool by specifying the DSEXTENT parameter if *any* log stream consists of more than 168 offload data sets.

The System Logger allocates records from the DSEXTENT pool as needed when any log stream consists of more than 168 offload data sets. After deleting all log stream offload data sets in a directory extent, the System Logger returns the directory extent record to the available pool of directory extents for the sysplex.

Log Stream Types

The logger component supports two types of log streams:

- Coupling Facility based
- DASD only

This table highlights the main differences between Coupling Facility based and DASD only log streams:

Characteristic	Coupling Facility Based	DASD Only
Interim storage	Coupling Facility List Structure	Data space
Multi-system sharing	Shared by multiple connectors executing on multiple systems simultaneously.	Shared by multiple connectors executing on a single system simultaneously.
Multi-system access	Accessible to all systems in the sysplex.	Accessible to all systems in the sysplex, but, only one system can connect to the log stream at any point in time.

Note: Even though CA SYSVIEW should not need to create Coupling Facility based log streams, the Logger component is designed to support both log stream types.

Log Stream Names

The log stream name identifier is composed of a series of one- to eight-character qualifiers, separated from one another by dot characters. The total length of this identifier must be less than or equal to 26 characters. Match the log stream name to your installation data set naming conventions, with a user ID or system logger application name as the first qualifier. This convention simplifies reporting, defining SAF generic profiles, and the coding of SMS ACS routines.

Log Stream Definition

The following JCL stream illustrates the definition of a DASD only log stream where:

- Each offload data set is sized at 12800 4-KB blocks
- The staging data set is sized at 25 MB

None of the SMS class names are explicitly specified (for example, the LS_STORCLAS parameter) in the JCL. For this reason, the rules embodied within the installation SMS ACS routines perform the class assignments.

```
//DEFINE  EXEC  PGM=IXCMIAPU,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
    DATA TYPE(LOGR) REPORT(NO)

    DEFINE LOGSTREAM
              NAME(SYSVIEW.SAMP.DASDLOG)
              AUTODELETE(YES)
              DASDONLY(YES)
              DESCRIPTION(TEST)
              DIAG(NO)
              EHLQ(LOGGER)
              HIGHOFFLOAD(70)
              LOWOFFLOAD(0)
              LS_SIZE(12800)
              MAXBUFSIZE(32767)
              MODEL(NO)
              OFFLOADRECALL(YES)
              RETPD(7)
              STG_SIZE(6400)

    LIST LOGSTREAM
              NAME(SYSVIEW.SAMP.DASDLOG)
              DETAIL(YES)
/*
```

For more information, see the IBM guide *z/OS MVS Setting Up a Sysplex* (SA22-7625).

Staging and Offload Data Sets

Staging and offload data sets are VSAM Linear data sets. They must be defined with Share Options (3,3) and should be SMS Managed. Staging and Offload data set names all begin with an extended high-level qualifier (EHLQ) which has a maximum length of 33 characters. If the installation does not specify an EHLQ, the system uses the string IXGLOGR.

Offload Data Set Names

To form the name of an offload data set, the MVS System Logger:

1. Appends the log stream name to the EHLQ
2. Appends a system generated sequence number

Note: The length of the data set name string must not exceed the VSAM imposed limit of 44 characters.

Staging Data Set Names

You can define staging data set names for DASD only and Coupling Facility based log streams. The resulting data set names for both DASD only and Coupling Facility based log streams:

- Must be unique within the sysplex
- Must not exceed the 44-character limit

Coupling Facility

To form the staging data set name for a Coupling Facility based log stream, the MVS System Logger concatenates the EHLQ, the log stream name, and the MVS system name derived from the IEASYSxx PARMLIB member, in that order.

If the system name begins with a decimal digit, that portion of the data set name qualifier is the string STG plus the last five significant characters of the system name.

DASD Only

To form the staging data set name for a DASD only log stream, the MVS System Logger concatenates the EHLQ, the log stream name, and the sysplex name derived from the COUPLExx PARMLIB member, in that order.

If the sysplex name begins with a decimal digit, that portion of the data set name qualifier is the string STG plus the last five significant characters of the sysplex name.

Staging Data Set Name Rules

The following examples demonstrate how these naming rules operate.

Coupling Facility

Log Stream Name: SMF.CFLOG

Log Stream Type: Coupling Facility based

EHLQ: SYSVIEW.TEST

System Name: MVSA

Sysplex Name: PRODPLEX

Resultant Staging Data Set Name: SYSVIEW.TEST.SMF.CFLOG.MVSA

Resultant Offload Data Set Name: SYSVIEW.TEST.SMF.CFLOG.A0000001

DASD only

Log Stream Name: SMF.DASDLOG

Log Stream Type: DASD only

EHLQ: SYSVIEW.TEST

System Name: MVSA

Sysplex Name: PRODPLEX

Resultant Staging Data Set Name: SYSVIEW.TEST.SMF.DASDLOG.PRODPLEX

Resultant Offload Data Set Name: SYSVIEW.TEST.SMF.DASDLOG.A0000001

DASD only with EHLQ set to null

Log Stream Name: SMF.DASDLOG

Log Stream Type: DASD only

EHLQ: Null

System Name: MVSA

Sysplex Name: 1PLEX

Resultant Staging Data Set Name: IXGLOGR.SMF.DASDLOG.STG1PLEX

Resultant Offload Data Set Name: IXGLOGR.SMF.DASDLOG.A0000001

Process and View Log Streams

CA SYSVIEW uses MVS log streams to store data collection information. This information is written to a log stream in the form of an SMF record. The primary purpose for using MVS log streams is to allow easy and quick viewing of historical data.

All data that is written to the MVS log streams can also be optionally written to the MVS SMF log. See online help topic: SMF records created by CA SYSVIEW.

Follow these steps:

1. View all log streams used by CA SYSVIEW using a corresponding online display command as follows:

   AUDITLOG

   Displays SYSVIEW audit event records.

   CSYSDATA

   Displays CICS system data interval records

   CTRANSUM

   Displays CICS transaction summary records

   CTRANLOG

   Displays CICS transaction detail records

   IMSRSLOG

   Displays IMS region summary records

   IMSTLOG

   Displays IMS transaction data records

   MQRLOG

   Displays MQ Application Request records

   PLOTLOG

   Displays Plot records

   SMFLOG

   Displays SMF records recorded by the Event Capture Option

   XLOG

   Displays CICS threshold and exception records

2. Issue the LGLOGS command

   The list of current CA SYSVIEW log streams displays.

3. Issue the LGSTREAM command

   The list of all log streams defined to MVS displays.

LOGR Exit Setup (Required)

CA SYSVIEW provides a LOGR subsystem exit so that you can use any MVS log stream as an input file to a batch program.

To set up the LOGR exit

The LOGR subsystem exit *must* reside in a linklist data set.

- If you defined the sysview.CNM4BLOD data set to the linklist, no other steps are required.
- If you have *not* defined the sysview.CNM4BLOD data set to the linklist, then you *must* copy the following modules to an existing linklist data set.

  ```
  GSVXLGEX
  GSVXLGXG (alias of GSVXLGEX)
  ```

Log Streams Used as Input to a Batch Program

You can use any MVS log stream as an input file to a batch program. However, the batch program needs the assistance of a log stream subsystem exit.

Review these log stream subsystems exits:

- IXGSEXIT is the generic log stream subsystem exit provided by IBM. The exit program must be located in a linklist data set.

  Example IXGSEXIT exit:

  ```
  LOGFILE DD DISP=SHR,DSN=log.stream.name,
  SUBSYS=(LOGR,IXGSEXIT,'SUBSYS-opts1','SUBSYS-opts2'),
  DCB=BLKSIZE=nnnnn
  ```

  This exit program receives control at these event points:

  - Converter
  - Allocation
  - Open
  - Get
  - Close
  - Unallocation

- GSVXLGEX is the CA SYSVIEW specific exit program that processes and returns a record that any program can use. Because the data stored in CA SYSVIEW log streams is in a compressed format, you must use this exit to process the records.

  Example GSVXLGEX exit:

  ```
  LOGFILE DD DISP=SHR,DSN=log.stream.name,
  SUBSYS=(LOGR,GSVXLGEX,'SUBSYS-opts1','SUBSYS-opts2'),
  DCB=BLKSIZE=32760
  ```

SUBSYS Statement—Specify Parameters

You can specify parameters on the SUBSYS statement of your CA SYSVIEW log stream using the following syntax:

```
//ddname DD DISP=SHR,DSN=log.stream.name,
// SUBSYS=(LOGR,exitname,'SUBSYS-opts1','SUBSYS-opts2')
//          DCB=BLKSIZE=nnnnn
```

exitname

Name of the LOGR exit module. This module *must* reside in a linklist data set.

- IXGSEXIT - IBM supplied exit
- GSVXLGEX - CA SYSVIEW exit

SUBSYS-*opts1*

Available opts1 parameters:

```
{ FROM=(yyyy/ddd,hh:mm:ss) |OLDEST   }
{,TO=(yyyy/ddd,hh:mm:ss)   |YOUNGEST }
{,DURATION=(nnnn,HOURS)              }
{,GMT| LOCAL                         }
```

FROM=

Indicates the start time of the log stream to process. The first block is the one with a time stamp later than or equal to the specified time.

OLDEST - Indicates the first block read is the oldest block on the log stream.

yyyy/*ddd* - Specifies the start date

hh:*mm*:*ss* - Specifies the start time

The FROM keyword is mutually exclusive with the DURATION keyword.

Default: OLDEST

TO=

Indicates the ending time of the log stream to process. The last block is the one with a time stamp earlier than or equal to the specified time.

YOUNGEST - Indicates the last block read is the youngest block on the log stream.

yyyy/*ddd* - Specifies the end date

hh:*mm*:*ss* - Specifies the end time

The FROM keyword is mutually exclusive with the DURATION keyword.

Default: YOUNGEST

DURATION=

Specifies which blocks to process.

Syntax: DURATION=(*nnnn*,HOURS)

Requests the last *nnnn* hours up to the youngest block be processed. The "last *nnnn* hours" is calculated based on the allocation time of the DD.

The first block is the one with a time stamp greater than or equal to the calculated start time. The last block read is the youngest block on the log stream at the time of allocation.

The DURATION keyword is mutually exclusive with the TO and the FROM keywords.

GMT | LOCAL

Specifies whether the time is local time (based on the time zone offset at the time the log was written) or GMT time.

Default: GMT

SUBSYS-*opts2*

Available *opts2* parameters:

FORWARD OLDTOYOUNG|BACKWARD YOUNGTOOLD

Specify FORWARD to read records from oldest to youngest.

Specify BACKWARD to read records from youngest to oldest.

Default: FORWARD

CNT=*nnnnnnnn*

Limits the number of records read.

Default: 0, no limit.

NOMSGS

Displays no messages, including error messages.

STATS|NOSTATS

Displays a statistics report.

Default: STATS

EXPAND|NOEXPAND

Expands compressed log records.

Default: EXPAND

TRACE|NOTRACE

Displays event trace points. No message is logged for the Converter event.

Default: NOTRACE

WARN|NOWARN

Displays the return/reason code message for warning results.

Default: NOWARN

TRUNCMSG|NOTRUNCMSG

Displays a message when an output record is truncated.

Default: NOTRUNCMSG

DELETE|NODELETE

Deletes blocks older than the oldest block read.

RDW|NORDW

Returns a record including RDW.

Do not include RDW, bump by 4 bytes.

Default: RDW

Note: If conflicting options are coded, the result of the last specified option is used.

Example: SUBSYS Syntax

The following examples show the SUBSYS statement syntax with optional parameters included:

```
//LOGFILE  DD DISP=SHR,DSN=log.stream.name,
//         SUBSYS=(LOGR,GSVXLGEX,
//         'FROM=OLDEST,TO=YOUNGEST,GMT',
//         'STATS,FORWARD,CNT=1000')

//LOGFILE  DD DISP=SHR,DSN=log.stream.name,
//         SUBSYS=(LOGR,GSVXLGEX,
//         'FROM=(2005/001,00:00:00),TO=(2005/365,23:59:59),GMT',
//         'STATS,FORWARD')

//LOGFILE  DD DISP=SHR,DSN=log.stream.name,
//         SUBSYS=(LOGR,GSVXLGEX,,'BACKWARD,CNT=1000')
```

The following example is suggested if a log is being used as input to a CA Easytrieve report:

```
//SMFIN    DD DISP=SHR,DSN=log.stream.name,
//         SUBSYS=(LOGR,GSVXLGEX,
//         'FROM=OLDEST,TO=YOUNGEST',
//         'STATS,FORWARD,NORDW'),
//          DCB=(DSORG=PS,RECFM=VB,LRECL=32756,BLKSIZE=32760)
```

Size the Log Streams

A log stream consists of staging data sets and offload data sets. Size these two pieces individually but with respect to each other.

When sizing log streams, determine the following:

- How much data to keep for review
- The sizes required for your staging and offload data sets

Follow these steps:

1. Use the IBM report utility IXGRPT1.

   This utility provides information concerning how much data is being moved to the offload data sets within an SMF interval.

2. Once you know the average amount of data being moved within an interval, compute the total size required based on the number of intervals needed.

   The number of intervals you needed becomes your retention period.

Sizing Table for the Offload Data Sets

The data contained within each log stream can be different. The differences include average record size and the rate at which data is written to the log.

The following is a sample sizing table:

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4 KB Blocks	Cyls
14	194	100	6,518,400	14	114	1
30	194	100	13,968,000	10	341	2
30	194	1000	139,680,000	10	3,410	10

RetPd Days

Specifies the retention period is the number-of-days worth of data to maintain within the log stream.

Formula variable: RetPd

Avg Rec Length

Specifies the average length of a record as it exists within the physical log stream. The records are maintained in a compressed format with the log stream.

Formula variable: Size

Hourly Count

Specifies the average number of records written to the log in a period of one hour.

Formula variable: Count

Total Byte Requirement

Specifies the total amount of storage in bytes required to maintain the desired amount of data.

Formula variable: Bytes

Example: Bytes = Size * (Count * 24) * RetPd

Max Dsns

Specifies the maximum number of offload data sets that you want to be used to maintain the desired amount of data.

The System Logger allocates up to 168 offload data sets. No parameter exists that limits the number of offload data sets created. Therefore, use the number of offload data sets in your equation to help determine the size of a single offload data set. The more offload data sets that are used reduces the size requirement of a single offload data set. This equation is a balancing act. Offload data sets are allocated on a DASD device as a single extent. If the size is too large, you have trouble finding free space.

For log streams with a high volume of data, size the offload data set to maintain data for a single day or less in one offload data set.

Formula variable: Dsns

4K Blocks

Specifies the number of 4-KB blocks to allocate for a single offload data set.

The System Logger allocates the offload data sets based on the number of 4-KB blocks specified through the LS_SIZE parameter on the log stream definition. Select a size capable of holding at least 10 offloads worth of data. This size is dependent on the volume of data being written to the log stream.

Formula variable: LS_SIZE

Example: LS_SIZE = (Bytes / 4096) / Dsns

Cyls

Specifies the number of cylinders on a 3390 DASD device required to allocate the offload data set based on the specified number of 4-KB blocks.

The System Logger does not use this value. This number is here as a reference point. The number of cylinders represented is for one offload data set.

Formula variable: Cyls

Example: Cyls = (LS_SIZE * 4096) / 712140

Sizing Log Streams for CICS Threshold and Exception Records

This sample table sizes log streams for CICS threshold and exception records:

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4-KB Blocks	Cyls
7	250	250	10,500,000	10	256	2
14	250	100	8,400,000	14	146	1
14	250	300	25,200,000	14	439	3

Consider the following:

- Hourly Count

  This count equates to:

  - The average number of CICS thresholds triggered
  - The number of CICS exceptions that occur per hour

  If the data from multiple CICS regions get logged to the log stream, then this value must be the combined total for those CICS regions.

- The volume of this log stream should be low.

  The large amount of data can be maintained.

Sizing Log Streams for CICS System Data Interval Records

This sample table sizes log streams for CICS system data interval records:

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4-KB Blocks	Cyls
30	550	80	31,680,000	30	258	2
60	550	120	95,040,000	60	387	2

Consider the following:

- Hourly Count

  Each CICS data collector writes a System Interval record at the conclusion of its system data collection interval. By default, the interval is 15 minutes. This means that each CICS writes four (4) records per hour.

  If the data from multiple CICS regions get logged to the log stream, then this value must be the combined total for those CICS regions.

- The above example assumes:
  - The system interval is 15 minutes
  - That 10 CICS regions are being logged to the log stream
- The volume of this log stream should be low.

  The large amount of data can be maintained.

Sizing Log Streams for CICS Transaction Summary Records

This sample table sizes log streams for CICS transaction summary records:

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4-KB Blocks	Cyls
7	150	4000	100,800,000	10	2,461	12
14	150	4000	201,600,000	14	3,516	17

Consider the following:

- Hourly Count

  Each CICS data collector writes a CICS Transaction Summary record for each unique transaction ID that executed during the system data collection interval.

  If the data from multiple CICS regions get logged to the log stream, then this value must be the combined total for those CICS regions.
- The above example assumes that the system interval is 15 minutes.

  On average we are assuming that 100 unique transaction IDs execute during each interval.

  It also assumes that 10 CICS regions are being logged to the log stream.

  Hourly count = 100 * 4 * 10
- The volume of this log stream should be low to medium.

  The large amount of data can be maintained.

Sizing Log Streams for CICS Transaction Detail Records

This sample table sizes log streams for CICS transaction detail records:

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4-KB Blocks	Cyls
5	600	10000	720,000,000	100	1,758	9
5	600	50000	3,600,000,000	100	8,789	43

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4-KB Blocks	Cyls
5	600	100000	7,200,000,000	120	14,648	71

Consider the following:

- Hourly Count

  This count equates to the average number of CICS transactions that execute per hour. If the average daily total is known, divide the average daily total by 24 to obtain the hourly value.

  If the data from multiple CICS regions get logged to the log stream, then this value must be the combined total for those CICS regions.

- The CICS Transaction log stream usually contains a large volume of data and gets written to at a high rate.
- Size the offload data sets so each data set contains data for one day or less.

Sizing Log Streams for IMS Transaction Data Records

This sample table sizes log streams for IMS transaction data records:

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4-KB Blocks	Cyls
5	450	10000	540,000,000	100	1,318	7
5	450	50000	2,700,000,000	100	6,592	32
5	450	100000	5,400,000,000	120	10,986	53

Consider the following:

- Hourly Count

  This count equates to the average number of IMS transactions that execute per hour. If the average daily total is known, divide the average daily total by 24 to obtain the hourly value.

  If the data from multiple IMS control regions gets logged to the log stream, then this value must be the combined total for those IMS regions.

- The IMS Transaction log stream usually contains a large volume of data and gets written to at a high rate.
- Size the offload data sets so each data set contains one day or less worth of data.

Sizing Log Streams for IMS Summary Data Records

This sample table sizes log streams for IMS summary data records:

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4-KB Blocks	Cyls
5	520	10000	624,000,000	100	1,523	8
5	520	50000	3,120,000,000	100	7,617	37
5	520	100000	6,240,000,000	120	12,695	62

Consider the following:

- Hourly Count

  This count equates to the average number of IMS region termination events that execute per hour. The region summary region is written when an IMS region terminates.

  This figure applies to message, BMP and DBT region types. If the average daily total is known, divide the average daily total by 24 to obtain the hourly value. If the data from multiple IMS control regions get logged to the log stream, then this value must be the combined total for those IMS regions.

- The IMS Region Summary log stream usually contains a large volume of data and gets written to at a high rate.
- Size the offload data sets so each data set contains one-day or less worth of data.

Sizing Log Streams for MQ Application Request Records

This sample table sizes log streams for MQ Application Request records:

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4-KB Blocks	Cyls
5	300	800	28,800,000	80	88	1
14	300	800	80,640,000	100	197	1
14	300	8000	806,400,000	100	1,969	10

Consider the following:

- Hourly Count

  The WebSphere MQ data collector writes a record for each queue used during the system data collection interval.

  If multiple Queue Managers are being monitored, this value must be the combined total for those Queue Managers.

  The above example assumes that the system interval is 15 minutes.

  - Examples 1 and 2 assume that 100 queues were used and two Queue Managers were being monitored.

    Hourly count = 100 * 4 * 2

  - Example 3 assumes that 1000 queues were used and two Queue Managers were being monitored.

    Hourly count = 1000 * 4 * 2

- The volume of this log stream should be medium.

  The large amount of data can be maintained.

Sizing Log Streams for Plot Records

This sample table sizes log streams for Plot records:

RetPd Days	Avg Rec Length	Hourly Count	Total Byte Requirement	Max Dsns	4-KB Blocks	Cyls
7	450	500	37,800,200	7	1318	7
14	450	500	75,600,000	14	1318	7
14	450	500	75,600,000	100	185	1

Consider the following:

- The volume of this log stream should be low.

  The large amount of data can be maintained.

Sizing Log Streams for SMF Records

Consider the following:

- The amount of SMF records that can be maintained in the log when dependent on the type of SMF records recorded. Each type of SMF record can vary widely in size.
- The SMF log stream usually contains a large volume of data and gets written to at a high rate.
- Size the offload data sets so each data set contains one day or less worth of data.

How to Produce a Report

After you acquire the statistics from the contents of a log stream, you can produce a report.

Use the following utilities to produce a report:

- The IBM report utility IXGRPT1

  IXGRPT1 provides information concerning how much data is being moved to the offload data sets within an SMF interval. This utility creates a report using SMF 88 records.

- The IEBGENER utility

  IEBGENER dumps the contents of a log stream to a sequential data set. No data is actually copied because the output is going to DUMMY. The exit program produces information about the log stream being read.

The following is a sample IEBGENER job:

```
//jobcard
//*
//GENER1   EXEC PGM=IEBGENER
//*
//SYSUT1   DD DISP=SHR,DSN=log.stream.name,
//         SUBSYS=(LOGR,GSVXLGEX),
//         DCB=BLKSIZE=32760
//*
//SYSPRINT DD SYSOUT=*
//SYSUT2   DD DUMMY
//SYSIN    DD DUMMY
/*
```

Sample output from the IEBGENER job:

```
(01) SYSUT1   Open     Log Stream TEST.SMFDATA.LOG
(02) SYSUT1   Options  GSVXLGEX
     SYSUT1   Options  FROM=OLDEST,TO=YOUNGEST,LOCAL
     SYSUT1   Options  FORWARD
(03) SYSUT1   Oldest   record read     yyyy/07/27 07:14:00.784201 Local
     SYSUT1   Youngest record read     yyyy/07/28 08:12:45.089467 Local
(04) SYSUT1   Oldest   block ID read   00000007254DFCBA
     SYSUT1   Youngest block ID read   0000000736E1FA43
     DDname   Record Type                   Count
     -------- ----------------------- ----------
(05) SYSUT1   CICS System Data               344
     SYSUT1   CICS Threshold                   3
     SYSUT1   CICS Transaction Detail        114
     SYSUT1   CICS Transaction Summary       114
     SYSUT1   IMS Transaction              62693
     SYSUT1   Plot Metric                   3508
     SYSUT1   WebSphere MQ                  1610
     SYSUT1   Other                      160985
     -------- ----------------------- ----------
(06) SYSUT1   Records returned           229371
     SYSUT1   Log records expanded       170604
     SYSUT1   SMF records expanded        57303
     SYSUT1   Average compression           46%
     SYSUT1   Max input record length     23674
     SYSUT1   Max output record length    32738
     SYSUT1   Avg input record length       1246
     SYSUT1   Avg output record length      2725
     SYSUT1   Avg expanded length           2725
     SYSUT1   Avg non-expanded length         94
     SYSUT1   CPU time used in exit       12.841
```

- (01) - <ddname> Open Log Stream <logstream>

  Indicates the specified log stream was opened.

- (02) - <ddname> Options <options>

  Defines the list of options used on the SUBSYS= parameter.

- (03) - <ddname> <message> <date> <time> <GMT|Local>

  Displays the date and time of the oldest and youngest records read.

- (04) - <ddname> Oldest block ID read 00000007254DFCBA

  Displays the block ID of the oldest and youngest records read.

- (05) - <ddname> <record type> <count>

  Displays the number of records read of a specific SMF type. SMF records created by CA SYSVIEW get listed specifically. All others get listed as "Other".

- (06) - <ddname> <statistics type> <value>

  Statistics Type:

  - Records returned - Total number of records returned to the caller.
  - Log records expanded - Number of records that were compressed and needed expanded.
  - SMF records expanded - The SMF records written by CA SYSVIEW can be compressed to save space in the SMF data sets or log. The number of records that were compressed.
  - Average compression - The average compress rate of compressed records.
  - Max input record length - The maximum input length of the records read.
  - Max output record length - The maximum output length of the records written. This value is the length after any expansion has been done.
  - Avg input record length - The average input length of the records read.
  - Avg output record length - The average output length of the records written. This value is the length after any expansion has been done.
  - Avg expanded length - The maximum output length of the records written that required expansion.
  - Avg non-expanded length - The maximum output length of the records written that did not require expansion.
  - CPU time used in exit - The amount of CPU used by the exit program.

Chapter 17: Usage of SMF Records

By default CA SYSVIEW creates SMF records as type 255.

Using online display commands you can view:

- CA SYSVIEW created SMF records
- Any SMF record

This section contains the following topics:

View Product Created SMF Records

CA SYSVIEW can optionally create SMF records that you can view using online commands.

The following table shows the relationship between the SMF records created by CA SYSVIEW and the corresponding online display commands.

Subtype	Online Command	Description
001 x01	AUDITLOG	Audit records
002 x02	PLOTLOG	Plot records
003 x03	XLOG	Threshold records
004 x04	XLOG	State Exception records
023 x17	CSYSDATA	CICS system data interval records
024 x18	XLOG	CICS exception records
025 x19	CTRANSUM	CICS transaction summary records
027 x1B	CTRANLOG	CICS transaction detail records
028 x1C	CSYSDATA	CICS system interval data records
032 x20	IMSTLOG	IMS transaction data records
033 x21	IMSRSLOG	IMS region summary records
048 x30	MQRLOG	MQ application request records

SMF Record Layouts

The CA SYSVIEW created SMF record layouts or DSECTs are distributed in the *sysview*.CNM4BMAC data set. The following table describes the SMF type 255 subtype records and shows the macro library in which they are distributed:

Subtype	MACLIB	Description
001 x01	ZSMFADTR	Audit records
002 x02	ZSMFPLOT	Plot records
003 x03	ZSMF003	Threshold Exception records
004 x04	ZSMF004	State Exception records
024 x18	CSMF024	CICS exception records
025 x19	CSMF025	CICS transaction summary records
027 x1B	CSMF027	CICS transaction detail records
028 x1C	CSMF028	CICS system interval data records
034 x22	IMSIMTR	IMS transaction data records
035 x23	IMSIMRA	IMS region summary records
048 x30	ZSMFMQRR	MQ Application Request records

Note: For a list of macros containing CA Easytrieve definitions of SMF record types, see the chapter "Creating Reports with CA Easytrieve Common Reporting Service" in the *User Guide*.

View Any SMF Record

The CA SYSVIEW SMF data collector, which is part of the Event Capture Option, lets you:

- Capture SMF records for online viewing through the SMFLOG command
- Suppress SMF records from the SMF data sets

Issue the SMFLOG command to display the log of captured SMF records. You define the configuration options for the SMF data collector in the SMFDATA parmlib member.

SMF Record Format

Each record type displayed from the SMFLOG command must have a format routine to generate a report. If a format routine is not available, then the SMF record displays in dump format.

Code to format an SMF record is available in the SAMPLIB member SMFF0017.

Report Generation

The address of an SMF record is passed to the report program, which allocates a buffer and generates a report. The address of the buffer is then returned to the calling program.

Configuration Options

A configuration option specifies or controls how each record is written to SMF. You define the configuration option in the parmlib members.

The following table shows the relationship between a subtype of the default CA SYSVIEW type 255 SMF record and the corresponding parmlib member and configuration option:

Subtype	Parmlib Member	Configuration Option
001 x01	AUDITDEF	n/a
002 x02	SYSDATA	LOG-PLOT-RECORDS-SMF
003 x03	SYSDATA CICSLOGR	LOG-XLOG-RECORD-SMF LOG-EXCEPTIONS-SMF
004 x04	SYSDATA CICSLOGR	LOG-XLOG-RECORD-SMF LOG-EXCEPTIONS-SMF
024 x18	CICSLOGR	LOG-EXCEPTIONS-SMF
025 x19	CICSLOGR	LOG-TRANSACTION-SUMMARY-SMF
027 x1B	CICSLOGR	LOG-TRANSACTION-SMF
028 x1C	CICSLOGR	LOG-SYSTEM-DATA-SMF
034 x22	IMSLOGR	LOG-IMSTRAN-RECORDS-TO-SMF
035 x23	IMSLOGR	LOG-IMSREGN-RECORDS-TO-SMF
048 x30	MQSDATA	LOG-REQUESTS-TO-SMF

How to Define Collection Options

This section contains initialization and setup parameters for the collection process.

The following examples define which records to capture or suppress by specifying a specific record type and subtype:

```
LOG-SMF-RECORD-TYPE          255:002:INCLUDE
LOG-SMF-RECORD-TYPE          030:005:INCLUDE
LOG-SMF-RECORD-TYPE          030:006:INCLUDE
LOG-SMF-RECORD-TYPE          070:001:INCLUDE
```

Using the SMFDATA parmlib member, you can instruct SMF to suppress or not record specific SMF record types and subtypes.

To include, exclude, or suppress SMF records by type and subtype, review the following example definitions:

- Log type 30 and all subtypes:

  ```
  LOG-SMF-RECORD-TYPE          030:*:INCLUDE
  ```

- Log type 30 with only subtype 4:

  ```
  LOG-SMF-RECORD-TYPE          030:4:INCLUDE
  ```

- Do not log type 30 subtype 5:

  ```
  LOG-SMF-RECORD-TYPE          030:5:EXCLUDE
  ```

- Suppress SMF type 255 subtype 50:

  ```
  SUPPRESS-SMF-RECORD-TYPE    255:50:INCLUDE
  ```

- Do not suppress any records:

  ```
  SUPPRESS-SMF-RECORD-TYPE    *:*:EXCLUDE
  ```

Default Configuration Values

The following is a list of configuration options and the default values. These options are also provided in the SMFDATA parmlib member.

- **LOG-SMF-RECORD-TYPE**-Defines Records by Type and Subtype

  Specifies the SMF records by type and subtype that are included or excluded in the log.

  Syntax:

  `type:subtype:action`

  type

  000 to 255 or *

  subtype

  000 to 255 or *

  action

  INCLUDE - Log

  EXCLUDE - Do not log

  Default: *:*:EXCLUDE

- **LOGSTREAM-SMFDATA-NAME**-Defines the MVS Logger log stream name

  Specifies the name of the MVS logger log stream used to capture SMF data. If the log stream name is specified as NONE, no logging occurs.

  If a log stream has not been defined, set the value to NONE.

- **LOGSTREAM-SMFDATA-DELETE-ALL**-Defines Logger Delete Options

  Specifies whether the logger is to delete all existing records.

  Default: No

- **MODE-BYPASS**-Defines Dynamic Exit Processing

  Specifies whether all dynamic exit processing is bypassed. In this case, the dynamic exits are in place. No processing is performed on the SMF records.

  Default: No

- **MODE-LOG**-Defines the Processing of Logged SMF Records

  Specifies whether the logging of SMF records extracted through the SMF dynamic exits are processed. This option overrides all specific record type requests.

  Default: Yes

- **MODE-SUPPRESS**-Defines Whether to Suppress SMF Records

  Specifies whether the SMF records are suppressed through the SMF dynamic exits. This option enables the suppression function.

  Default: Yes

- **SUPPRESS-SMF-RECORD-TYPE**-Defines Whether to Suppress the Writing of SMF Records

  Specifies whether the SMF data collector is to suppress the writing of specific SMF records. This option suppresses the records from the SYS1.MANxx data sets.

  Syntax:

  `type:subtype:action`

  type

  000 to 255 or *

  subtype

  000 to 255 or *

  action

  INCLUDE - Suppress

  EXCLUDE - Do not suppress

  Default: *:*:EXCLUDE

SMF Records and Descriptions

The SMFTYPE parmlib member associates a text description with an SMF record type and subtype. If the record is to be formatted by the SMFLOG command, then a record formatting routine can also be associated with the record type and subtype.

Note: Initial descriptions have been extracted from the IBM guide *OS/390 MVS System Management Facilities (SMF)*.

Define Default Keyword Values

Definitions are created using keyword and value parameters. This step lets you define default keyword values.

Follow these steps:

1. Start each definition start with the keyword DEFINE.
2. Code each keyword and value pair on the same line.
3. Assign default values for keywords by coding a SETDEFAULT definition.

Subsequent definitions that do not have all keywords coded use the values specified in the default definition.

SETDEFAULT—Define Default Values

The SETDEFAULT definition can be coded multiple times in this member. The default values assigned are in effect from that point forward.

Format of the data:

```
RESETDefault

SETDEFAULT
  TYPE          000
  SUBType       000
  DESC          ' '
  FORMAT        @
ENDDEFAULT

DEFine
  TYPE          nnn
  SUBType       nnn
  DESC          'enclose description in single quotes'
  FORMAT        module
ENDdefine
```

The following describes each keyword:

DEFine

Begins the definition.

TYPE

Specifies the SMF record type.

The value must be in the range of 0 to 255.

The type and subtype values can also be defined together as 000:000.

If the first character of the type or subtype value in type:subtype is an "x", then the value is interpreted as a hexadecimal value.

SUBType

Specifies the SMF record subtype.

Some but not all SMF records have subtypes. The value must be in the range of 0 to 255.

DESC

Provides the SMF record description.

The description of the record must be enclosed in single quotes. The maximum description length is 48 characters, not including the single quotes.

FORMAT

Specifies the name of the record-formatting module.

Any valid member name can be used.

- * - Code the "*" as a placeholder. If coded, then no format routine is used.
- @ - Code the "@" to use the default formatting routine supplied with the product.

If there is a request to format an SMF record and a format routine is not available, then the record displays in dump format.

ENDDEFINE

(Optional) Ends the definition.

SETDEFAULT

Begins the default values definition.

ENDDEFAULT

Ends the default values definition.

RESETDefault

Resets default definition values back to the following original defaults:

```
TYPE        000
SUBTYPE     000
DESC        ' '
FORMAT      @
```

Notes:

- Records containing an "*" in column 1 or blanks in columns 1:72 are ignored.
- Record format routines are supplied in the following modules for the specified record types and subtypes:

Module	Type	Subtype
GSVVF014	014	*
GSVVF015	015	*
GSVVF017	017	*
GSVVF018	018	*
GSVVF030	030	*
GSVVF042	042	09
GSVVF042	042	20
GSVVF042	042	21
GSVVF042	042	24
GSVVF070	070	*
GSVVF074	074	*
GSVVF080	080	*
GSVVF088	088	*
GSVVF089	089	*
GSVVF090	006	*
GSVVF090	030	*
GSVVF090	031	*
GSVVF097	097	*
GSVVF118	118	*

Module	Type	Subtype
GSVXF001	225	1
GSVZF002	255	2
GSVZF003	255	3
GSVZF004	255	4
GSVYF008	255	08
GSVYF009	255	09
GSVYF017	255	17
GSVYF023	255	23
GSVYF024	255	24
GSVYF025	255	25
GSVYF026	255	26
GSVYF027	255	27
GSVYF028	255	28
GSVPF034	255	34
GSVPF035	255	35
GSVSF048	255	48

- The SAMPLIB member SMFF0017 contains sample source code for writing a record formatting routine.

Sample of the Default Definitions

The following shows a sample of the default definitions. For a complete list of definitions, see the SMFTYPE parmlib.

```
DEF TYPE 000:000 DESC 'IPL'
DEF TYPE 001:000 DESC ''
DEF TYPE 002:000 DESC 'Dump header'
DEF TYPE 003:000 DESC 'Dump trailer'
DEF TYPE 004:000 DESC 'Step termination'
DEF TYPE 005:000 DESC 'Job termination'
DEF TYPE 006:000 DESC 'External writer'
DEF TYPE 007:000 DESC 'Data lost'
DEF TYPE 008:000 DESC 'I/O configuration'
DEF TYPE 009:000 DESC 'VARY device ONLINE'
.
.
DEF TYPE 014:000 DESC 'INPUT data set activity'
DEF TYPE 015:000 DESC 'OUTPUT data set activity'
DEF TYPE 016:000 DESC 'DFSORT statistics'
DEF TYPE 017:000 DESC 'Scratch data set status'
.
.
DEF TYPE 030:000 DESC 'Common address space work'
DEF TYPE 030:001 DESC 'Job start or start of other work unit'
DEF TYPE 030:002 DESC 'Activity since previous interval ended'
DEF TYPE 030:003 DESC 'Activity for the last interval before step term'
DEF TYPE 030:004 DESC 'Step total'
DEF TYPE 030:005 DESC 'Job termination or term of other work unit'
DEF TYPE 030:006 DESC 'System address space'
.
.
DEF TYPE 070:000 DESC 'RMF CPU activity'
DEF TYPE 070:001 DESC 'RMF CPU activity'
DEF TYPE 070:002 DESC 'RMF Crypto activity'
.
.
DEF TYPE 074:000 DESC 'RMF activity of several resources'
DEF TYPE 074:001 DESC 'Device activity'
DEF TYPE 074:002 DESC 'XCF activity'
DEF TYPE 074:003 DESC 'OMVS kernel activity'
DEF TYPE 074:004 DESC 'Coupling facility activity'
DEF TYPE 074:005 DESC 'Cache subsystem device activity'
DEF TYPE 074:006 DESC 'HFS statistics'
.
.
DEF TYPE 080:000 DESC 'RACF processing'
.
.
```

```
DEF TYPE 088:000 DESC 'System logger data'
.
.
DEF TYPE 097:000 DESC 'Foreign enclave resource data'
.
.
DEF TYPE 115:000 DESC 'WebSphere MQ statistics'
DEF TYPE 116:000 DESC 'WebSphere MQ statistics'
.
.
DEF TYPE 118:000 DESC 'TCP/IP statistics'
DEF TYPE 118:001 DESC 'TCP/IP - TCP API initialization'
DEF TYPE 118:002 DESC 'TCP/IP - TCP API termination'
DEF TYPE 118:003 DESC 'TCP/IP - FTP client'
DEF TYPE 118:004 DESC 'TCP/IP - TN3270 client'
DEF TYPE 118:005 DESC 'TCP/IP statistics'
DEF TYPE 118:020 DESC 'TCP/IP - TN3270 server initialization'
DEF TYPE 118:021 DESC 'TCP/IP - TN3270 server termination'
DEF TYPE 118:070 DESC 'TCP/IP - FTP server append subcommand'
DEF TYPE 118:071 DESC 'TCP/IP - FTP server delete subcommand'
DEF TYPE 118:072 DESC 'TCP/IP - FTP server logon failures'
DEF TYPE 118:073 DESC 'TCP/IP - FTP server rename'
DEF TYPE 118:074 DESC 'TCP/IP - FTP server retrieve'
DEF TYPE 118:075 DESC 'TCP/IP - FTP server store'
.
.
DEF TYPE 249:000 DESC 'User record type 249 xF9'
DEF TYPE 250:000 DESC 'User record type 250 xFA'
DEF TYPE 251:000 DESC 'User record type 251 xFB'
DEF TYPE 252:000 DESC 'User record type 252 xFC'
DEF TYPE 253:000 DESC 'User record type 253 xFD'
DEF TYPE 254:000 DESC 'User record type 254 xFE'

DEF TYPE 255:000 DESC 'CA SYSVIEW'

DEF TYPE 255:002 DESC 'CA SYSVIEW Plot records'

                 FORMAT GSVZF002

DEF TYPE 255:003 DESC 'CA SYSVIEW Threshold Exceptions'

                 FORMAT GSVYF003

DEF TYPE 255:004 DESC 'CA SYSVIEW State Exceptions'

                 FORMAT GSVZF004

DEF TYPE 255:017 DESC 'CA SYSVIEW CICS Transaction'

                 FORMAT GSVYF017

DEF TYPE 255:023 DESC 'CA SYSVIEW CICS System Data'
```

```
                  FORMAT GSVYF023
DEF TYPE 255:024 DESC 'CA SYSVIEW CICS Exception'
                  FORMAT GSVYF024
DEF TYPE 255:025 DESC 'CA SYSVIEW CICS Transaction Summary'
                  FORMAT GSVYF025
DEF TYPE 255:026 DESC 'CA SYSVIEW CICS Transaction'
                  FORMAT GSVYF026
DEF TYPE 255:027 DESC 'CA SYSVIEW CICS Transaction'
                  FORMAT GSVYF027
DEF TYPE 255:032 DESC 'CA SYSVIEW IMS Transaction Data'
                  FORMAT GSVPF032
DEF TYPE 255:033 DESC 'CA SYSVIEW IMS Region Summary'
                  FORMAT GSVPF033
DEF TYPE 255:048 DESC 'CA SYSVIEW MQ Requests'
                  FORMAT GSVSF048
)EOF
```

Chapter 18: External Line Commands

The ability to define external line commands lets you define your own line commands for any given display.

This section contains the following topics:

How External Line Commands are Processed

When you define external line commands, understand the actions CA SYSVIEW takes to find the specified line command. This understanding helps ensure CA SYSVIEW locates and processes your line command.

With externally defined line commands active, the following process occurs when you enter a line command:

1. The user enters a line command on a specific data row.
2. The CA SYSVIEW display module invokes the line command subroutine LOCATE to search the following command types in sequence until it locates the specified line command:
   a. Internally defined line commands
   b. Externally defined product, user, and system line commands
   c. Primary commands (limited usage)
3. If no match is found, the following error message displays:

   ```
   GSVX156E Invalid line command - <linecmd>
   ```

Display the Available Line Commands

You can display externally defined line commands as follows:

- To display all externally defined line commands, enter the following command:

  ```
  LISTLCMD
  ```

- To display only the externally defined line commands that are available for the current command, enter the following line command:

  ```
  ??
  ```

  Note: The line command ?? is implemented using external line command definitions.

External Line Command Organization

The external line command definitions are organized into three categories:

- Product definitions

  Location: SYSVIEW.CNM4BPRM(GSVXLCMD)

  This member contains the external line command product definitions, which are internal and shipped with the product. Data in this member should not be altered.

- User definitions

  Location: USER.CNM4BPRM(LINECMDS)

  This member contains external line command definitions that an individual CA SYSVIEW user creates and are specific to that user. This member contains example definitions that are inactive.

- System definitions

  Location: SYSVIEW.CNM4BPRM.(LINECMDS)

  This member contains external line command definitions that the CA SYSVIEW administrator creates. These definitions are site definitions and are available to all users. This member contains example definitions that are inactive.

External Line Command Parts

An external line command entry contains the following parts:

- The external line command definition key
- The OPTIONS keyword
- The command string to be associated with the key

Define External Line Commands

An externally defined line command associates with an existing primary command.

To define external line commands

1. Set default configuration options for external line commands.

   The OPTIONS parmlib member sets the default configurations for all tasks. The OPTIONS parmlib member contains the following configuration option for external line commands:

   ExternalLineCommands

   Specifies whether external line commands defined in the parmlib member LINECMDS are supported.

2. Use the parameter library member LINECMDS to:
   - Externally define line commands
   - Associate the definition with an existing CA SYSVIEW primary command

The external line command is defined.

More information:

Parmlib Member LINECMDS (see page 219)

Parmlib Member LINECMDS

Use the LINECMDS parmlib member to define line commands externally and associate the definition with an existing CA SYSVIEW primary command.

Code the following parameters on the first line of each definition:

- Command
- LineCmd
- Min

The LINECMDS parameter has the following syntax:

```
Command.Screen... Linecmd. Min  Command-String
command.screen    linecmd  min  <OPTS options> CMD string
```

command

Specifies the name of the command to which this definition applies. This value is not case-sensitive.

Maximum Length: 8

screen

Specifies the screen name within a specific command. For most commands, the screen name is left blank. This value is not case-sensitive.

Maximum Length: 8

linecmd

Specifies the external line command name. This value is not case-sensitive.

Maximum Length: 8

min

Specifies the minimum allowable length of the external line command name. The minimum value is 1 and maximum value is 8. Any other value is ignored.

Default: 3

options

Specifies the options to be associated with the entry. To indicate that option keywords are to follow, enter the keyword OPTS before the option keywords.

The following are the valid options keywords:

PAD *pad*

Specifies a string to use during variable substitution when the field contains all blanks.

XSYS

Indicates an XSSYSTEM entry, which is only used for cross-system data lines where the XSSYSTEM name is different from the current system name.

string

Specifies the command string. To indicate that a command string is to follow, enter CMD before the command string.

The command string executes when the line command processes. Start the command string on the first or second line of a definition. If the string requires multiple lines, use a continuation character. Start or continue the command string in any column from 1 to 72.

Maximum Length: 250

When coding command strings, use the following guidelines:

- Separate multiple commands using the command delimiter.
- Continue the command string by coding a continuation character as the last nonblank character on a line.
- Use the hyphen (-) continuation character when you do not want the leading blanks in the next record ignored.
- Use the plus sign (+) continuation character when you want the leading blanks in the next record ignored.
- Do not let the number of continuation records exceed 256 characters. Any excess characters are truncated without any warning.

Examples: Parmlib member LINECMDS

The following example shows sample definitions:

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
*Command. Linecmd. Min  CmdString
 ACTIVITY ENQUEUE    1  CMD ENQUEUE;SELECT RNAME EQ &F_JOBNAME

 ACTIVITY ENQUEUE    1  CMD ENQUEUE +
                        ;SELECT RNAME EQ &F_JOBNAME

 ACTIVITY ASID       8  CMD ASID &F_JOBNAME

 ACTIVITY ASID       8  OPTS XSYS CMD ACTIVITY;ASID &F_JOBNAME +
                        FIND &F_JOBNAME JOBNAME

 ACTIVITY CANCEL     1  CMD CONFIRM Cancel job: &F_JOBNAME +
                        ;ASCANCEL &F_JOBNAME +
                        NODUMP NOCONFIRM NOMSG
```

Code and Display Data or Variable Substitution

The command string can contain variables or symbolics. You can code and display data or variable substitution. Dynamic variable substitution occurs at runtime.

Follow these steps:

1. Code any CA SYSVIEW symbolic value as part of the command string.
2. Insert fields from the selected data row at runtime using specially named variables.

   If a field or column with the name “columnheader” is found on the current screen, the data value for that column or field from the selected row is used to replace the variable.

   Field name coded for substitution:

   ```
   &F_columnheader
   ```

   ```
   &FLD_columnheader
   ```
3. Issue the SYSVIEW VLIST command to view the variables.

   Your variables are displayed for your review.

Request Command Confirmation

For certain commands, you can require command confirmation before the command is executed.

To request command confirmation, enter the following command as the first command within the command string:

```
CONFIRM
```

When the command initiates:

- A positive confirmation is returned and the remaining portion of the command string is executed.
- Otherwise, the remaining portion of the command string is discarded.

Drill Down to the Next Command Level

When using a line command, you can link or drill down to the next command level.

By default, the first command in the command string is linked to and the associated screen displays, unless the command is a function command.

Function commands:

- Do not have associated screen displays
- Some display messages
- Some cause other screen activity to occur

Example function command: ASID

When the first command in a command string is a function command, such as ASID, insert a LINK command in front of the next primary command.

Create Dynamic Command String Using REXX

An external line command definition can execute a REXX EXEC to create dynamically a command string to execute. The information contained in the data fields is passed to the REXX EXEC. The REXX EXEC uses this data to determine the nature of the command string to return.

Use the following syntax to create command string using a REXX EXEC:

Syntax:

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
*Command. Linecmd. Min  CmdString
 command  linecmd  min  CMD REXX execname parameters
```

Examples:

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
*Command. Linecmd. Min  CmdString.
 ACTIVITY TEST       8  CMD REXX ACTTEST &F_JOBNAME &F_TYPE
```

Only one REXX invocation is allowed per definition, and it must be the first command string.

The REXX EXEC must return a string to execute. The string can contain multiple commands separated by the command delimiter.

The returned result replaces the calling command string:

```
REXX execname parameters
```

Chapter 19: Parmlib Members

This section provides a high-level overview of the CA SYSVIEW parameter library (parmlib) members that affect the operation of the:

- Base product
- Its facilities
- Its optional interfaces to other operating systems or software

This section contains the following topics:

Parameter Library Members Overview

The parmlib members contain information and settings that let you tailor the operation of CA SYSVIEW to your requirements.

CA SYSVIEW creates a set of product variables and system variables that can be referenced in parmlib members. The system variables are created using the system symbols you have defined. The set of product variables that are available are listed in the appendix "Product Variables (see page 519)."

Detailed information about the syntax of the members and how to code the individual entries is found in the members themselves. For a list of keywords used to modify each parmlib member, see the appendix "Parameter Library Keywords (see page 513)."

The following explains the CA SYSVIEW support of the parmlib data sets:

1. CA SYSVIEW supports the following types of parmlib data sets:
   - System parmlib data set
   - User parmlib data set
2. The system parmlib data set is defined in the System Configuration Options member using the Dsn-System-PARMLIB keyword.
3. When a parmlib member is eligible to be read from the user parmlib data set and it is found there, it is read from that data set.
4. When a parmlib member is eligible to be read from the user parmlib data set and it is not found there, or if a parmlib member is not eligible to be read from the user parmlib data set, it is read from the system parmlib data set.

Initialization and Termination Members (Parmlib Members)

STARTCMD

Use the STARTCMD parmlib member to start automatically additional jobs after CA SYSVIEW has been initialized.

The STARTCMD parmlib member:

- Contains a list of z/OS operator commands
- Issues the commands immediately after the CA SYSVIEW main and user interface address spaces complete initialization

STARTCMD Summary

This member is read...

During the initialization of both the main and user interface address spaces

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is read when the address spaces are restarted.

Subtasks that use this data

CA SYSVIEW main and user interface address spaces

Commands that use this data

None

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

STARTMSG

The STARTMSG parmlib member contains message text that is displayed on the system console during initialization of the CA SYSVIEW main and user interface address spaces.

STARTMSG Summary

This member is read...

During the initialization of both the main and user interface address spaces

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is read when the address spaces are restarted.

Subtasks that use this data

CA SYSVIEW main and user interface address spaces

Commands that use this data

None

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

STATSMSG

The STATSMSG parmlib member contains message status that is displayed on the system console before the response to the following command:

```
MODIFY sysview,STATUS
```

STARTMSG Summary

This member is read...

The member is read each time you enter the following command:

```
MODIFY sysview,STATUS
```

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

None

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

STOPCMD

The STOPCMD parmlib member contains a list of z/OS operator commands. These commands are issued at the start of termination for CA SYSVIEW main and user interface address spaces.

STOPCMD Summary

This member is read...

During the initialization of both the main and user interface address spaces

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is read each time the address spaces are restarted.

Subtasks that use this data

CA SYSVIEW main and user interface address spaces

Commands that use this data

None

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

SYSVAUX

The SYSVAUX parmlib member provides the CA SYSVIEW auxiliary address space with initialization information. The definitions that are contained in the member determine which additional components and subtasks are started or enabled. The MEM= parameter contained on the SYSVAUX started task procedure specifies the parmlib member to use for initialization information. The SYSVAUX parmlib member is the default member that is specified.

Note: SYSVAUX is the default name of this parmlib member. You can rename it according to a naming convention that is specific to your site.

SYSVAUX Summary

This member is read...

During the initialization of an auxiliary address space.

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

None

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

SYSVIEW

The SYSVIEW parmlib member provides the CA SYSVIEW main address space with initialization information. The definitions that are contained in the member determine which additional components and subtasks are started or enabled. The MEM= parameter contained on the SYSVIEW started task procedure specifies the parmlib member to use for initialization information. The SYSVIEW parmlib member is the default member that is specified.

Note: SYSVIEW is the default name of this parmlib member. You can rename it according to a naming convention that is specific to your site.

SYSVIEW Summary

This member is read...

During the initialization of the CA SYSVIEW main address space

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the address space is restarted. The ASADMIN command can be used to control many of the main functions once the address space is active. An operator MODIFY command is also supported.

Subtasks that use this data

CA SYSVIEW main address space

Commands that use this data

None

Associated parmlib members

SYSVUSER

In user parmlib data set?

No

In Persistent data store?

No

SYSVUSER

The SYSVUSER parmlib member provides the CA SYSVIEW user interface address space with initialization information. The definitions that are contained in the member determine which additional components and subtasks are started or enabled. The MEM= parameter contained on the SYSVUSER started task procedure specifies the parmlib member to use for initialization information. The SYSVUSER parmlib member is the default member that is specified.

SYSVUSER is the default name of this parmlib member. You can rename it according to a naming convention that is specific to your site.

SYSVUSER Summary

This member is read...

During the initialization of the CA SYSVIEW user interface address space

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the address space is restarted. The ASADMIN command can be used to control many of the main functions once the address space is active. An operator MODIFY command is also supported.

Subtasks that use this data

CA SYSVIEW user interface address space

Commands that use this data

None

Associated parmlib members

SYSVIEW

In user parmlib data set?

No

In Persistent data store?

No

General Members

ASADMIN

The ASADMIN parmlib member is used to provide a list of the CA SYSVIEW jobs that are displayed when the ASADMIN command is issued. By default, the ASADMIN command displays all active CA SYSVIEW jobs. If you define a complete list of jobs in the ASADMIN parmlib member, then both active and inactive jobs are displayed. The ASADMIN command also displays information about the various subtasks in those address spaces and the status of the subtasks. The ASADMIN command provides an easy method of starting, stopping, or communicating with the individual jobs or subtasks.

ASADMIN Summary

This member is read...

During the initialization of the user session

Can information be modified in real time?

No, the list of entries cannot be altered. The new list is obtained when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

ASADMIN

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

AUDIT

The data in the AUDIT parmlib member is used to set configuration options for the Audit Event component.

AUDIT Summary

This member is read…

During the initialization of the AUDIT subtask

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

AUDIT

Associated parmlib members

AUDITDEF

In user parmlib data set?

No

In Persistent data store?

No

AUDITDEF

The data in the AUDITDEF parameter is used to define the events to audit.

AUDITDEF Summary

This member is read...

During the initialization of the AUDIT subtask

Can information be modified in real time?

Yes, the AUDITDEF command

Subtasks that use this data

None

Commands that use this data

AUDITDEF

Associated parmlib members

AUDIT

In user parmlib data set?

No

In Persistent data store?

Yes

CMDATTR

The CMDATTR parmlib member is used to provide a mechanism for the CA SYSVIEW administrator to override command attributes. Specifically, the CMDATTR parmlib member can be used to remove zIIP eligibility for a command or disable a command.

CMDATTR Summary

This member is read…

During the initialization of the user session.

Can information be modified in real time?

No, the list of command entries cannot be altered. The new command entry list is obtained when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

All commands can be overridden from this parmlib member.

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

DASHBORD

The DASHBORD member can be used to define site-specific dashboards.

DASHBORD Summary

This member is read…

Each time the user command DASHBOARD is entered

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

USERCMD, DASHBOARD

Associated parmlib members

Additional dashboard-related parmlib members can be included in this member.

In user parmlib data set?

Yes

In Persistent data store?

No

FILELIST

The FILELIST member is used to define lists of data sets used by the REXX EXEC FILELIST.

FILELIST Summary

This member is read…

Each time the user command FILELIST is entered

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

USERCMD, FILELIST

Associated parmlib members

GSVXFLST

In user parmlib data set?

No

FLDHELP

The CA SYSVIEW online display commands provide cursor sensitive online help. Cursor sensitivity is based on screen areas at the field level. Adding additional entries to the FLDHELP parmlib member can enhance field level sensitivity.

FLDHELP Summary

This member is read…

The first-time online help information is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

HELP

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

FTPCASTD

The data in this member is used as a template to build dynamically a JCL stream that will FTP data to CA.

FTPCASTD Summary Information:

This member is read…

By the FTPCA command.

Can information be modified in real time?

No.

Subtasks that use this data

None.

Commands that use this data

FTPCA

Associated parmlib members

FTPCATRS, FTPCAVAR, FTPCAXMI

In user parmlib data set?

No

In Persistent data store?

No

FTPCATRS

The data in this member is used as a template to build dynamically a JCL stream. This JCL stream uses the AMATERSE program to create a temporary data set to send to CA with FTP.

FTPCATRS Summary Information:

This member is read…

By the FTPCA command.

Can information be modified in real time?

No.

Subtasks that use this data

None.

Commands that use this data

FTPCA

Associated parmlib members

FTPCASTD, FTPCAVAR, FTPCAXMI

In user parmlib data set?

No

In Persistent data store?

No

FTPCAVAR

The data in this member is included in the FTPCASTD, FTPCATRS, and FTPCAXMI members. The data contains information about the available variables for use in these members.

FTPCAVAR Summary Information:

This member is read…

By the FTPCA command.

Can information be modified in real time?

No.

Subtasks that use this data

None.

Commands that use this data

FTPCA

Associated parmlib members

FTPCASTD, FTPCATRS, FTPCAXMI

In user parmlib data set?

No

In Persistent data store?

No

FTPCAXMI

The data in this member is used as a template to build dynamically a JCL stream. This JCL stream uses the TSO TRANSMIT command to create a temporary data set that to send to CA with FTP.

FTPCAXMI Summary Information:

This member is read…

By the FTPCA command.

Can information be modified in real time?

No.

Subtasks that use this data

None.

Commands that use this data

FTPCA

Associated parmlib members

FTPCASTD, FTPCATRS, FTPCAVAR

In user parmlib data set?

No

In Persistent data store?

No

GROUPS

The GROUPS parmlib member is used to create a list of similar items and then group them logically. Each logical group has an associated type that defines the type of members that can be in the group. A benefit of creating logical groups is that you can reference a list of items by a single name.

Following are some examples of groups that can be defined:

- A logical group of CICS MRO regions
- Logical groups for use by the System Condition Monitor
- A logical group of DASD volumes for exception processing

GROUPS Summary

This member is read...

During the initialization of the CA SYSVIEW main address space. All subtasks and user sessions use the information built by the main address space.

Can information be modified in real time?

Yes, with the GROUPS command

Subtasks that use this data

IMSDATA, MQSDATA, MVSDATA, TCPDATA

Commands that use this data

GROUPS

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

Yes

GSVHHCHK

The GSVHHCHK member can be used to define CA SYSVIEW Health Checks.

GSVHHCHK Summary

This member is read...

During the initialization of the HCHECK task

Can information be modified in real time?

No

Subtasks that use this data

HCHECK

Commands that use this data

None

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

GSVHHCMD

The GSVHHCHK member can be used to define externally Health Check drill-down commands.

GSVHHCMD Summary

This member is read...

When the HCCMD command is issued

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

HCCMD

Associated parmlib members

HCCMDS

In user parmlib data set?

No

In Persistent data store?

No

GSVXFLST

The GSVXFLST member defines product supplied FILELIST definitions for the REXX EXEC FILELIST.

GSVXFLST Summary

This member is read…

Each time the user command FILELIST is entered

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

USERCMD, FILELIST

Associated parmlib members

FILELIST

In user parmlib data set?

No

GSVXLCMD

The GSVXLCMD parmlib member can be used to define externally line commands and associate the definition with an existing CA SYSVIEW primary command.

GSVXLCMD Summary

This member is read...

The first-time an external line command is entered.

Can information be modified in real time?

No, the information cannot be dynamically altered. The information can be reloaded.

Subtasks that use this data

None

Commands that use this data

LISTLCMD

Associated parmlib members

LINECMDS

In user parmlib data set?

No

In Persistent data store?

No

GSVXMORE

The GSVXMORE member can be used to define externally more information drill-down commands. A user or system administrator can create a drill-down command for an associated data collection variable.

GSVXMORE Summary

This member is read…

Each time the MOREINFO command is issued

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

MOREINFO

Associated parmlib members

None

In user parmlib data set?

Yes

In Persistent data store?

No

GSVXUCMD

The GSVXUCMD member can be used to define externally user commands.

GSVXUCMD Summary

This member is read...

During session initialization

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

LISTUCMD

Associated parmlib members

USERCMDS

In user parmlib data set?

No

In Persistent data store?

No

HCCMDS

The HCCMDS parmlib member defines externally a Health Check drill-down command. A user or system administrator can create a real-time link or drill-down for a specific check. This information provides a real-time view of a check.

HCCMDS Summary

This member is read…

By the HCCLIST, HCCMD and HCHECKER commands

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

HCCLIST, HCCMD, HCHECKER

Associated parmlib members

None

In user parmlib data set?

Yes

In Persistent data store?

No

HCHECKS

The HCHECKS parmlib member lets you define site-specific CA SYSVIEW health checks. Use these health checks in place of the CA SYSVIEW health check definitions shipped in the GSVHHCHK parmlib member.

HCHECKS Summary

This member is read...

During the initialization of the HCHECK subtask

Can information be modified in real time?

No

Subtasks that use this data

HCHECK

Commands that use this data

None

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

INFOAREA

The data in the INFOAREA parmlib member is used to provide a user-defined override to the cursor selection process of the information and overview data areas.

INFOAREA Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

As defined in member

Associated parmlib members

None

In user parmlib data set?

Yes

In Persistent data store?

No

INVOKE

Products or programs that can execute in the IBM TSO or IBM ISPF environment can be directly invoked from a CA SYSVIEW session while connected to TSO or ISPF. The INVOKE parmlib member is used to define the external products or programs to CA SYSVIEW. Once defined, the external product can be invoked using the newly assigned command name.

INVOKE Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

INVOKE, PRODUCTS

Associated parmlib members

Any associated parmlib members described in the INVOKE parmlib member definitions

In user parmlib data set?

No

In Persistent data store?

No

JOBNAMES

The JOBNAMES member is used to provide descriptions for job names and started tasks.

JOBNAMES Summary

This member is read...

During session initialization

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

CICSLIST

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

LGLOOKUP

The LGLOOKUP parmlib member is used to associate a logname that is defined in this member with a log stream name. The data in this member is also used to display the list of logs on the LGLOGS command.

The data in this member defines the log streams associated with data collection functions. Use the following online commands to view these log streams:

- CSYSDATA
- CTRANLOG
- CTRANSUM
- XLOG
- IMSRSLOG
- IMSTLOG
- MQRLOG

- PLOTLOG
- SMFLOGI

If one of the previous commands is issued without specifying a log stream name, the data in this member:

- Determines the default log stream for the command.
- Defines any log streams that contain a specific type of data. The first entry of a specific type is defined to be the default log stream for that type.

  Note: The default log stream does not have to be the current log stream in use.

LGLOOKUP Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

LGLOGS, CSYSDATA, CTRANLOG, CTRANSUM, XLOG, IMSTLOG, IMSRSLOG, MQRLOG, PLOTLOG, SMFLOG

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

LINECMDS

The LINECMDS parmlib member can be used to define externally line commands and associate the definition with an existing CA SYSVIEW primary command.

LINECMDS Summary

This member is read...

The first-time an external line command is entered.

Can information be modified in real time?

No, the information cannot be dynamically altered. The RELOAD subcommand of LISTLCMD can be used to reload the definitions.

Subtasks that use this data

None

Commands that use this data

LISTLCMD

Associated parmlib members

GSVXLCMD

In user parmlib data set?

Yes

In Persistent data store?

No

LMPCODES

The LMPCODES parmlib member is used to associate a text product description with a CA Common Services (CCS) for z/OS two-character LMP code. CCS for z/OS occasionally writes messages that contain the two-character LMP code to the system console. This parmlib member and associated command can be used to determine the CA product associated with the code.

LMPCODES Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

CAILMP

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

LOGGER

The LOGGER parmlib member contains a list of configuration options. These options can alter the default operation of the CA SYSVIEW logger services component by overriding the default values.

LOGGER Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

None

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

MAILFOOT

The MAILFOOT parmlib member attaches as a footer the data defined in this member to all emails sent from the SENDMAIL command.

MAILFOOT Summary

This member is read…

When the SENDMAIL command is issued

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

SENDMAIL

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

MENU

CA SYSVIEW uses text members found in the *sysview.*CNM4BPNL data set to define the online menu system. The MENU parmlib member allows you to assign an externally known name to each member in the panellib data set. Additional menus can be created and existing menus can be altered to meet site requirements.

MENU Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session. An individual menu panel can be reloaded by using the MENU RELOAD subcommand.

Subtasks that use this data

None

Commands that use this data

MENU

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

MIBLIST

CA SYSVIEW can browse Management Information Block (MIB) structures created by MIB definitions in the *sysview*.CNM4BMIB data set. The MIBLIST parmlib member provides a list of MIB definitions and associated default TCP/IP options that can be used.

MIBLIST Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

MIBLIST

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

MOREINFO

The MOREINFO parmlib member defines drill-down commands to be associated with data collection metrics.

MOREINFO Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

MOREINFO

Associated parmlib members

None

In user parmlib data set?

Yes

In Persistent data store?

No

MSGACTOV

The MSGACTOV parmlib member defines an alternate action character for messages written to the console.

MSGACTOV Summary

This member is read...

During the initialization of all subtasks and user sessions

Can information be modified in real time?

No

Subtasks that use this data

All

Commands that use this data

None

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

OPTIONS

The OPTIONS parmlib member is used to set the default configuration options for all internal subtasks and online user sessions. These options are not saved in the profile of a user because they are global in nature.

The OPTIONS parmlib member allows you to specify the following:

- Extended private storage limits
- TCP/IP parameters for MIB browsing
- Message uppercase or lowercase translation

OPTIONS Summary

This member is read…

During the initialization of all subtasks and user sessions

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the session is reestablished.

Subtasks that use this data

All

Commands that use this data

All

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

QLIST

The QLIST parmlib member defines the quick list command entries displayed by the QLIST command. The QLIST command screen shows a list of stored SELECT and SORT statements. With this member, you can predefine commonly used command strings such as SELECT and SORT criteria. An entry can contain any valid command string. Multiple command strings can be concatenated within one entry. Use the predefined quick list entries to customize a display or quickly execute a list of commands.

Some example uses of this function include:

- SELECT statements
- SORT statements
- SELECT and SORT statements

Note: You can further simplify the use of the QLIST command by assigning it to a PF key. For instructions about changing PF key definitions, see the *User Guide*.

QLIST Summary

This member is read…

The first-time the QLIST command is entered.

Can information be modified in real time?

Yes, the QLIST command supports a REBUILD subcommand. The REBUILD subcommand rereads the QLIST parameter library members.

Subtasks that use this data

None

Commands that use this data

QLIST

Associated parmlib members

None

In user parmlib data set?

Yes, in both the user and system

In Persistent data store?

No

QUICKCAT

The QUICKCAT parmlib member defines MVS/QuickRef vendor, product, and release names. These definitions are used for categories that do not display a vendor, product, and release name menu.

QUICKCAT Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

QUICKREF

Associated parmlib members

QUICKREF, QUICKVPR

In user parmlib data set?

No

In Persistent data store?

No

QUICKREF

The QUICKREF parmlib member is used to associate a two-character topic identifier with a product message identifier prefix. The CA SYSVIEW online QUICKREF command uses this information to determine the MVS/QuickRef topic identifier when only an item name is provided.

QUICKREF Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

QUICKREF

Associated parmlib members

QUICKCAT, QUICKVPR

In user parmlib data set?

No

In Persistent data store?

No

QUICKVPR

The QUICKVPR member defines the MVS/QuickRef vendor, product, and release names to use for generating internal calls to the QUICKREF command.

QUICKVPR Summary

This member is read…

The first-time the data is requested.

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

QUICKREF

Associated parmlib members

QUICKREF, QUICKCAT

In user parmlib data set?

No

In Persistent data store?

No

SAFAUTH

The SAFAUTH member can be used to define externally the SAF resources to verify using the SAFAUTH command.

SAFAUTH Summary

This member is read…

Each time the SAFAUTH command is issued

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

SAFAUTH

Associated parmlib members

None

In user parmlib data set?

Yes

In Persistent data store?

No

SCHEDULE

CA SYSVIEW provides a component to schedule time-based events. This scheduling component is able to perform various functions at a specified time or a recurring interval. This ability can be used to initiate functions in CA SYSVIEW and functions outside the CA SYSVIEW boundaries. The SCHEDULE parmlib member is used to define a list of scheduled events.

SCHEDULE Summary

This member is read...

During the initialization of the SCHEDULR subtask in the main address space. The data is refreshed or reread each time the SCHEDULR subtask is started.

Can information be modified in real time?

Yes, with the SCHEDULE command

Subtasks that use this data

SCHEDULR

Commands that use this data

SCHEDULE

Associated parmlib members

A scheduled event can be defined as a list of functions. If the event is defined as a list, then that list is defined in a member located in the parmlib data set.

In user parmlib data set?

No

In Persistent data store?

Yes

SYSDATA

CA SYSVIEW has several data collector options. The SYSDATA parmlib member is used to set configuration options that are common for all data collectors.

SYSDATA Summary

This member is read...

During the initialization of the CA SYSVIEW main address space

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is read each time the main address space is restarted.

Subtasks that use this data

CA SYSVIEW main address space

Commands that use this data

None

Associated parmlib members

MVSDATA, MQSDATA, IMSDATA, TCPDATA

In user parmlib data set?

No

In Persistent data store?

No

SYSNAMES

The SYSNAMES parmlib member is used to associate text descriptions and alias names to MVS systems. These alias names can be used when connecting to remote systems using the cross-system component.

SYSNAMES Summary

This member is read…

During the initialization of all subtasks and user sessions

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the session is reestablished.

Subtasks that use this data

All

Commands that use this data

All cross-system commands

Associated parmlib members

None

In user parmlib data set?

Yes

In Persistent data store?

No

TIMEZONE

The TIMEZONE parmlib member is used to define available time zones and associate a text description with each time zone. Each user can select a time zone to control how the date and time information is displayed. The time zone information is saved in the profile of the user.

TIMEZONE Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

TIMEZONE

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

TOPICS

The TOPICS parmlib member defines a list of available online help topics. Individual online help topics are stored as members in the *sysview*.CNM4BHLP data set. Multiple parmlib members can contain lists of online help topics. The online help topics parmlib members are also used when searching online help using the FINDHELP command.

TOPICS Summary

This member is read...

Each time the data is requested

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

FINDHELP, TOPICS

Associated parmlib members

None

In user parmlib data set?

Yes

In Persistent data store?

No

USERCMDS

The USERCMDS parmlib member creates new externally defined user commands. These user command definitions provide an association between a command name and a list of concatenated commands.

USERCMDS Summary

This member is read…

During the initialization of a user session

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

USERCMD

Associated parmlib members

None

In user parmlib data set?

Yes

In Persistent data store?

No

VARIABLE

The VARIABLE parmlib member is used to associate a text description with a data collection metric.

VARIABLE Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

IMSDATA, MQSDATA, MVSDATA, TCPDATA, CICS data collector

Commands that use this data

ACTJOB, ALERTS, CSTATUS, CTHRESH, CTRANLOG, CVARLIST, XLOG, IMSALERT, IMSDATA, IMSMON, IMSSTATE, IMSTHRSH, IMSVARS, IMSWAITS, MONITOR, MQALERTS, MQMON, MQSDATA, MQSTATES, MQTHRESH, MQVARS, MVSDATA, PAGING, PLOT, PLOTLOG, SMFLOG, STATES, STORAGE, SWAPPING, TCPDATA, TCPMON, TCPSTATE, TCPTHRSH, THRESH, URESOURC, VARS

Associated parmlib members

CICSTHRS, CICSSTAT, IMSTHRSH, MQSTHRSH, MVSTHRSH, TCPTHRSH

In user parmlib data set?

No

In Persistent data store?

No

XSYSTEM

The XSYSTEM parmlib member contains a list of configuration options that alter the default operation of the CA SYSVIEW cross-system component. The cross-system component allows a user to log on to a session and remotely connect to other systems. The cross-system communications use the CAICCI component of CCS for z/OS.

The configuration options found in the XSYSTEM parmlib member can be used to configure the timeout parameters of the cross-system communications.

XSYSTEM Summary

This member is read...

During the initialization of all subtasks and user sessions

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user session or subtask is restarted.

Subtasks that use this data

All

Commands that use this data

All cross-system related commands

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

z/OS Members

APPLMON

The APPLMON parmlib member is used to provide an initial list of VTAM application IDs to monitor for availability by the VTAM application monitor component. The VTAM application monitor runs as a subtask in the CA SYSVIEW main address space.

The availability of the applications listed in this member is displayed when the APPLMON command is issued. The current availability status of the applications and their availability over the past 24 hours is displayed. Exception alerts can also be generated based on this collection by creating MONITOR and STATE definitions in the MVSMON and MVSSTATE parmlib members.

The APPLMON parmlib member contains two sections, a control section and a monitoring section. The control section is where you specify setup options. The control statement defines the VTAM APPL name that the APPLMON task uses to communicate with VTAM. This statement must be supplied and it must be active at the time the APPLMON task is started.

The monitoring section is where you define VTAM applications to monitor for availability. By default, no applications are monitored. At least one monitor statement must be supplied; however, there is no limit to the number of statements you can supply.

APPLMON Summary

This member is read...

During the initialization of the APPLMON subtask in the main address space. The data is refreshed or reread each time the APPLMON subtask is started.

Can information be modified in real time?

Yes, with the APPLMON command

Subtasks that use this data

APPLMON

Commands that use this data

APPLMON

Associated parmlib members

MVSMON

Defines MVS-related resources monitored by the MVS data collector.

MVSSTATE

Defines the states of MVS-related resources for monitoring.

SYSVIEW

Starts the APPLMON subtask, which is not started by default.

In user parmlib data set?

No

In Persistent data store?

Yes

CMDACCPT

The CMDACCPT parmlib member is used to provide a list of those responses to a z/OS operator command that is produced immediately. Then a relatively long pause follows before the remainder of the response lines is returned. For some groups of messages, there is no way to define the last message in a group.

CMDACCPT Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the list of entries cannot be altered. The new list is obtained when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

XMVS

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

DESTID

The DESTID parmlib member is used to associate a text description with a JES2 destination ID.

DESTID Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

DESTID

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

DYNEXIT

The DYNEXIT parmlib member is used to associate a text description with an MVS dynamic exit name.

DYNEXIT Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

DYNEXIT

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

ENFLIST

The ENFLIST parmlib member is used to associate a text description with an ENF event code. An ENF event code is a decimal number.

ENFLIST Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

ENFLIST

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

ESRTABLE

The ESRTABLE parmlib member is used to associate a text description with an Extended SVC router table entry.

ESRTABLE Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

ESRTABLE, TASK, TRACE

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

HCOWNER

The data is this member is used to provide the descriptive text displayed by the HCSUMM command.

HCOWNER Summary Information:

This member is read…

By the HCOWNER command

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

HCOWNER

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

IPLSTATS

The IPLSTATS member is used to associate a text description with an IPLSTATS command type and module name.

IPLSTATS Summary

This member is read…

Each time the IPLSTATS command is issued

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

USERCMD, DASHBOARD

Associated parmlib members

DASHBORD

In user parmlib data set?

No

In Persistent data store?

No

JES

The JES parmlib member associates JES configuration modules and related data such as MAP member names to a specific system name.

JES Summary

This member is read…

During the initialization of all subtasks and user sessions

Can information be modified in real time?

No

Subtasks that use this data

All

Commands that use this data

All JES-related commands

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

MVSDATA

The CA SYSVIEW MVS data collector can be customized using the configuration options in the MVSDATA parmlib member. The following are some of the ways the MVS data collector can be configured:

- Specify the threshold definitions member
- Enable monitoring options

MVSDATA Summary

This member is read...

During the initialization of the MVSDATA subtask in the main address space. The data is refreshed or reread each time the MVSDATA subtask is started.

Can information be modified in real time?

Yes, the information can be updated online.

Subtasks that use this data

MVSDATA

Commands that use this data

MVSDATA

Associated parmlib members

MVSTHRSH, MVSMON, MVSSTATE

In user parmlib data set?

No

In Persistent data store?

No

MVSMON

The MVS data collector monitors MVS-related resources. Exception or threshold alerts can be generated when resource usage exceeds the defined range. The MVSMON parmlib member is used to define MVS-related resources for monitoring by the MVS data collector.

Following are MVS-related resources that can be monitored:

- Address spaces
- Channel, DASD, and tape devices
- UNIX System Services
- JES2

- Processors
- VTAM application availability
- Storage, paging, and swapping

The MVS data collector uses the definitions or rules defined in this member to determine which resources to monitor and specify threshold intervals by resource group.

MVSMON Summary

This member is read…

During the initialization of the MVSDATA subtask in the main address space. The data is refreshed or reread each time the MVSDATA subtask is started.

Can information be modified in real time?

Yes, the information can be updated online.

Subtasks that use this data

MVSDATA

Commands that use this data

MVSDATA

Associated parmlib members

MVSDATA, MVSSTATE, MVSTHRSH

In user parmlib data set?

No

In Persistent data store?

Yes

MVSSTATE

The MVS data collector monitors MVS-related resources. Exception or threshold alerts can be generated when resource usage exceeds the defined range. The MVSSTATE parmlib member is used to define the states of MVS-related resources for monitoring.

The alerting mechanism can trigger any of the following actions:

- Write a message to the job log
- Write a message to the console
- Execute a predefined IMOD (REXX EXEC)
- Send an SNMP alert trap

- Send an event notification request to CA OPS/MVS
- Execute an event capture member

The MVS data collector (MVSDATA subtask) uses the definitions or rules defined in this member to specify individual or generic wildcard state definitions.

MVSSTATE Summary

This member is read...

During the initialization of the MVSDATA subtask in the main address space. The data is refreshed or reread each time the MVSDATA subtask is started.

Can information be modified in real time?

Yes, with the STATES command.

Subtasks that use this data

MVSDATA

Commands that use this data

STATES

Associated parmlib members

MVSDATA

In user parmlib data set?

No

In Persistent data store?

Yes

MVSTHRSH

The MVS data collector monitors MVS-related resources. Exception or threshold alerts can be generated when resource usage exceeds the defined range. The MVSTHRSH parmlib member is used to define thresholds for the resources being monitored.

The alerting mechanism can cause the MVS data collector to generate alerts as follows:

- Write a message to the job log
- Write a message to the console
- Execute a predefined IMOD (REXX EXEC)
- Send an SNMP alert trap
- Send an event notification request to CA OPS/MVS
- Execute an event capture member

Following are MVS-related resources that can be monitored:

- Address spaces
- Channel, DASD, and tape devices
- UNIX System Services
- JES2
- Processors
- VTAM application availability
- Storage, paging, and swapping

MVSTHRSH Summary

This member is read…

During the initialization of the MVSDATA subtask in the main address space. The data is refreshed or reread each time the MVSDATA subtask is started.

Can information be modified in real time?

Yes, with the THRESH command.

Subtasks that use this data

MVSDATA

Commands that use this data

ALERTS, ACTJOB, PLOT, THRESH, STORAGE, PAGING, SWAPPING

Associated parmlib members

MVSDATA, MVSMON, MVSSTATE

In user parmlib data set?

No

In Persistent data store?

Yes

PFTQNAME

The PFTQNAME parmlib member is used to associate a text description with a Page Frame Table queue identifier.

PFTQNAME Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

PFT

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

PPT

The PPT parmlib member is used to associate a text description with a Program Properties Table program name.

PPT Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

PPT

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

PROCLIST

The PROCLIST parmlib member is used to identify additional PROCLIB ddnames allocated to a JES2 address space. The ddnames allocated to the JES2 address space that are of the format PROC*nn* are assumed to be PROCLIBs and do not need specified in this member.

PROCLIST Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

PROCLIST

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

SFTABLE

The SFTABLE parmlib member is used to associate a text description with a System Function Table entry number.

SFTABLE Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

SFTABLE

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

SMF

The SMF parmlib member is used to define the name of a procedure that is used to dump an SMF data set. Issuing the DUMP line command from the SMF display starts the specified procedure.

SMF Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

SMF

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

SUBSYS

The SUBSYS parmlib member is used to associate a text description with a subsystem vector table function code.

SUBSYS Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

SUBSYS

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

SVCTABLE

The SVCTABLE parmlib member is used to associate a text description with a supervisor call number.

SVCTABLE Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

SVCTABLE, TASK, TRACE

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

TASK

The TASK parmlib member is used to associate a text description with a subtask or Request Block (RB) name.

TASK Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

TASK

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

TRACE

The TRACE parmlib member is used to associate a text description with a system service identifier (SSRV).

TRACE Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

TRACE

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

UNITNAME

The UNITNAME parmlib member is used to associate a unit name with a UCBTYP value. Make an entry in this member only for those Unit Control Block (UCB) device types not supported by the following standard IBM services:

- IEFEB4UV
- EDTINFO RTNUNIT

UNITNAME Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

ALLOCAS, CACHECTL, CACHEDEV, DASD, DEVPATH, DSALLOC, DSCAT, DSINFO, EDT, IOCDATA, IPLINFO, LISTDIR, LLASTATS, LOGREC, MQPAGE, PAGEDS, PLEXCPL, QUERY, SET, SMSVOLS, SPACE, STATUS, SUBCHAN, SWAPDS, TAPE, TRACE, UNIT, VTOC

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

VTAM

The VTAM parmlib member defines a list of simplified or shortcut ACF/VTAM operator commands. The shortcuts are used with the VTAM online command.

VTAM Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

VTAM

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

WTORSTAT

The WTORSTAT parmlib member is used to associate a status with a write-to-operator reply message. The status can be assigned based on any of the following items associated with the message:

- Issuing system name
- Issuing job name
- Message text
- Message identifier

WTORSTAT Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

MVSDATA

Commands that use this data

WTOR

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

CICS Members

CICSARTM

The CICSARTM parmlib member contains a list of Automated Response Time Management (ARTM) definitions. Each definition describes a CICS transaction that is to participate in dynamic priority assignment. Each transaction that is specified must have a target response time and maximum priority specified for it. The priority of the transaction is raised and lowered to achieve the target response time.

CICSARTM Summary

This member is read...

During CA SYSVIEW Option for CICS data collection initialization in each CICS address space. The data is refreshed or reread each time the collector is started.

Can information be modified in real time?

Yes, with the CARTM command.

Subtasks that use this data

CICS data collection

Commands that use this data

CARTM

Associated parmlib members

CICSOPTS

Use this member to set the following options to alter the default attributes of ARTM:

- ARTM-ACTIVE
- ARTM-DEFAULT-TARGET
- ARTM-DYNAMIC-ADD
- ARTM-MAXIMUM-PRIORITY
- ARTM-MEMBER
- ARTM-TRIGGER-COUNT
- ARTM-WTRTIME

In user parmlib data set?

No

In Persistent data store?

Yes

CICSCMDS

The CICSCMDS parmlib member contains a list of MVS operator commands to issued immediately after each CICS data logger completes initialization. The CICS data logger executes in the main address space. The main address space is the anchor point for all associated CICS data collectors executing in a CICS address space.

Because the CICS logger subtask must be active before the initialization of CICS data collection, the CICSCMDS member provides a convenient method of automatically starting all associated CICS regions.

CICSCMDS Summary

This member is read...

During the initialization of the CICSLOGR subtask in the main address space. The data is refreshed or reread each time the CICSLOGR subtask is started.

Can information be modified in real time?

No

Subtasks that use this data

CICSLOGR

Commands that use this data

None

Associated parmlib members

CICSLOGR

Use this member to set the following options:

- STARTUP-COMMANDS
- STARTUP-COMMANDS-MEMBER

In user parmlib data set?

No

In Persistent data store?

No

CICSCNCL

The CICSCNCL parmlib member contains a list of transactions for CA SYSVIEW Option for CICS to cancel automatically at CICS shutdown or termination. CICS requires all transactions be completed before normal shutdown. For this reason, it is common to have a CICS shutdown delay due to a conversational transaction waiting on terminal input. If you specify long-running transactions in this member, then CICS can terminate quickly because you are not forced to wait for these transactions to complete.

Transactions that are specified in this member are canceled immediately upon detection of CICS shutdown. The following transaction attributes can be specified to determine which transactions, if any, to cancel:

- Transaction ID
- Facility name (terminal, transient data queue)
- Suspend type
- Suspend value

CICS transactions can be automatically canceled based on a threshold exception definition. The threshold definitions can be found in the CICSTHRS parmlib member. You can override or exclude a transaction from cancellation based on the current wait reason of the transaction.

CICSCNCL Summary

This member is read...

During CA SYSVIEW Option for CICS data collection initialization in each CICS address space. The data is refreshed or reread each time the collector is started.

Can information be modified in real time?

No

Subtasks that use this data

CICS data collection

Commands that use this data

None

Associated parmlib members

CICSOPTS

Use this member to set the following options:

- TRANSACTION-CANCEL-AT-SHUTDOWN
- TRANSACTION-CANCEL-MEMBER

In user parmlib data set?

No

In Persistent data store?

Yes

CICSDMPM

The CA SYSVIEW Option for CICS Dump Management function can be used to enhance the CICS-provided function to suppress unwanted dumps. The built-in function provided by CICS only allows dumps to be suppressed based on the dump code. The CICSDMPM parmlib member provides an initial list of dump suppression criteria.

This parmlib member allows you to suppress dumps based on the following parameters:

- Transaction ID
- Program name
- Dump code

CICSDMPM Summary

This member is read...

During CA SYSVIEW Option for CICS data collection initialization in each CICS address space. The data is refreshed or reread each time the collector is started.

Can information be modified in real time?

Yes, with the CDUMPMGT command

Subtasks that use this data

CICS data collection

Commands that use this data

CDUMPMGT

Associated parmlib members

CICSOPTS

Use this member to set the following options:

- DUMP-MANAGEMENT
- DUMP-MANAGEMENT-MEMBER

In user parmlib data set?

No

In Persistent data store?

Yes

CICSLOGR

The CICS Data Logger is the common anchor point for CICS data collector subtasks that are executing in each CICS address space. CICSLOGR performs the logging of records created by the data collectors. Data collection includes the logging of records to SMF and to the internal logs for online display. The CICS Data Logger subtask must be active before the initialization of CICS Data Collection. You can use the subtask to automate the starting of all associated CICS regions. More than one CICS Data Logger can be created.

The CICSLOGR parmlib member allows you to override the default values of the following:

- Size of online logs
- SMF record type
- Log destinations
- Start commands member

CICSLOGR Summary

This member is read...

During the initialization of the CICSLOGR subtask in the main address space. The data is refreshed or reread each time the CICSLOGR subtask is started.

Can information be modified in real time?

No

Subtasks that use this data

CICSLOGR

Commands that use this data

None

Associated parmlib members

CICSCMDS

Use this member to specify a list of z/OS commands to issue during initialization.

In user parmlib data set?

No

In Persistent data store?

No

CICSOPTS

The CICSOPTS parmlib member contains a list of CICS data configuration options, which alter the default operation of the CICS data collector. The configuration options can control the following list of items:

- Definition of the CICS Data Logger connection
- Detailed transaction collection options
- Creation of SMF or log records
- Component activation
- Component parmlib members

CICSOPTS Summary

This member is read...

During CA SYSVIEW Option for CICS data collection initialization in each CICS address space. The data is refreshed or reread each time the collector is started.

Can information be modified in real time?

Yes, with the CCONFIG command

Subtasks that use this data

CICS data collection

Commands that use this data

CCONFIG

Associated parmlib members

CICSARTM, CICSCNCL, CICSDMPM, CICSLOGR,

CICSSTAT, CICSTHRS, CICSTOPT

In user parmlib data set?

No

In Persistent data store?

No

CICSSTAT

Additional user-requested CICS data can be collected and summarized using the variable collection and status option. The CICSSTAT parmlib member is used to define those variables for which additional collection is performed. The data is collected and summarized over a one-hour interval.

You can have any of the following information collected and summarized:

- CICS system resources
- Transaction ID
- Terminal ID
- Program name
- File name

CICSTAT Summary

This member is read...

During CA SYSVIEW Option for CICS data collection initialization in each CICS address space. The data is refreshed or reread each time the collector is started.

Can information be modified in real time?

Yes, with the CSTATUS command

Subtasks that use this data

CICS data collection

Commands that use this data

CSTATUS

Associated parmlib members

CICSOPTS

Use this member to set the following options:

- VSTATUS-COLLECTION
- VSTATUS-MEMBER
- VSTATUS-TOTAL-INTERVAL
- VSTATUS-UMBRELLA

In user parmlib data set?

No

In Persistent data store?

Yes

CICSTHRS

The CICS data collector monitors CICS system and transaction resources. Exception or threshold alerts can be generated when resource usage exceeds the defined range. The CICSTHRS parmlib member is used to define the system and transaction resources to monitor.

The alerting mechanism can trigger any of the following actions:

- Write a message to the job log
- Write a message to the console
- Execute a predefined IMOD (REXX EXEC)
- Cancel the transaction
- Send an SNMP alert trap

- Send an event notification request to CA OPS/MVS
- Execute an event capture member

CICSTHRS Summary

This member is read...

During CA SYSVIEW Option for CICS data collection initialization in each CICS address space. The data is refreshed or reread each time the collector is started.

Can information be modified in real time?

Yes, with the CTHRESH command

Subtasks that use this data

CICS data collection

Commands that use this data

CTHRESH, CSTATUS, CTASKS, CTRANS, PLOT

Associated parmlib members

CICSOPTS

Use this member to set the following options:

- THRESHOLD-LOAD-INACTIVE
- THRESHOLD-MEMBER
- THRESHOLD-SYSTEM
- THRESHOLD-TASK-DYNAMIC
- THRESHOLD-TASK-END
- THRESHOLD-UMBRELLA

In user parmlib data set?

No

In Persistent data store?

Yes

CICSTOPT

The CICS data collector can monitor every CICS transaction executing in a CICS region. On some occasions, you want to alter the monitoring characteristics for specific transaction IDs. The CICSTOPT parmlib member is used to define rules for monitoring specific transactions.

The following options can be set for each transaction ID:

- Exclude from online statistics and logging
- Exclude from logging
- Do not perform threshold or exception checking
- Bypass all data collection
- Suppress CICS SMF 110 records

CICSTOPT Summary

This member is read…

During CA SYSVIEW Option for CICS data collection initialization in each CICS address space. The data is refreshed or reread each time the collector is started.

Can information be modified in real time?

Yes, with the CTRANOPT command

Subtasks that use this data

CICS data collection

Commands that use this data

CTRANOPT

Associated parmlib members

CICSOPTS

Use this member to set the following options:

- UMBRELLA-COUNT-LIMIT
- UMBRELLA-MEMBER
- UMBRELLA-TRAN-WILDCARD

In user parmlib data set?

No

In Persistent data store?

Yes

EIBCODES

The EIBCODES parmlib member is used to associate a text description with CICS EIB function codes.

EIBCODES Summary

This member is read…

The first time the data is requested

Can information be modified in realtime?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

CTASKS, CTRANLOG

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

CA Insight DPM for DB2 Members

DB2

CA SYSVIEW can communicate with and report on CA Insight DPM for DB2 monitored DB2 subsystems. The DB2 parmlib member is used to define the CA Insight DPM for DB2 connections.

DB2 Summary

This member is read...

During the initialization of all subtasks and user sessions

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the session is reestablished.

Subtasks that use this data

All

Commands that use this data

All CA SYSVIEW Component for CA Insight DPM for DB2 related commands.

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

CA Datacom Members

DATACOM

CA SYSVIEW can communicate with and report on multiple CA Datacom/DB Multi-User Facility (MUF) address spaces. The DATACOM parmlib member can be used to provide a list of possible MUF job names. A definition entry for each MUF job name is *not* required. A list of the active CA Datacom jobs is obtained dynamically. You can define all potential entries so that the DCLIST command display includes jobs that have never been active.

DATACOM Summary

This member is read...

During the initialization of all subtasks and user sessions

Can information be modified in real time?

No, the list of entries cannot be manually altered online. The list is automatically updated for any additional active jobs. A new list is obtained when the session is reestablished.

Subtasks that use this data

All

Commands that use this data

All CA Datacom related commands

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

IMS Members

IMS

The IMS parmlib member contains a list of configuration options that alter the default operation of the CA SYSVIEW Option for IMS. These options are not saved in the profile of a user because they are global in nature.

IMS Summary

This member is read...

During the initialization of all subtasks and user sessions

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the session is reestablished.

Subtasks that use this data

All

Commands that use this data

All IMS-related online commands

Associated parmlib members

IMSLIST

In user parmlib data set?

No

In Persistent data store?

No

IMSCMDS

The IMSCMDS parmlib member contains a list of z/OS operator commands to issue immediately after the IMS data collection subtask completes initialization. The IMS data collector executes in the main address space.

The IMSCMDS member provides a convenient method of automatically starting IMS regions after CA SYSVIEW has been initialized.

IMSCMDS Summary

This member is read...

During the initialization of the IMSDATA subtask in the main address space. The data is refreshed or reread each time the IMSDATA subtask is started.

Can information be modified in real time?

No

Subtasks that use this data

IMSDATA

Commands that use this data

None

Associated parmlib members

IMSDATA

Use this member to define the following configuration options, which control the usage of the IMSCMDS parmlib member:

- IMS-STARTUP-COMMANDS
- IMS-STARTUP-COMMANDS-MEMBER

In user parmlib data set?

No

In Persistent data store?

No

IMSDATA

You can customize the CA SYSVIEW Option for IMS data collection option using the configuration options in the IMSDATA parmlib member. The following is a sample of items that you can configure:

- Enable autostart of IMS data loggers
- Define the IMS initialization commands member

- Enable IMS monitoring options
- Enable IMS plot collection

IMSDATA Summary

This member is read…

During the initialization of the IMSDATA subtask in the main address space. The data is refreshed or reread each time the IMSDATA subtask is started.

Can information be modified in real time?

Yes, the information can be updated online.

Subtasks that use this data

IMSDATA

Commands that use this data

IMSDATA

Associated parmlib members

IMSCMDS, IMSLOGR, IMSTHRSH, IMSMON, IMSSTATE

In user parmlib data set?

No

In Persistent data store?

No

IMSLIST

The IMSLIST parmlib member is used to provide the names and descriptions of IMS control regions displayed by the IMSLIST command. If an IMS control region is defined in this member, then the IMSLIST command displays the status of the job even when it is not active. IMS control regions, which are not defined in this member, are only displayed when active.

MSLIST Summary

This member is read...

The first-time the IMSLIST command requests the data

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

IMSLIST

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

IMSLOGR

The IMS Data Logger collects detailed IMS transaction information. The IMS Data Loggers are subtasks that execute in the CA SYSVIEW main address space. IMS Data Loggers are not required when detailed IMS transaction information is not needed. A separate IMS Data Logger is required for each IMS control region that requires monitoring. IMS Data Loggers use the autostart function defined in the IMSDATA parmlib member to start automatically when a new IMS control region is found.

The IMS Data Logger performs:

- The logging of records created by the data collector
- The logging of records to SMF
- The logging of records to the internal logs for online display

The IMSLOGR parmlib member lets you override the default values for the following:

- SMF record type
- Log destinations

IMSLOGR Summary

This member is read...

During the initialization of the IMSLOGR subtask. The data is refreshed or reread each time the IMSLOGR subtask is started.

Can information be modified in real time?

No

Subtasks that use this data

IMSLOGR

Commands that use this data

None

Associated parmlib members

IMSDATA

In user parmlib data set?

No

In Persistent data store?

No

IMSMON

The IMSMON parmlib member is used to define the system and transaction resources to monitor.

Following are IMS-related resources that can be monitored:

- Control regions
- Transactions
- Pools

The IMS data collector uses the definitions or rules defined in this member to:

- Determine which resources to monitor
- Specify threshold intervals by resource group

IMSMON Summary

This member is read...

During the initialization of the IMSDATA subtask in the main address space. The data is refreshed or reread each time the IMSDATA subtask starts.

Can information be modified in real time?

Yes, the information can be updated online.

Subtasks that use this data

IMSDATA

Commands that use this data

IMSDATA

Associated parmlib members

IMSDATA, IMSSTATE, IMSTHRSH

In user parmlib data set?

No

In Persistent data store?

Yes

IMSSTATE

The CA SYSVIEW Option for IMS data collector monitors IMS system and transaction resources. Exception or threshold alerts can be generated when resource usage exceeds the defined range. The IMSSTATE parmlib member is used to define the state of system and transaction resources to monitor.

The alerting mechanism can trigger any of the following actions:

- Write a message to the job log
- Write a message to the console
- Execute a predefined IMOD (REXX EXEC)
- Send an SNMP alert trap
- Send an event notification request to CA OPS/MVS
- Execute an event capture member

The IMS data collector uses the definitions or rules defined in this member to specify individual or generic wildcard state definitions.

IMSSTATE Summary

This member is read...

During the initialization of the IMSDATA subtask in the main address space. The data is refreshed or reread each time the IMSDATA subtask is started.

Can information be modified in real time?

Yes, with the IMSSTATE command.

Subtasks that use this data

IMSDATA

Commands that use this data

IMSSTATE

Associated parmlib members

IMSDATA, IMSMON, IMSTHRSH

In user parmlib data set?

No

In Persistent data store?

Yes

IMSTHRSH

The CA SYSVIEW Option for IMS data collector monitors IMS system and transaction resources. Exception or threshold alerts can be generated when resource usage exceeds the defined range. The IMSTHRSH parmlib member is used to define the system and transaction resources to monitor.

The alerting mechanism can trigger any of the following actions:

- Write a message to the job log
- Write a message to the console
- Execute a predefined IMOD (REXX EXEC)
- Send an SNMP alert trap
- Send an event notification request to CA OPS/MVS
- Execute an event capture member

Following are IMS-related resources that can be monitored:

- Control regions
- Transactions
- Pools

IMSTHRSH Summary

This member is read…

During the initialization of the IMSDATA subtask in the main address space. The data is refreshed or reread each time the IMSDATA subtask is started.

Can information be modified in real time?

Yes, with the IMSTHRSH command.

Subtasks that use this data

IMSDATA, IMSLOGR

Commands that use this data

IMSTHRSH

Associated parmlib members

IMSDATA, IMSMON, IMSSTATE

In user parmlib data set?

No

In Persistent data store?

Yes

IMSTYPE

The IMSTYPE parmlib member is used to associate a text description with an IMS monitor log record type and subtype.

IMSTYPE Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

IMSLOG

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

TCP/IP Members

IPPORTS

The IPPORTS parmlib member displays the names and descriptions of TCP and UDP port numbers. The names are used on various TCPIP option commands to name the ports displayed.

IPPORTS Summary

This member is read...

The first-time the data is requested

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

IPLISTEN, IPPORTS, IPTCONN, IPUCONN

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

TCPDATA

The TCPDATA parmlib member provides configuration options to customize the CA SYSVIEW Option for TCP/IP data collector.

The following are some of the ways the TCP data collector can be configured:

- Specify the monitor, state, and threshold members
- Enable monitoring options

TCPDATA Summary

This member is read...

During the initialization of the TCPDATA subtask in the main address space. The data is refreshed or reread each time the TCPDATA subtask is started.

Can information be modified in real time?

Yes, the information can be updated online.

Subtasks that use this data

TCPDATA

Commands that use this data

TCPDATA

Associated parmlib members

TCPTHRSH, TCPMON, TCPSTATE

In user parmlib data set?

No

In Persistent data store?

No

TCPMON

The TCPMON parmlib member is used to define the TCP/IP jobnames that you want monitored by the CA SYSVIEW Option for TCP/IP data collector. Exception or threshold alerts can be generated when resource usage exceeds the defined range.

The TCP/IP data collector uses the definitions or rules defined in this member to do the following:

- Determine which resources to monitor
- Specify threshold intervals by resource group

TCPMON Summary

This member is read...

During the initialization of the TCPDATA subtask in the main address space. The data is refreshed or reread each time the TCPDATA subtask is started.

Can information be modified in real time?

Yes, the information can be updated online.

Subtasks that use this data

TCPDATA

Commands that use this data

TCPDATA

Associated parmlib members

TCPDATA, TCPSTATE, TCPTHRSH

In user parmlib data set?

No

In Persistent data store?

Yes

TCPSTATE

The TCPSTATE parmlib member is used to define the state of TCP/IP resources that you want monitored by the CA SYSVIEW Option for TCP/IP data collector. Exception or threshold alerts can be generated when resource usage exceeds the defined range.

The alerting mechanism can trigger any of the following actions:

- Write a message to the job log
- Write a message to the console
- Execute a predefined IMOD (REXX EXEC)
- Send an SNMP alert trap
- Send an event notification request to CA OPS/MVS
- Execute an event capture member

The TCP/IP data collector uses the definitions or rules defined in this member to specify individual or generic wildcard state definitions.

TCPSTATE Summary

This member is read...

During the initialization of the TCPDATA subtask in the main address space. The data is refreshed or reread each time the TCPDATA subtask is started.

Can information be modified in real time?

Yes, with the TCPSTATE command.

Subtasks that use this data

TCPDATA

Commands that use this data

TCPSTATE

Associated parmlib members

TCPDATA

In user parmlib data set?

No

In Persistent data store?

Yes

TCPTHRSH

The TCPTHRSH parmlib member is used to define the TCP/IP resources that you want monitored by the CA SYSVIEW Option for TCP/IP data collector. Exception or threshold alerts can be generated when resource usage exceeds the defined range.

The alerting mechanism can trigger any of the following actions:

- Write a message to the job log
- Write a message to the console
- Execute a predefined IMOD (REXX EXEC)
- Send an SNMP alert trap
- Send an event notification request to CA OPS/MVS
- Execute an event capture member

The TCP/IP data collector uses the definitions or rules defined in this member to specify individual or generic wildcard threshold definitions.

TCPTHRSH Summary

This member is read...

During the initialization of the TCPDATA subtask in the main address space. The data is refreshed or reread each time the TCPDATA subtask is started.

Can information be modified in real time?

Yes, with the TCPTHRSH, TCPSTATE, and TCPMON commands.

Subtasks that use this data

TCPDATA

Commands that use this data

TCPTHRSH

Associated parmlib members

TCPDATA, TCPMON, TCPSTATE

In user parmlib data set?

No

In Persistent data store?

Yes

WebSphere MQ Members

MQPARMS

The CA SYSVIEW Option for WebSphere MQ uses the MQPARMS parmlib member to associate text descriptions with initialization parameters on the MQPARMS command.

MQPARMS Summary

This member is read...

The first-time the data is requested.

Can information be modified in real time?

No

Subtasks that use this data

None

Commands that use this data

MQPARMS

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

MQSDATA

The CA SYSVIEW Option for WebSphere MQ data collection can be customized using the configuration options in the MQSDATA parmlib member.

The following is a sample of items that can be configured:

- Specify the monitor, state, and threshold members
- Enable monitoring options

MQSDATA Summary

This member is read…

During the initialization of the MQSDATA subtask in the main address space. The data is refreshed or reread each time the MQSDATA subtask is started.

Can information be modified in real time?

Yes, the information can be updated online.

Subtasks that use this data

MQSDATA

Commands that use this data

MQSDATA

Associated parmlib members

MQSERIES, MQSTHRSH, MQSMON, MQSSTATE

In user parmlib data set?

No

In Persistent data store?

No

MQSERIES

The MQSERIES parmlib member contains a list of configuration options that alter the default operation of the CA SYSVIEW Option for WebSphere MQ online option. These options are not saved in the profile of a user because they are global in nature.

MQSERIES Summary

This member is read…

During the initialization of all subtasks and user sessions.

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the session is reestablished.

Subtasks that use this data

All

Commands that use this data

All WebSphere MQ-related online commands

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

MQSMON

The WebSphere MQ data collector monitors WebSphere MQ resources. Exception or threshold alerts can be generated when resource usage exceeds the defined range.

The MQSMON parmlib member is used to define the WebSphere MQ resources to monitor.

Following are the WebSphere MQ-related resources that can be monitored:

- Queue managers
- Queues
- Channels
- Pagesets

The WebSphere MQ data collector uses the definitions or rules defined in this member to:

- Determine which resources to monitor
- Specify threshold intervals by resource group

MQSMON Summary

This member is read…

During the initialization of the MQSDATA subtask in the main address space. The data is refreshed or reread each time the MQSDATA subtask is started.

Can information be modified in real time?

Yes, the information can be updated online.

Subtasks that use this data

MQSDATA

Commands that use this data

MQSDATA

Associated parmlib members

MQSDATA, MQSSTATE, MQSTHRSH

In user parmlib data set?

No

In Persistent data store?

Yes

MQSREQS

The MQSREQS parmlib member contains a list of configuration options that alter the default operations of the CA SYSVIEW Option for WebSphere MQ MQI Request History option.

MQSREQS Summary

This member is read…

When MQSDATA determines that a queue manager is eligible for MQI request history monitoring.

Can information be modified in real time?

YES, with the MQRCONF command.

Subtasks that use this data

MQSDATA

Commands that use this data

MQRCONF

Associated parmlib members

MQSDATA

In user parmlib data set?

No

In Persistent data store?

No

MQSSTATE

The WebSphere MQ data collector monitors WebSphere MQ resources. Exception or threshold alerts can be generated when resource usage exceeds the defined range. The MQSSTATE parmlib member is used to define the states of WebSphere MQ resources to monitor.

The alerting mechanism can trigger any of the following actions:

- Write a message to the job log
- Write a message to the console
- Execute a predefined IMOD (REXX EXEC)
- Send an SNMP alert trap
- Send an event notification request to CA OPS/MVS
- Execute an event capture member

The WebSphere MQ data collector uses the definitions or rules defined in this member to specify individual or generic wildcard state definitions.

MQSSTATE Summary

This member is read...

During the initialization of the MQSDATA subtask in the main address space. The data is refreshed or reread each time the MQSDATA subtask is started.

Can information be modified in real time?

Yes, with the MQSTATES command.

Subtasks that use this data

MQSDATA

Commands that use this data

MQSTATES

Associated parmlib members

MQSDATA, MQSMON, MQSTHRSH

In user parmlib data set?

No

In Persistent data store?

Yes

MQSTHRSH

The WebSphere MQ data collector monitors WebSphere MQ resources. Exception or threshold alerts can be generated when resource usage exceeds the defined range. The MQSTHRSH parmlib member is used to define the thresholds of resources to monitor. The alerting mechanism can trigger any of the following actions:

- Write a message to the job log
- Write a message to the console
- Execute a predefined IMOD (REXX EXEC)
- Send an SNMP alert trap
- Send an event notification request to CA OPS/MVS
- Execute an event capture member

MQSTHRSH Summary

This member is read…

During the initialization of the MQSDATA subtask in the main address space. The data is refreshed or reread each time the MQSDATA subtask is started.

Can information be modified in real time?

Yes, with the MQTHRESH command.

Subtasks that use this data

MQSDATA

Commands that use this data

MQALERTS, MQTHRESH, PLOT

Associated parmlib members

MQSDATA, MQSMON, MQSSTATE

In user parmlib data set?

No

In Persistent data store?

Yes

UNIX System Services Members

USS

The USS parmlib member contains a list of configuration options that alter the default operation of the CA SYSVIEW USS online option.

USS Summary

This member is read...

During the initialization of all subtasks and user sessions

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the session is reestablished.

Subtasks that use this data

All

Commands that use this data

All USS-related commands

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

Event Capture Members

CAPTURE

The CAPTURE parmlib member is used to set the default configuration options of the CA SYSVIEW Event Capture option. The CA SYSVIEW Event Capture option provides a set of tools that enables the capturing of critical data. This data lets you diagnose current or potential problems on your system. Any data that is viewable through a CA SYSVIEW command display can be captured for later redisplay.

The CAPTURE parmlib member allows you to specify the following:

- A capture data set naming convention
- Default data retention time
- Default data set allocation parameters

Individual capture events are specified in CAPLIB members found in the *sysview*.CNM4BCAP data set. Each Capture library (CAPLIB) member can override the default allocation parameters.

CAPTURE Summary

This member is read...

During the initialization of the CAPTURE subtask in the user interface address space. The data is refreshed or reread each time the CAPTURE subtask is started.

Can information be modified in real time?

Each CAPLIB member can override the default allocation parameters.

Subtasks that use this data

CAPTURE

Commands that use this data

CAPTURE, CAPLIST, CAPMAINT

Associated parmlib members

SCHEDULE

Use this member to define interval-based events to capture.

MVSTHRSH, MQSTHRSH, IMSTHRSH, TCPTHRSH

Use these members to define exception-based events to capture.

In user parmlib data set?

No

In Persistent data store?

No

SMFDATA

The CA SYSVIEW SMF data collector lets you view SMF records through the SMFLOG online display. This option is part of the CA SYSVIEW Event Capture option. The SMFDATA parmlib member is used to set configuration options for the SMF data collector.

The SMF data collector performs the following functions:

- Captures SMF records for online viewing
- Suppresses SMF records from the SMF data sets

You can define the capturing or suppression of records by record type and subtype.

SMFDATA Summary

This member is read...

During the initialization of the SMFDATA subtask in the main address space. The data is refreshed or reread each time the SMFDATA subtask is started.

Can information be modified in real time?

Yes, with the SMFDATA command.

Subtasks that use this data

SMFDATA

Commands that use this data

SMFDATA

Associated parmlib members

SMFTYPE

In user parmlib data set?

No

In Persistent data store?

No

SMFTYPE

The SMFTYPE parmlib member is used to associate a text description with an SMF record type and subtype. If the SMFLOG command formats the record, then a record formatting routine can also be associated with the record type and subtype.

SMFTYPE Summary

This member is read…

The first-time the data is requested

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

SMFDATA, SMFLOG

Associated parmlib members

SMFDATA

In user parmlib data set?

No

In Persistent data store?

No

System Condition Monitor Members

SCM

The CA SYSVIEW System Condition Monitor (SCM) component provides a single focal point for problem determination and resolution. The SCM parmlib member defines the information that the System Condition Monitor is to manage.

Included with CA SYSVIEW is the CCS for z/OS product CA GSS. The System Condition Monitor collection process executes as IMODs in the CA GSS address space. IMODs are intelligent modules that are written using a compiled REXX language.

CA SYSVIEW subtasks and online user sessions do not directly use this member.

SCM Summary

This member is read...

During the initialization of the System Condition Monitor in the CA GSS address space. The SCMDRIVE IMOD processes the parmlib information. The data is refreshed or reread each time the System Condition Monitor is started.

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed each time the System Condition Monitor is started.

Subtasks that use this data

None

Commands that use this data

The SCM family of commands

Associated parmlib members

SCMDCOM, SCMDESC, SCMSPACE, SCMSTGRP

In user parmlib data set?

No

In Persistent data store?

No

SCMDCOM

The SCMDCOM parmlib member is input to the System Condition Monitor SCM_DATACOM IMOD. The data in the member is used to define alert values for CA Datacom metrics.

CA SYSVIEW subtasks and online user sessions do not directly use this member.

SCMDCOM Summary

This member is read...

By the System Condition Monitor SCM_DATACOM IMOD

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed each time the System Condition Monitor is started in CA GSS.

Subtasks that use this data

None

Commands that use this data

The SCM family of commands

Associated parmlib members

SCM

In user parmlib data set?

No

In Persistent data store?

No

SCMDESC

The SCMDESC parmlib member is used to associate a text description with a System Condition Monitor summary entry.

CA SYSVIEW subtasks and online user sessions do not directly use this member.

SCMDESC Summary

This member is read...

By the System Condition Monitor SCMDESC IMOD each time a System Condition Monitor summary level entry is created

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed each time the System Condition Monitor is started in CA GSS.

Subtasks that use this data

None

Commands that use this data

The SCM family of commands

Associated parmlib members

SCM

In user parmlib data set?

No

In Persistent data store?

No

SCMSPACE

The SCMSPACE parmlib member is input to the System Condition Monitor SCM_SPACE IMOD. The data in this member is used to define alert or exception values for DASD space usage.

CA SYSVIEW subtasks and online user sessions do not directly use this member.

SCMSPACE Summary

This member is read...

By the System Condition Monitor SCM_SPACE IMOD

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed each time the System Condition Monitor is started in CA GSS.

Subtasks that use this data

None

Commands that use this data

The SCM family of commands

Associated parmlib members

SCM

In user parmlib data set?

No

In Persistent data store?

No

SCMSTGRP

The SCMSTGRP parmlib member is input to the System Condition Monitor SCM_STORGRP IMOD. The data in this member is used to define alert or exception values for SMS storage group space usage.

CA SYSVIEW subtasks and online user sessions do not directly use this member.

SCMSTGRP Summary

This member is read…

By the System Condition Monitor SCM_STORGRP IMOD

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed each time the System Condition Monitor is started in CA GSS.

Subtasks that use this data

None

Commands that use this data

The SCM family of commands

Associated parmlib members

SCM

In user parmlib data set?

No

In Persistent data store?

No

Invoking External Products

XISPBKMG

The XISPBKMG parmlib member is a sample member used by the INVOKE command. This member provides the necessary information and parameters to invoke BookManager Read/MVS from IBM.

XISPBKMG Summary

This member is read...

When the INVOKE command requests the specific member

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

INVOKE

Associated parmlib members

INVOKE

In user parmlib data set?

No

In Persistent data store?

No

XISPDLST

The XISPDLST parmlib member is a sample member used by the INVOKE command. This member provides the necessary information and parameters to invoke the ISPF data set list utility.

XISPDLST Summary

This member is read...

When the INVOKE command requests the specific member

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

INVOKE

Associated parmlib members

INVOKE

In user parmlib data set?

No

In Persistent data store?

No

XISPIMOD

The XISPIMOD parmlib member is a sample member used by the INVOKE command. This member provides the necessary information and parameters to invoke the CA GSS IMOD editor.

XISPIMOD Summary

This member is read...

When the INVOKE command requests the specific member

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

INVOKE

Associated parmlib members

INVOKE

In user parmlib data set?

No

In Persistent data store?

No

XISPQWRF

The XISPQWRF parmlib member is a sample member used by the INVOKE command. This member provides the necessary information and parameters to invoke MVS/Quick-Ref from Chicago-Soft.

XISPQWRF Summary

This member is read...

When the INVOKE command requests the specific member

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

INVOKE

Associated parmlib members

INVOKE

In user parmlib data set?

No

In Persistent data store?

No

XTSORMFM

The XTSORMFM parmlib member is a sample member used by the INVOKE command. This member provides the necessary information and parameters to invoke the IBM RMFMON command.

XTSORMFM Summary

This member is read...

When the INVOKE command requests the specific member

Can information be modified in real time?

No, the information cannot be dynamically altered. The information is refreshed when the user reestablishes the session.

Subtasks that use this data

None

Commands that use this data

INVOKE

Associated parmlib members

INVOKE

In user parmlib data set?

No

In Persistent data store?

No

GMI Members

GSVMACTT

The GSVMACTT parmlib member contains a list of Graphical Management Interface object actions. The contents of this member should not be modified.

GSVMACTT Summary

This member is read...

When the CA SYSVIEW component task of CA Vantage GMI is started during GMI initialization.

Can information be modified in real time?

No

Subtasks that use this data

XSXS

Commands that use this data

Various objects under CA Vantage GMI.

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

GSVMZMDT

The GSVMZMDT parmlib member contains a list of Graphical Management Interface object zoom definitions. The contents of this member should not be modified.

GSVMACTT Summary

This member is read...

When the CA SYSVIEW component task of CA Vantage GMI is started during GMI initialization.

Can information be modified in real time?

No

Subtasks that use this data

XSXS

Commands that use this data

Various objects under CA Vantage GMI.

Associated parmlib members

None

In user parmlib data set?

No

In Persistent data store?

No

Chapter 20: CICS Transaction Variables

This section describes the CICS performance variables.

This section contains the following topics:

Data Fields

Performance class data is detailed transaction-level information, such as the processor and elapsed time for a transaction, or the time spent waiting for I/O. CICS owns the transaction variables in this section. The variables are separated into groups and displayed alphabetically.

The following data fields are available:

Variable

The resource name of the variable

Owner

The resource to which the variable belongs

Group

The name of the group to which the variable belongs

Subgroup

The name of the subgroup to which the variable belongs

Type

The type of variable; such as average, count, or time

Limit

The threshold value for the variable, which has a limit of either upper or lower

Storage

The variable is related to storage when the value is Yes. Storage values are displayed as follows:

K = 1024 bytes

k = 1000 bytes

DFH Group and MCT Number

The associated CICS Monitor Control (MCT) Field and its number

Resource 1

The type of resource.

If blank, then the variable does not support a resource.

Resource 2

The type of resource.

If blank, then the variable does not support a resource.

Description

The description of the variable.

CICS Business Transaction Service

This section displays a list of the variables and their performance data that belong to the CICS business transaction service (CBTS) group.

BAACDCCT Variable

Records the number of CBTS delete, get, or put activity counts.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 217

Resource 1:

Transaction

Resource 2:

Terminal

BAACQPCT Variable

Records the CICS BTS acquire process or acquire activity counts.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 214

Resource 1:

Transaction

Resource 2:

Terminal

BADACTCT Variable

Records the CICS BTS define activity count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 209

Resource 1:

Transaction

Resource 2:

Terminal

BADCPACT Variable

Records the CICS BTS delete activity, cancel activity, or cancel process count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 213

Resource 1:

Transaction

Resource 2:

Terminal

BADFIECT Variable

Records the CICS BTS define input event count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 220

Resource 1:

Transaction

Resource 2:

Terminal

BADPROCT Variable

Records the CICS BTS define process count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 208

Resource 1:

Transaction

Resource 2:

Terminal

BALKPACT Variable

Records the CICS BTS link process or link activity count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 207

Resource 1:

Transaction

Resource 2:

Terminal

BAPRDCCT Variable

Records the CICS BTS delete, get, or put requests for process data containers.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 216

Resource 1:

Transaction

Resource 2:

Terminal

BARASYCT Variable

Records the CICS BTS run process or run activity asynchronous count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 206

Resource 1:

Transaction

Resource 2:

Terminal

BARATECT Variable

Records the CICS BTS retrieve and reattach request count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 219

Resource 1:

Transaction

Resource 2:

Terminal

BARMPACT Variable

Records the CICS BTS resume process or activity count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 212

Resource 1:

Transaction

Resource 2:

Terminal

BARSPACT Variable

Records the CICS BTS reset process or activity count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 210

Resource 1:

Transaction

Resource 2:

Terminal

BARSYNCT Variable

Records the CICS BTS run process or run activity synchronous count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 205

Resource 1:

Transaction

Resource 2:

Terminal

BASUPACT Variable

Records the CICS BTS suspend process or activity count.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 211

Resource 1:

Transaction

Resource 2:

Terminal

BATIAECT Variable

Records the CICS BTS define, check, delete, and force timer event counts.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 221

Resource 1:

Transaction

Resource 2:

Terminal

BATOTCCT Variable

Total CICS BTS delete, get, or put process and activity counts.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 218

Resource 1:

Transaction

Resource 2:

Terminal

BATOTECT Variable

Total CICS BTS event request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 222

Resource 1:

Transaction

Resource 2:

Terminal

BATOTPCT Variable

Total CICS BTS process and activity requests issued.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCBTS, 215

Resource 1:

Transaction

Resource 2:

Terminal

CICS

This section displays a list of the variables and their performance data that belong to the CICS group.

CFCAPICT Variable

Records the CICS OO class request count, which includes Java API for CICS classes.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCICS, 025

Resource 1:

Transaction

Resource 2:

Terminal

EXWTIME Variable

Records the CICS exception wait time as follows:

- The 32-bit clock contains the total elapsed time for which the user waited on exception conditions.
- The 24-bit period count equals the number of exception conditions that have occurred for this task.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCICS, 103

Resource 1:

Transaction

Resource 2:

Terminal

LIFETIME Variable

Records one of the following measurement interval start times:

- The time of the attach.
- The time of the most recent data recording reset.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCICS, 005

Resource 1:

Transaction

Resource 2:

Terminal

PERRECNT Variable

Records the performance class record counts, which the CICS Monitoring Facility (CMF) wrote.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHCICS, 131

Resource 1:

Transaction

Resource 2:

Terminal

DATA

This section displays a list of the variables and their performance data that belong to the DATA group.

DB2CONWT Variable

Records the DB2 connection wait time. This field is the elapsed time the user task waited for the availability of a DB2 connection for use with the user task open TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDATA, 188

Resource 1:

Transaction

Resource 2:

Terminal

DB2RDYQW Variable

Records the DB2 availability wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDATA, 187

Resource 1:

Transaction

Resource 2:

Terminal

DB2REQCT Variable

Records the total DB2 EXEC SQL and Instrumentation Facility Interface (IFI) requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Data

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDATA, 180

Resource 1:

Transaction

Resource 2:

Terminal

DB2WAIT Variable

For CICS connections to DB2 Release 6 or later, this field does not apply and is zero.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDATA, 189

Resource 1:

Transaction

Resource 2:

Terminal

IMSREQCT Variable

Records the IMS (DBCTL) request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

Data

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDATA, 179

Resource 1:

Transaction

Resource 2:

Terminal

IMSWAIT Variable

Records the wait time for DBCTL to service the IMS requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDATA, 186

Resource 1:

Transaction

Resource 2:

Terminal

Destination

This section displays a list of the variables and their performance data that belong to the Destination group.

TDATTIME Variable

Records the VSAM transient data I/O request wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

TDATA

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDEST, 101

Resource 1:

Transaction

Resource 2:

Terminal

TDPGETS Variable

Records the transient data GET requests.

Owner:

CICS

Group:

Transaction

Subgroup:

TDATA

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDEST, 041

Resource 1:

Transaction

Resource 2:

Terminal

TDPPURGE Variable

Records the transient data PURGE requests.

Owner:

CICS

Group:

Transaction

Subgroup:

TDATA

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDEST, 043

Resource 1:

Transaction

Resource 2:

Terminal

TDPPUTS Variable

Records the transient data PUT requests.

Owner:

CICS

Group:

Transaction

Subgroup:

TDATA

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDEST, 042

Resource 1:

Transaction

Resource 2:

Terminal

TDPREQS Variable

Records the total transient data requests, which are the sum of TDGETCT, TDPUTCT, and TDPURCT.

Owner:

CICS

Group:

Transaction

Subgroup:

TDATA

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDEST, 091

Resource 1:

Transaction

Resource 2:

Terminal

Document Handler

This section displays a list of the variables and their performance data that belong to the Document Handler group.

DHCRECT Variable

Records the document handler CREATE request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

DOC

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDOCH, 226

Resource 1:

Transaction

Resource 2:

Terminal

DHINSCT Variable

Records the document handler INSERT request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

DOC

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDOCH, 227

Resource 1:

Transaction

Resource 2:

Terminal

DHRETCT Variable

Records the document handler RETRIEVE request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

DOC

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDOCH, 229

Resource 1:

Transaction

Resource 2:

Terminal

DHSETCT Variable

Records the document handler SET request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

DOC

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDOCH, 228

Resource 1:

Transaction

Resource 2:

Terminal

DHTOTCT Variable

Records the document handler total request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

DOC

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDOCH, 230

Resource 1:

Transaction

Resource 2:

Terminal

DHTOTDCL Variable

Records the total length of all documents.

Owner:

CICS

Group:

Transaction

Subgroup:

DOC

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHDOCH, 240

Resource 1:

Transaction

Resource 2:

Terminal

Enterprise JavaBeans

This section displays a list of the variables and their performance data that belong to the Enterprise JavaBeans (EJBS) group.

EJBCRECT Variable

Records the occurrences of bean creation calls.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHEJBS, 314

Resource 1:

Transaction

Resource 2:

Terminal

EJBMTHCT Variable

Records the executed requests of bean method calls.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHEJBS, 316

Resource 1:

Transaction

Resource 2:

Terminal

EJBREMCT Variable

Records the occurrences of bean removal calls.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHEJBS, 315

Resource 1:

Transaction

Resource 2:

Terminal

EJBSACCT Variable

Records the occurrences of bean activations.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHEJBS, 312

Resource 1:

Transaction

Resource 2:

Terminal

EJBSPACT Variable

Records the occurrences of bean inactivations.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHEJBS, 313

Resource 1:

Transaction

Resource 2:

Terminal

EJBTOTCT Variable

Records the total bean requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHEJBS, 317

Resource 1:

Transaction

Resource 2:

Terminal

Front End Programming Interface

This section displays a list of the variables and their performance data that belong to the Front End Programming Interface (FEPI) group.

SZALLCTO Variable

Records the number of times the user task timed out awaiting allocation.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 157

Resource 1:

Transaction

Resource 2:

Terminal

SZALLOCT Variable

Records the number of FEPI conversions allocated by the user task.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 150

Resource 1:

Transaction

Resource 2:

Terminal

SZCHRIN Variable

Records the number of FEPI characters received.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 155

Resource 1:

Transaction

Resource 2:

Terminal

SZCHROUT Variable

Records the number of FEPI characters sent.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 154

Resource 1:

Transaction

Resource 2:

Terminal

SZRCVCT Variable

Records the number of FEPI receive requests.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 151

Resource 1:

Transaction

Resource 2:

Terminal

SZRCVTO Variable

Records the number of times the task timed out awaiting data.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 158

Resource 1:

Transaction

Resource 2:

Terminal

SZSENDCT Variable

Records the number of FEPI send requests.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 152

Resource 1:

Transaction

Resource 2:

Terminal

SZSTRTCT Variable

Records the number of FEPI start requests.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 153

Resource 1:

Transaction

Resource 2:

Terminal

SZTOTCT Variable

Records the total number of FEPI API and SPI user task requests.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 159

Resource 1:

Transaction

Resource 2:

Terminal

SZWAIT Variable

Records the user task wait time for all FEPI services.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 156

Resource 1:

Transaction

Resource 2:

Terminal

SZWAITCT Variable

Records the FEPI elapsed time count.

Owner:

CICS

Group:

Transaction

Subgroup:

FEPI

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFEPI, 156

Resource 1:

Transaction

Resource 2:

Terminal

File Control

This section displays a list of the variables and their performance data that belong to the File control group.

CFDTWAIT Variable

Records the wait time for a data table access request to the Coupling Facility data table (CFDT) I/O to complete.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 176

Resource 1:

Transaction

Resource 2:

Terminal

FCPADD Variable

Records the number of FCP add requests.

Owner:

CICS

Group:

Transaction

Subgroup:

File

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 039

Resource 1:

Transaction

Resource 2:

Terminal

FCPAMCT Variable

Records the number of FCP access method requests, excluding open and close requests.

Owner:

CICS

Group:

Transaction

Subgroup:

File

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 070

Resource 1:

Transaction

Resource 2:

Terminal

FCPBROWS Variable

Records the number of FCP browse requests, excluding any start and end browse requests.

Owner:

CICS

Group:

Transaction

Subgroup:

FILE

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 038

Resource 1:

Transaction

Resource 2:

Terminal

FCPDEL Variable

Records the number of FCP delete requests.

Owner:

CICS

Group:

Transaction

Subgroup:

FILE

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 040

Resource 1:

Transaction

Resource 2:

Terminal

FCPGETS Variable

Records the number of FCP get requests.

Owner:

CICS

Group:

Transaction

Subgroup:

FILE

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 036

Resource 1:

Transaction

Resource 2:

Terminal

FCPPUTS Variable

Records the number of FCP put requests.

Owner:

CICS

Group:

Transaction

Subgroup:

FILE

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 037

Resource 1:

Transaction

Resource 2:

Terminal

FCPREQS Variable

Records the total number of FCP requests, excluding requests to open, close, enable, or disable a file.

Owner:

CICS

Group:

Transaction

Subgroup:

FILE

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 093

Resource 1:

Transaction

Resource 2:

Terminal

FCPTIME Variable

Records the FCP I/O wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

FILE

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 063

Resource 1:

Transaction

Resource 2:

Terminal

RLSCPUT Variable

Records the service request block (SRB) CPU time this transaction spent processing VSAM record-level sharing (RLS) file requests.

When considering the measurement of the total CPU time consumed by a transaction, add this field to the transaction CPU time field (USRCPUT). Because RLS field requests execute asynchronously under an MVS SRB, it cannot be considered a subset of any other single CMF field.

Note: This clock field could contain a CPU time of zero with a count of greater than zero. This discrepancy occurs because the CMF timing granularity is measured in 16-microsecond units and the RLS file requests can complete in less than that time unit.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 175

Resource 1:

Transaction

Resource 2:

Terminal

RLSWAIT Variable

Records the RLS file I/O wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 174

Resource 1:

Transaction

Resource 2:

Terminal

RLSWAITC Variable

Records the average RLS file I/O wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHFILE, 174

Resource 1:

Transaction

Resource 2:

Terminal

Journal

This section displays a list of the variables and their performance data that belong to the Journal group.

JCPREQS Variable

Records the number of JCP output requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Journal

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHJOUR, 058

Resource 1:

Transaction

Resource 2:

Terminal

JCPTIME Variable

Records the elapsed JCP I/O time.

Owner:

CICS

Group:

Transaction

Subgroup:

Journal

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHJOUR, 010

Resource 1:

Transaction

Resource 2:

Terminal

LOGWRTCT Variable

Records the logger write count.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHJOUR, 172

Resource 1:

Transaction

Resource 2:

Terminal

Basic Mapping Support

This section displays a list of the variables and their performance data that belong to the Basic Mapping Support (BMS) group.

BMSIN Variable

Records the number of BMS input requests.

Owner:

CICS

Group:

Transaction

Subgroup:

BMS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHMAPP, 051

Resource 1:

Transaction

Resource 2:

Terminal

BMSMAP Variable

Records the number of BMS map requests.

Owner:

CICS

Group:

Transaction

Subgroup:

BMS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHMAPP, 050

Resource 1:

Transaction

Resource 2:

Terminal

BMSOUT Variable

Records the number of BMS output requests.

Owner:

CICS

Group:

Transaction

Subgroup:

BMS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHMAPP, 052

Resource 1:

Transaction

Resource 2:

Terminal

BMSREQS Variable

Records the total number of BMS requests. This field is the sum of receive map, receive map from, send map, send text, and send control requests.

Owner:

CICS

Group:

Transaction

Subgroup:

BMS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHMAPP, 090

Resource 1:

Transaction

Resource 2:

Terminal

Program

This section displays a list of the variables and their performance data that belong to the Program group.

PCDPLCT Variable

Records the distributed program link (DPL) counts.

Owner:

CICS

Group:

Transaction

Subgroup:

Program

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHPROG, 073

Resource 1:

Transaction

Resource 2:

Terminal

PCLURMCT Variable

Records the program LINK user-replaceable module (URM) counts. For detailed information about CICS user-replaceable programs, see the *CICS Customization Guide*.

Owner:

CICS

Group:

Transaction

Subgroup:

CBTS

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHPROG, 072

Resource 1:

Transaction

Resource 2:

Terminal

PCPLINK Variable

Records the number of PCP LINK requests, excluding LINK URM requests.

Owner:

CICS

Group:

Transaction

Subgroup:

PROGRAM

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHPROG, 055

Resource 1:

Transaction

Resource 2:

Terminal

PCPLOAD Variable

Records the number of PCP LOAD requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Program

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHPROG, 057

Resource 1:

Transaction

Resource 2:

Terminal

PCPLTIME Variable

Records the user task wait time for the program library fetches. Only program fetches that are automatically installed or installed with program definitions are included. However, because installed programs residing in the LPA do not incur a physical fetch from a library, they are not included.

Owner:

CICS

Group:

Transaction

Subgroup:

Program

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHPROG, 115

Resource 1:

Transaction

Resource 2:

Terminal

PCPXCTL Variable

Records the number of PCP XCTL requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Program

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHPROG, 056

Resource 1:

Transaction

Resource 2:

Terminal

Socket

This section displays a list of the variables and their performance data that belong to the Socket group.

PORTNUM Variable

Records the TCP/IP service port number.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 246

Resource 1:

Transaction

Resource 2:

Terminal

SOBYDECT Variable

Records the bytes decrypted for the secure sockets.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSOCK, 243

Resource 1:

Transaction

Resource 2:

Terminal

SOBYENCT Variable

Records the bytes encrypted for the secure sockets.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSOCK, 242

Resource 1:

Transaction

Resource 2:

Terminal

SOCHRIN Variable

Records the total outbound socket characters received.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSOCK, 295

Resource 1:

Transaction

Resource 2:

Terminal

SOCHRIN1 Variable

Records the inbound socket characters received.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSOCK, 302

Resource 1:

Transaction

Resource 2:

Terminal

SOCHROUT Variable

Records the total outbound socket characters sent.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSOCK, 297

Resource 1:

Transaction

Resource 2:

Terminal

SOCHROU1 Variable

Records the inbound socket characters sent.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSOCK, 304

Resource 1:

Transaction

Resource 2:

Terminal

SOCNPSCT Variable

Records the total create nonpersistent outbound socket requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 290

Resource 1:

Transaction

Resource 2:

Terminal

SOCPSCT Variable

Records the total create persistent outbound socket requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 291

Resource 1:

Transaction

Resource 2:

Terminal

SOEXTRCT Variable

Records the EXTRACT TCPIP and EXTRACT CERTIFICATE request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 289

Resource 1:

Transaction

Resource 2:

Terminal

SOIOWTT Variable

Records the inbound socket I/O wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 241

Resource 1:

Transaction

Resource 2:

Terminal

SOMSGIN1 Variable

Records the inbound socket receive request count.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 301

Resource 1:

Transaction

Resource 2:

Terminal

SOMSGOU1 Variable

Records the inbound socket send request count.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 303

Resource 1:

Transaction

Resource 2:

Terminal

SONPSHWM Variable

Records the nonpersistent high-water mark (HWM).

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 292

Resource 1:

Transaction

Resource 2:

Terminal

SOOIOWTT Variable

Records the outbound socket I/O wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 299

Resource 1:

Transaction

Resource 2:

Terminal

SOPSHWM Variable

Records the persistent outbound socket HWM.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 293

Resource 1:

Transaction

Resource 2:

Terminal

SORCVCT Variable

Records the total outbound persistent and nonpersistent socket receive request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 294

Resource 1:

Transaction

Resource 2:

Terminal

SOSENDCT Variable

Records the total outbound persistent and nonpersistent socket send request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 296

Resource 1:

Transaction

Resource 2:

Terminal

SOTOTCT Variable

Records the total socket request counts.

Owner:

CICS

Group:

Transaction

Subgroup:

Socket

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSOCK, 298

Resource 1:

Transaction

Resource 2:

Terminal

Storage

This section displays a list of the variables and their performance data that belong to the Storage group. The Storage group has fields for user, shared, and program storage.

User Storage Fields

This section contains information about the user storage fields performance data.

CDSAGET Variable

Records the CICS dynamic storage area (CDSA) GETMAIN count.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSTOR, 117

Resource 1:

Transaction

Resource 2:

Terminal

CDSAHWM Variable

Records the HWM of user storage allocated in CDSA.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 116

Resource 1:

Transaction

Resource 2:

Terminal

ECDSAGET Variable

Records the extended CICS dynamic storage area (ECDSA) GETMAIN count.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSTOR, 120

Resource 1:

Transaction

Resource 2:

Terminal

ECDSAHWM Variable

Records the ECDSA HWM above the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 119

Resource 1:

Transaction

Resource 2:

Terminal

STGPGMH Variable

Records the storage occupancy below the 16 MB line in the UDSA. This measures the area under the curve of storage in use against elapsed time.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 095

Resource 1:

Transaction

Resource 2:

Terminal

STGUS24H Variable

Records the SCP user HWM below the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 033

Resource 1:

Transaction

Resource 2:

Terminal

STGUS31H Variable

Records the SCP user HWM above the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 106

Resource 1:

Transaction

Resource 2:

Terminal

STG24GET Variable

Records the SCP user getmain count below the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSTOR, 054

Resource 1:

Transaction

Resource 2:

Terminal

STG31GET Variable

Records the SCP user getmain count above the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSTOR, 105

Resource 1:

Transaction

Resource 2:

Terminal

Shared Storage Fields

This section contains information about the shared storage fields performance data.

SC24FSHR Variable

Records the SDSA bytes freemained below the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 146

Resource 1:

Transaction

Resource 2:

Terminal

SC24GSHR Variable

Records the SDSA bytes getmained below the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 145

Resource 1:

Transaction

Resource 2:

Terminal

SC24SGCT Variable

Records the SDSA getmain count.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSTOR, 145

Resource 1:

Transaction

Resource 2:

Terminal

SC31FSHR Variable

Records the ESDSA bytes freemained above the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 149

Resource 1:

Transaction

Resource 2:

Terminal

SC31GSHR Variable

Records the ESDSA bytes getmained above the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 148

Resource 1:

Transaction

Resource 2:

Terminal

SC31SGCT Variable

Records the ESDSA getmain count above the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSTOR, 147

Resource 1:

Transaction

Resource 2:

Terminal

Program Storage Fields

This section contains information about the program storage fields performance data.

PC24CHWM Variable

Records the program storage HWM below the 16 MB line in CDSA.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 143

Resource 1:

Transaction

Resource 2:

Terminal

PC24RHWM Variable

Records the program storage HWM below the 16 MB line in read-only dynamic storage area (RDSA).

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 162

Resource 1:

Transaction

Resource 2:

Terminal

PC24SHWM Variable

Records the program storage HWM below the 16 MB line in SDSA.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 160

Resource 1:

Transaction

Resource 2:

Terminal

PC31AHWM Variable

Records the program storage HWM above the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 139

Resource 1:

Transaction

Resource 2:

Terminal

PC31CHWM Variable

Records the program storage HWM above the 16 MB line in ECDSA.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 142

Resource 1:

Transaction

Resource 2:

Terminal

PC31RHWM Variable

Records the program storage HWM above the 16 MB line in ERDSA.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 122

Resource 1:

Transaction

Resource 2:

Terminal

PC31SHWM Variable

Records the program storage HWM above the 16 MB line in ESDSA.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 161

Resource 1:

Transaction

Resource 2:

Terminal

STGPGMBH Variable

Records the program storage HWM below the 16 MB line.

Owner:

CICS

Group:

Transaction

Subgroup:

Storage

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHSTOR, 108

Resource 1:

Transaction

Resource 2:

Terminal

Synchronization Point

This section displays a list of the variables and their performance data that belong to the Synchronization Point (synch point) group.

OTSINDWT Variable

Records the object transaction service (OTS) in-doubt wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSYNC, 199

Resource 1:

Transaction

Resource 2:

Terminal

SRVSTWTT Variable

Records the wait time for the Coupling Facility data tables server synch point or resynchronization processing to complete.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSYNC, 177

Resource 1:

Transaction

Resource 2:

Terminal

SYNCDLY Variable

Records the user task wait time for a synch point request to be issued by the parent transaction.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSYNC, 196

Resource 1:

Transaction

Resource 2:

Terminal

SYNCREQS Variable

Records the number of synch point requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSYNC, 060

Resource 1:

Transaction

Resource 2:

Terminal

SYNCTIME Variable

Records the total elapsed time to process synch point requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHSYNC, 173

Resource 1:

Transaction

Resource 2:

Terminal

Task Group

This section displays a list of the variables and their performance data that belong to the Task group.

CHMODECT Variable

Records the total CICS TCB change modes issued.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 248

Resource 1:

Transaction

Resource 2:

Terminal

CPUTIME Variable

Records the CPU time used by each CICS TCB transaction.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 008

Resource 1:

Transaction

Resource 2:

Terminal

DISPTIME Variable

Records the total wait time for tasks to dispatch on each CICS TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 007

Resource 1:

Transaction

Resource 2:

Terminal

DSMMSCWT Variable

Records the TCB unavailable wait time, due to MVS storage constraints.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 279

Resource 1:

Transaction

Resource 2:

Terminal

DSPDELAY Variable

Records the delay for the first dispatch.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 125

Resource 1:

Transaction

Resource 2:

Terminal

DSPDLYCT Variable

Records the dispatch delay time count.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 125

Resource 1:

Transaction

Resource 2:

Terminal

DSTCBMWT Variable

Records the TCB mismatch wait time due to no TCB being available that matches the request.

For transactions that invoke a Java Virtual Machine (JVM) program, this mismatch provides the time spent waiting for a TCB of the correct mode (J8 or J9) and JVM profile.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 268

Resource 1:

Transaction

Resource 2:

Terminal

ENQDELAY Variable

Records the task control local ENQ delay.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 129

Resource 1:

Transaction

Resource 2:

Terminal

ENQDLYCT Variable

Records the task control local ENQ delay count.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 129

Resource 1:

Transaction

Resource 2:

Terminal

GNQDELAY Variable

Records the CICS task control global ENQ delay time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 123

Resource 1:

Transaction

Resource 2:

Terminal

GVUPWAIT Variable

Records the user task wait time caused by another task having control. The user task control changes through these EXEC CICS commands: SUSPEND, CHANGE TASK PRIORITY, CICS POST, and PERFORM RESETTIME.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 184

Resource 1:

Transaction

Resource 2:

Terminal

ICDELAY Variable

Records the interval control delay wait time that resulted from issuing an:

- EXEC CICS DELAY time interval or an expiration time
- EXEC CICS RETRIEVE with the WAIT option

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 183

Resource 1:

Transaction

Resource 2:

ICPREQS Variable

Records the number of start or initiate interval control requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 059

Resource 1:

Transaction

Resource 2:

Terminal

ICTOTCT Variable

Records the total number of start, cancel, delay, and retrieve interval control requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 066

Resource 1:

Transaction

Resource 2:

Terminal

JVMITIME Variable

Records the CICS JVM initialize elapse time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 273

Resource 1:

Transaction

Resource 2:

Terminal

JVMRTIME Variable

Records the CICS JVM reset elapse time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 275

Resource 1:

Transaction

Resource 2:

Terminal

JVMSUSP Variable

Records the CICS dispatcher suspended user task suspend time in the JVM.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 254

Resource 1:

Transaction

Resource 2:

Terminal

JVMTIME Variable

Records the total elapsed time the user task spent in the JVM.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 253

Resource 1:

Transaction

Resource 2:

Terminal

J8CPUT Variable

Records the CPU time of the dispatched user task on a J8 mode TCB.

Note: A transaction invoked Java program defined with EXECKEY=CICS is allocated and uses a CICS J8 mode TCB. Once a task has been allocated a J8 mode TCB, that same TCB remains associated with the task until the Java program completes.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 260

Resource 1:

Transaction

Resource 2:

Terminal

J9CPUPT Variable

Records the CPU time of the dispatched user task on a J9 mode TCB.

Note: A transaction invoked Java program defined with EXECKEY=USER is allocated and uses a CICS J9 mode TCB. Once a task has been allocated a J9 mode TCB, that same TCB remains associated with the task until the Java program completes.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 267

Resource 1:

Transaction

Resource 2:

Terminal

J9CPUT Variable

Records the CPU time of the dispatched user task on a J9 mode TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 267

Resource 1:

Transaction

Resource 2:

Terminal

KY9CPUT Variable

Records the CPU time of the dispatched user task on a Key 9 mode TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 263

Resource 1:

Transaction

Resource 2:

Terminal

KY9CPUTT Variable

Records the CPU time of the dispatched user task on a CICS Key 9 mode TCB.

Note: A transaction invoked Java program defined with EXECKEY=USER is allocated and uses a CICS J9 mode TCB. Once a task is allocated a J9 mode TCB, that same TCB remains associated with the task until the Java program completes.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 265

Resource 1:

Transaction

Resource 2:

Terminal

KY9DISPT Variable

Records the total elapsed CPU time of the dispatched user task on a CICS Key 9 mode TCB.

Note: A transaction invoked Java program defined with EXECKEY=USER is allocated and dispatched as a CICS J9 mode TCB. TCB remains associated with the task until the Java program completes.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 264

Resource 1:

Transaction

Resource 2:

Terminal

LMDELAY Variable

Records the user task wait time to acquire a lock on a resource through the CICS lock manager (LM) domain.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 128

Resource 1:

Transaction

Resource 2:

Terminal

L8CPUT Variable

Records the CPU time of the dispatched user task on a CICS L8 mode TCB.

Note: A task-related user exit program enabled with the OPENAPI option is allocated as a CICS L8 mode TCB. The TCB and task remain associated until the transaction is detached.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 259

Resource 1:

Transaction

Resource 2:

Terminal

MAXHTDLY Variable

Records the time waited to obtain a CICS Hot-pooling TCB (H8 mode), which reached the limit specified through MAXHPTCBS.

Note: HOTPOOL(YES) defined HPJ-compiled Java programs exclusively use the J8 mode open TCBs.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 278

Resource 1:

Transaction

Resource 2:

Terminal

MAXJTDLY Variable

Records the time waited to obtain a CICS JVM TCB (J8 mode), which reached the limit specified through MAXJVMTCBS.

Note: JVM(YES) defined Java programs exclusively use the J8 mode open TCBs.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 277

Resource 1:

Transaction

Resource 2:

Terminal

MAXOTDLY Variable

Records the time waited to obtain a CICS open TCB, which reached the limit specified with CICS MAXOPENTCBS.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 250

Resource 1:

Transaction

Resource 2:

Terminal

MSCPUT Variable

Records the CPU time of dispatched user task MS TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 258

Resource 1:

Transaction

Resource 2:

Terminal

MSDISPT Variable

Records the elapsed user-task MS TCB dispatch time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 257

Resource 1:

Transaction

Resource 2:

Terminal

MXTDELAY Variable

Records the MXT delay time of the first dispatch.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 127

Resource 1:

Transaction

Resource 2:

Terminal

MXTDLYCT Variable

Records the MXT delay count of the first dispatch.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 127

Resource 1:

Transaction

Resource 2:

Terminal

PRIORITY Variable

Records the task priority.

Owner:

CICS

Group:

Transaction

Subgroup:

Priority

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 109

Resource 1:

Transaction

Resource 2:

Terminal

PTPWAIT Variable

Records the 3270 bridge partner transaction wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 285

Resource 1:

Transaction

Resource 2:

Terminal

QRCPUT Variable

Records the CPU time for the dispatched user task on the CICS QR TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 256

Resource 1:

Transaction

Resource 2:

Terminal

QRDISPT Variable

Records the elapsed time for the dispatched user task on the CICS QR TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 255

Resource 1:

Transaction

Resource 2:

Terminal

QRMODDLY Variable

Records the elapsed time of the user task QR TCB wait-for-dispatch.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 249

Resource 1:

Transaction

Resource 2:

Terminal

RMICOUNT Variable

Records the total elapsed time spent in the resource manager interface (RMI).

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 170

Resource 1:

Transaction

Resource 2:

Terminal

RMISUSP Variable

Records the total elapsed time of a dispatcher suspended task while in RMI suspend time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 171

Resource 1:

Transaction

Resource 2:

Terminal

RMISUSPC Variable

Records the total RMI suspend count.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 171

Resource 1:

Transaction

Resource 2:

Terminal

RMITIME Variable

Records the RMI total elapsed time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 170

Resource 1:

Transaction

Resource 2:

Terminal

ROCPUT Variable

Records the CPU time of the dispatched user task on the RO mode TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 270

Resource 1:

Transaction

Resource 2:

Terminal

RODISPT Variable

Records the elapsed time of the dispatched user task on the RO mode TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 269

Resource 1:

Transaction

Resource 2:

Terminal

RQPWAIT Variable

Records the request processor user task wait time for outstanding replies to satisfy.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 193

Resource 1:

Transaction

Resource 2:

Terminal

RQRWAIT Variable

Records the request receiver wait time for outstanding replies to satisfy.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 192

Resource 1:

Transaction

Resource 2:

Terminal

RRMSWAIT Variable

Records the RRMS/MVS wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 191

Resource 1:

Transaction

Resource 2:

Terminal

RUNTRWTT Variable

Records the completion wait time of a CICS BTS run process or activity synchronously request.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 195

Resource 1:

Transaction

Resource 2:

Terminal

S8CPUT Variable

Records the CPU time of the dispatched user task on a CICS S8 mode TCB.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 261

Resource 1:

Transaction

Resource 2:

Terminal

SUSPTIME Variable

Records the total elapsed user task suspend time, which includes:

- Elapsed time waiting for the first dispatch
- Task suspended time
- Elapsed redispatch wait time after resuming a suspended task

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 014

Resource 1:

Transaction

Resource 2:

Terminal

TCBATTCT Variable

Records the CICS TCB attach count.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 251

Resource 1:

Transaction

Resource 2:

Terminal

TCLDELAY Variable

Records the transaction class delay time, due to the TCLSNAME limits, for the first dispatch.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 126

Resource 1:

Transaction

Resource 2:

Terminal

TCLDLYCT Variable

Records the wait time, due to transaction class limits, for the first dispatch.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 126

Resource 1:

Transaction

Resource 2:

Terminal

WTCEWAIT Variable

Records the elapsed time the user task waited for:

- One or more ECBs to be MVS POSTed using EXEC CICS WAITCICS.
- Completion of a user task initiated event. To give up control directly to some other task until the event being waited on is completed, use the EXEC CICS WAIT EVENT command.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 182

Resource 1:

Transaction

Resource 2:

Terminal

WTEXWAIT Variable

Records the wait time for posting one or more ECBs that the user task passed to CICS using the EXEC CICS WAIT EXTERNAL ECBLIST command.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 181

Resource 1:

Transaction

Resource 2:

Terminal

WTRTIME Variable

Records the waiting to run time.

Note: This field does not include the elapsed time spent waiting for first dispatch.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTASK, 102

Resource 1:

Transaction

Resource 2:

Terminal

Temporary Storage

The section displays a list of the variables and their performance data that belong to the Temporary Storage group.

TSPGETS Variable

Records the temporary storage GET requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Tempstor

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTEMP, 044

Resource 1:

Transaction

Resource 2:

Terminal

TSPPUTSA Variable

Records the temporary storage PUT auxiliary requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Tempstor

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTEMP, 046

Resource 1:

Transaction

Resource 2:

Terminal

TSPPUTSM Variable

Records the temporary storage PUT main requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Tempstor

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTEMP, 047

Resource 1:

Transaction

Resource 2:

Terminal

TSPREQS Variable

Records the total number of temporary storage requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Tempstor

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTEMP, 092

Resource 1:

Transaction

Resource 2:

Terminal

TSPTIME Variable

Records the TSP I/O wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Tempstor

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTEMP, 011

Resource 1:

Transaction

Resource 2:

Terminal

TSSHWAIT Variable

Records the wait time for an asynchronous shared temporary storage I/O to complete.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTEMP, 178

Resource 1:

Transaction

Resource 2:

Terminal

Terminal

This section displays a list of the variables and their performance data that belong to the Terminal group.

LU61WTT Variable

Records the time a user task waited for an I/O on the LU6.1 session.

Note: This time does not include the waits incurred due to LU6.1 synch point flows.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 133

Resource 1:

Transaction

Resource 2:

Terminal

LU61WTTC Variable

Records the time a user task waited for an I/O on the LU6.1 session.

Note: This wait time does not include the waits incurred due to LU6.1 synch point flows.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 133

Resource 1:

Transaction

Resource 2:

Terminal

LU62WTT Variable

Records the time a user task waited for an I/O on the LU6.2 (APPC) session.

Note: This wait time does not include the waits incurred due to LU6.2 synch point flows.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 134

Resource 1:

Transaction

Resource 2:

Terminal

LU62WTTC Variable

Records the time a user task waited for an I/O on the LU6.2 (APPC) session.

Note: This wait time does not include the waits incurred due to LU6.2 synch point flows.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 134

Resource 1:

Transaction

Resource 2:

Terminal

MROTIME Variable

Records the multiregion operation (MRO) I/O wait time.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 100

Resource 1:

Transaction

Resource 2:

Terminal

TCALLOCT Variable

Records the TCP TCTTE allocate requests.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 069

Resource 1:

Transaction

Resource 2:

Terminal

TCCHRIN1 Variable

Records the TCP bytes input from the task primary terminal facility.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHTERM, 083

Resource 1:

Transaction

Resource 2:

Terminal

TCCHRIN2 Variable

Records the TCP bytes input from LU6.1.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHTERM, 085

Resource 1:

Transaction

Resource 2:

Terminal

TCCHROU1 Variable

Records the TCP bytes output from the task primary terminal facility.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHTERM, 084

Resource 1:

Transaction

Resource 2:

Terminal

TCCHROU2 Variable

Records the TCP bytes output to LU6.1.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFHTERM, 086

Resource 1:

Transaction

Resource 2:

Terminal

TCC62IN2 Variable

Records the LU6.2 characters received.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 137

Resource 1:

Transaction

Resource 2:

Terminal

TCC62OU2 Variable

Records the LU6.2 characters sent.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 138

Resource 1:

Transaction

Resource 2:

Terminal

TCMSGIN1 Variable

Records the TCP messages input from the primary terminal facility.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 034

Resource 1:

Transaction

Resource 2:

Terminal

TCMSGIN2 Variable

Records the TCP messages received from LU6.1.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 067

Resource 1:

Transaction

Resource 2:

Terminal

TCMSGOU1 Variable

Records the TCP messages sent to the primary terminal facility.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 035

Resource 1:

Transaction

Resource 2:

Terminal

TCMSGOU2 Variable

Records the TCP messages output to LU6.1.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 068

Resource 1:

Transaction

Resource 2:

Terminal

TCM62IN2 Variable

Records the LU6.2 messages received.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 135

Resource 1:

Transaction

Resource 2:

Terminal

TCM62OU2 Variable

Records the LU6.2 messages sent.

Owner:

CICS

Group:

Transaction

Subgroup:

Count

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 136

Resource 1:

Transaction

Resource 2:

Terminal

TERMTIME Variable

Records the elapsed terminal I/O wait time after issuing a receive request.

Owner:

CICS

Group:

Transaction

Subgroup:

Clock

Type:

Time

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFHTERM, 009

Resource 1:

Transaction

Resource 2:

Terminal

Web Support

This section displays a list of the variables and their performance data that belong to the web support group.

WBBRWCT Variable

Records the web browse request count.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFWEBB, 239

Resource 1:

Transaction

Resource 2:

Terminal

WBCHRIN Variable

Records the web characters received.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFWEBB, 232

Resource 1:

Transaction

Resource 2:

Terminal

WBCHROUT Variable

Records the web characters sent.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

Yes

Associated CICS Variable MCT Field and Number:

DFWEBB, 234

Resource 1:

Transaction

Resource 2:

Terminal

WBEXTRCT Variable

Records the web extract request count.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFWEBB, 238

Resource 1:

Transaction

Resource 2:

Terminal

WBRCVCT Variable

Records the web receive request count.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFWEBB, 231

Resource 1:

Transaction

Resource 2:

Terminal

WBREADCT Variable

Records the web read request count.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFWEBB, 224

Resource 1:

Transaction

Resource 2:

Terminal

WBREPRCT Variable

Records the web repository read request count.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFWEBB, 236

Resource 1:

Transaction

Resource 2:

Terminal

WBREPWCT Variable

Records the web repository write request count.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFWEBB, 237

Resource 1:

Transaction

Resource 2:

Terminal

WBSENDCT Variable

Records the web send request count.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFWEBB, 233

Resource 1:

Transaction

Resource 2:

Terminal

WBTOTCT Variable

Records the web total request count.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFWEBB, 235

Resource 1:

Transaction

Resource 2:

Terminal

WBWRITCT Variable

Records the web write request count.

Owner:

CICS

Group:

Transaction

Subgroup:

WEB

Type:

Average

Limit:

Upper

Storage:

No

Associated CICS Variable MCT Field and Number:

DFWEBB, 225

Resource 1:

Transaction

Resource 2:

Terminal

Appendix A: Parameter Library Keywords

This section describes the keywords that are available when modifying the parameter library members.

This section contains the following topics:

Using the Keywords

The keywords described in this appendix enable you to specify different characteristics for one CA SYSVIEW parmlib member.

Use these keywords in any member of the CA SYSVIEW parmlib library, unless you are instructed otherwise.

Keywords

You can use the following keywords within parmlib members to direct CA SYSVIEW. All keywords must start in column 1.

)INCLUDE *member* <*dataset-name*> <MSG|NOMSG> <DSNREQ>

Includes parmlib statements from the specified parmlib member.

<dataset-name>

Parmlib members from an alternate data set can be included by specifying an optional data set name.

MSG|NOMSG

Indicates whether to include an error message when the member is not found.

DSNREQ

Indicates that the data set must be present, which is useful when using a substitution variable for dataset-name. If the substituted data set name is blank, the statement would appear as a normal member include with no data set. This parameter causes the member include to be skipped in this circumstance.

)IF SYSNAME=*sysname*

)IF SYSNAME¬=*sysname*

Include the lines that follow if the current z/OS system name matches or does not match the specified name.

)IF FEATURE=HCHECK

Is evaluated as yes if you specified Yes for the Component-Health-Check keyword in the System Configuration Options member.

)IF USERID=*userid*

)IF USERID¬=*userid*

Include the lines that follow if the user ID of the CA SYSVIEW user matches or does not match the specified user ID.

)IF GROUP=*security-group*

Include the lines that follow if the security-group matches the specified security-group.

)IF INTERACTIVE

Interactive product interface

Examples: TSO, VTAM, CICS, ISPF

)IF NONINTERACTIVE

Noninteractive product interface

Example: Batch and API

)IF SMFID=*smfid*

)IF SMFID¬=*smfid*

Include the lines that follow if the current z/OS SMF ID matches or does not match the specified SMF ID.

)IF SSID=*ssid*

)IF SSID¬=*ssid*

Include the lines that follow if the CA SYSVIEW subsystem ID matches or does not match the specified subsystem ID.

)IF SYSPLEX=*sysplex*

Include the lines that follow if the current sysplex name matches or does not match the specified sysplex name.

)IF JOBNAME=*jobname*

)IF JOBNAME¬=*jobname*

Include the lines that follow if the CA SYSVIEW home address space matches or does not match the home address space that is specified.

)IF INTERFACE=*interface*

)IF INTERFACE¬=*interface*

Include the lines that follow if the CA SYSVIEW interface matches or does not match the specified interface. For a list of interfaces, see Interfaces (see page 517) in this appendix.

)IF TASKID=*taskid*

)IF TASKID¬=*taskid*

Include the lines that follow if the CA SYSVIEW task ID matches or does not match the specified task ID.

)IF CICSLOGR=*cicslogr*

)IF CICSLOGR¬=*cicslogr*

Include the lines that follow if the CICS logger task ID matches or does not match the specified CICS logger task ID.

)IF IMSLOGR=imslogr

)IF IMSLOGR¬=imslogr

Include the lines that follow if the IMS logger task ID matches or does not match the specified IMS logger task ID.

)IF DAY=*day*

)IF DAY¬=*day*

Include the lines that follow if the day of the week matches or does not match the specified day.

)IF MONTH=*month*

)IF MONTH¬=*month*

Include the lines that follow if the month of the year matches or does not match the specified month.

)IF JOBACTIVE=*jobname*

)IF ACTIVE=*jobname*

)IF JOBINACTIVE=*jobname*

)IF INACTIVE=*jobname*

Include the lines that follow if the active or inactive job name matches the specified jobname/STC/TSU.

)IF QMGR=*qmgr*

Include the lines that follow if the WebSphere MQ Queue Manager (QMGR) matches.

)IF CICS=*vr*

)IF CICS¬=*vr*

Include the lines that follow if the current CICS release matches or does not match the release that you specify. The following are values for *vr* and the CICS versions that they stand for:

vr	CICS Release
6+	6.x.x
60	6.x.x
62	6.2.x
63	6.3.x
64	6.4.x
65	6.5.x

)IF SPLEVEL=SP*v.r.m*

)IF SPLEVEL¬=SP*v.r.m*

)IF SPLEVEL>=SP*v.r.m*

)IF SPLEVEL<=SP*v.r.m*

Include the lines that follow if the z/OS operating system SPLEVEL matches, or does not match, is higher than, or lower than the version, release, and modification that you specify.

)ENDIF

End the prior IF statement.

)SKIP

)ENDSKIP

Provide dynamic skipping of entries. These keywords allow you to skip or bypass the included entries without marking each line as a comment.

)END

)EOF

Process this record as the logical end of file.

Interfaces

The following table lists the CA SYSVIEW interfaces. You can use these interfaces in place of the *interface* variable in the following statement:

```
)IF INTERFACE
```

Online	Tasks	Utilities
API	MAIN	INDXCLUP
BATCH	AMSTASK	INDXINIT
CICS	APPLMON	PROFCONV
ISPF	CICSLOGR	SECUCONV
LCL3270D	IMSDATA	SECUREPT
ROSCOE	JOBTASK	SRCHUTIL
TSO	LOGTASK	
VTAM	MQSDATA	
	MVSDATA	
	SCHEDULR	
	SMFDATA	
	TCPDATA	
	CAPTCNTL	
	CICSDRVR	
	VTAMDRVR	

Appendix B: Product Variables

This section describes the use of product variables in parmlib members.

Variables in PARMLIB Members

You can use product variables in parmlib members. The variable name must be preceded with an '&' for variable substitution to occur.

Important: If a system symbol is defined with the same name as a product variable, the product variable value is used.

The following table lists the names and values of the product variable.

Variable Name	Value
RESUNIT	<unit type of the RES volume>
RESDEVN	<device number of the RES volume>
RESVOL	<VOLSER of the RES volume>
IPLTYPE	<COLD|WARM|QUICK>
IPLTIME	<IPL time>
IPLDATE	<IPL date>
SYSDATE	<system date>
SYSTIME	<system time>
PRIMASID	<ASID number of the primary CA SYSVIEW address space>
PRIMJOBN	<job name of the primary CA SYSVIEW address space>
SPLEVEL	<z/OS SP level>
OSLEVEL	<z/OS level>
OSNAME	z/OS
USERID	<user ID>
ProfileID	<profile ID>
SMFID	<SMF name>
SYSNAME	<System name>
G$BUILD	<product build number>

Variable Name	Value
G$VERSN	<product version number>
G$PRM	<product release number (no decimals)>
G$UniqId	<subsystem ID + release number>
JESNODE	<JES node name>
G$Interface	<API|BATCH|CICS|ISPF|LCL3270D|ROSCOE|TSO|VTAM|XSSI|XSXI>
G$RELSE	<product release number>
G$FMID	<SMP/E FMID>
G$SSID	<subsystem ID>
G$XNAME	CA SYSVIEW Performance Management

Appendix C: Product Abend Codes

This section describes the abend and reason codes.

User Abend and Reason Codes

The following list provides the descriptions for the hexadecimal and decimal abend and reason codes.

- Get register save area errors

Hex Reason	Decimal Reason	Description
101	257	The register save area stack is empty. There are no more register save areas available.

- Put register save area errors

Hex Reason	Decimal Reason	Description
201	513	The register save area stack is corrupt.

- Free storage errors

Hex Reason	Decimal Reason	Description
301	769	The storage chain is corrupt.
302	770	The active command was not matched.
303	771	The temp storage stack is empty.
304	772	The caller was not in key 8.
305	773	The address to be freed was 0.
306	774	A request to free storage above 2G was made before memory object initialization was completed.
307	775	A request to free storage above 2G was made. The element was found on the allocated list.
308	776	The STORAGE RELEASE or FREEMAIN macro failed.

Hex Reason	Decimal Reason	Description
309	777	A request to free storage found that the owning TCB address was not matched.
30A	778	A request to free storage above 2G found that the owning TCB address was not matched.

- Get storage errors

Hex Reason	Decimal Reason	Description
401	1025	The requested length is less than the minimum.
402	1026	The requested length is greater than the maximum.
403	1027	The STORAGE OBTAIN or GETMAIN macro failed.
404	1028	The caller was not in key 8.
405	1029	A request to get storage above 2G was made before memory object initialization was completed.
406	1030	An unknown error occurred attempting to obtain GPVT storage.
407	1031	An obtain GPVT storage request failed because there was not enough storage available to satisfy the request.
408	1032	An obtain GPVT storage request failed because the requested length was less than the minimum allowed.
409	1033	An obtain GPVT storage request failed because the requested length was greater than the maximum allowed.
40A	1034	An obtain GPVT storage request failed because the IBM "IARV64 REQUEST=GETSTOR" macro failed.
40B	1035	An obtain GPVT storage request failed because the GPVT storage limit was exceeded.

- Job summary task errors

Hex Reason	Decimal Reason	Description
501	1281	No jobs found by the job summary task.

- Storage truncation errors

Hex Reason	Decimal Reason	Description
601	1537	The storage chain is corrupt.
602	1538	The requested length is less than the minimum.
603	1539	The requested length is greater than the maximum.
604	1540	The caller was not in key 8.

- Storage chain verification errors

Hex Reason	Decimal Reason	Description
801	2049	Temporary storage chain is corrupt.
802	2050	Permanent storage chain is corrupt.
803	2051	Grande control area is corrupt.
804	2052	Grande allocated storage chain is corrupt.
805	2053	Grande free storage chain is corrupt.

- Module release and build check errors

Hex Reason	Decimal Reason	Description
901	2305	The release of the interface driver module does not match the release found in the common data area module.
902	2306	The release of the nucleus module does not match the release found in the common data area module.
903	2307	Some other module release does not match the release found in the common data area module.

- Modify storage attribute errors

Hex Reason	Decimal Reason	Description
A01	2561	The storage prefix block does not contain the correct identifier.
A02	2562	The grande storage prefix block does not contain the correct identifier.
A04	2564	The caller was not in key 8.

- Architectural level check errors

Hex Reason	Decimal Reason	Description
B01	2817	The architectural level is not z/Architecture or higher. This reason will always use the default user abend code.

- Recursion errors

Hex Reason	Decimal Reason	Description
C01	3073	Recursive entry in the storage get/free service module.

- Storage query errors

Hex Reason	Decimal Reason	Description
D01	3329	The storage prefix block does not contain the correct identifier.
D02	3330	The grande storage prefix block does not contain the correct identifier.

- Health check exit errors

Hex Reason	Decimal Reason	Description
1001	4097	The HCE address was not valid.
1002	4098	The HCE identifier was not valid.
1003	4099	The HCE event was not valid.
1004	4100	The HCE code was not valid.
1005	4101	The HCDE address was not valid.

Hex Reason	Decimal Reason	Description
1006	4102	The HCDE identifier was not valid.
1007	4103	The check module elapsed time limit exceeded.
1008	4104	The check module CPU time limit exceeded.
1009	4105	The HCE INIT code was received when not expected.
100A	4106	The HCE RUN code was received when not expected.
100B	4107	The HCE TERM code was received when not expected.

- CPU time limit exceeded errors

Hex Reason	Decimal Reason	Description
1101		The CPU time limit exceeded was detected by the standard CPU monitoring routine. This limit is defined by the CpuTimeLimit parameter in the OPTIONS parmlib member.
1102		The CPU time limit exceeded was detected by AMBLIST interface module. This limit is defined by the UtilTaskAmbListMaxCpu parameter in the OPTIONS parmlib member.

- zIIP enablement service errors

Hex Reason	Decimal Reason	Description
1201	4609	The zIIP index stack went negative.
1202	4610	The zIIP index stack was full.

- Diagnostic and testing errors

Hex Reason	Decimal Reason	Description
1301	4865	An abend was forced in a module linkedited into the nucleus module.

Hex Reason	Decimal Reason	Description
1302	4866	An abend was forced in a REXX EXEC with the REXX ABEND() function.
1303	4867	An invalid ESR (ExtendedServiceRouter) code was passed to an ESR service module.

- Capture interface errors

Hex Reason	Decimal Reason	Description
1401	5121	The capture interface send screen routine was entered unexpectedly.
1402	5122	The capture interface get response routine was entered unexpectedly.

- Escape function errors

Hex Reason	Decimal Reason	Description
1501	5377	The escape function could not be executed because the required escape address was not available.

- Early nucleus initial program load errors

Hex Reason	Decimal Reason	Description
1601	5633	Invalid module identifier found. This reason will always use the default user abend code. The following 64-bit register contents are defined for this reason: R2 - The module identification area address. R5 - The module name.
1602	5634	Invalid module release or build found. This reason will always use the default user abend code. The following 64-bit register contents are defined for this reason: R2 - The module identification area address. R3 - The module release. R4 - The module build.

Appendix D: Integration with Other CA Products

This section contains information on sending event notifications.

This section contains the following topics:

Integration with CA OPS/MVS Event Notification

When an exception alert within CA SYSVIEW is triggered based on a defined threshold or state rule, multiple actions can be taken. One action is to send an event notification to CA OPS/MVS.

Traditionally, CA SYSVIEW sent information about events to CA OPS/MVS through console messages. The)MSG rules process console messages. Using the event notification action can reduce the number of console messages and the overhead associated with processing the messages.

Enabling Event Notification

The action to generate an event notification is specified on each threshold or state rule definition. This notification can be coded as a default for all rules or specifically on each definition in the respective threshold and state definition parmlib members.

The)API type rules process the events.

Set Configuration Options

The configuration options enable the event notification process. This procedure explains how you set the configuration options.

Follow these steps:

1. In CA OPS/MVS, set the APIACTIVE parameter to ON.

   This parameter *must* be set for the interface to work.

2. In CA SYSVIEW, set the following options in the configuration parmlib members:

   ***sysview*.CNM4BPRM(SYSDATA)**

   This option setting enables sending event notification of MVS, WebSphere MQ, IMS, and TCP/IP exception data to CA OPS/MVS.

   Configuration Option: OPSMVS-EVENT-NOTIFICATION

   Value: Yes

   ***sysview*.CNM4BPRM(CICSOPTS)**

   This option setting enables sending event notification of CICS exception data to CA OPS/MVS.

   Configuration Option: OPSMVS-EVENT-NOTIFICATION

   Value: Yes

   The configuration options are set.

API Rules

For each reportable event, the interface triggers an)API rule, and supplies data to the rule in the form of environmental variables. Each type of event has an associated set of variable names.

Important! All of the variables are a data type of character except the API.COLOR variable. The API.COLOR variable value is binary. All of the variables are read-only. An attempt to store into these variables results in program failure.

More information:

Using the Application Programming Interface (see page 109)

Example: MVS Event Notification

This example format shows the first line of an MVS event notification API rule.

```
)API CAGSVX0001
```

The available REXX variables for MVS event notification are:

REXX Variable	Value
API.APPLICATION	SYSVIEW
API.VERSION	Current release
API.LEVEL	Service Pack/Build
API.ID	CAGSVX0001
API.COLOR	Alert color ■ x'00' - Default ■ x'01' - Green ■ x'02' - Blue ■ x'03' - Red ■ x'04' - White ■ x'05' - Pink ■ x'06' - Yellow ■ x'07' - Turquoise
API.GROUP	Variable metric group name
API.NAME	Variable metric name
API.DESCRIPTION	Variable metric description
API.RESOURCE	Associated resource
API.STATUS	Current status
API.VALUE	Current value
API.ELAPSED	Elapsed time since the last notification
API.ASID	Provides the ASID
API.JOBID	Provides the JES job ID
API.TEXT	Exception message text

API.TEXT has the following format:

Parameter	Contents	Max Length
01	System name	8
02	Subsystem - 'MVS'	8

Parameter	Contents	Max Length
03	Jobname	8
04	Metric group	8
05	Metric subgroup	8
06	Metric name	8
07	Resource	8
08	Old metric status	8
09	New metric status	8
10	Elapsed time since notification	8
11	Metric value	12
12	Exception warning limit	8
13	Exception problem limit	8
14	Exception rule type	6

Example: CICS Event Notification

This example format shows the first line of a CICS event notification API rule.

```
)API CAGSVY0001
```

The available REXX variables for CICS event notification are:

REXX Variable	Value
API.APPLICATION	SYSVIEW
API.VERSION	Current release
API.LEVEL	Service Pack/Build
API.ID	CAGSVY0001

REXX Variable	Value
API.COLOR	Alert color ■ x'00' - Default ■ x'01' - Green ■ x'02' - Blue ■ x'03' - Red ■ x'04' - White ■ x'05' - Pink ■ x'06' - Yellow ■ x'07' - Turquoise
API.GROUP	Variable metric group name
API.NAME	Variable metric name
API.DESCRIPTION	Variable metric description
API.RSCE1	Associated resource
API.RSCE2	Associated resource
API.STATUS	Current status
API.VALUE	Current value
API.ELAPSED	Elapsed time after notification
API.TEXT	Exception message text

API.TEXT has the following format:

Parameter	Contents	Max Length
01	Exception type	8
02	Metric group	8
03	Metric name	8
04	Resource 1	8
05	Resource 2	8
06	Old metric status	8
07	New metric status	8
08	'V ='	2
09	Metric value	12

Parameter	Contents	Max Length
10	'W='	2
11	Exception warning limit	8
12	'P='	2
13	Exception problem limit	8
14	Exception rule type	6
15	Elapsed time since notification	8
16	CICS jobname	8
17	Transaction ID	4
18	Transaction number	8
19	Terminal ID	4
20	User ID	8

Example: IMS Event Notification

This example format shows the first line of an IMS event notification API rule.

```
)API CAGSVP0001
```

The available REXX variables for IMS event notification are:

REXX Variable	Value
API.APPLICATION	SYSVIEW
API.VERSION	Current release
API.LEVEL	Service Pack/Build
API.ID	CAGSVP0001

REXX Variable	Value
API.COLOR	Alert color ■ x'00' - Default ■ x'01' - Green ■ x'02' - Blue ■ x'03' - Red ■ x'04' - White ■ x'05' - Pink ■ x'06' - Yellow ■ x'07' - Turquoise
API.GROUP	Variable metric group name
API.NAME	Variable metric name
API.DESCRIPTION	Variable metric description
API.IMSID	IMS subsystem ID
API.RESOURCE	Associated resource
API.STATUS	Current status
API.VALUE	Current value
API.ELAPSED	Elapsed time after notification
API.TEXT	Exception message text

API.TEXT has the following format:

Parameter	Contents	Max Length
01	System name	8
02	Subsystem - 'IMS'	8
03	IMS subsystem ID	4
04	Metric group	8
05	Metric subgroup	8
06	Metric name	8
07	Resource	8
08	Old metric status	8
09	New metric status	8

Parameter	Contents	Max Length
10	Elapsed time since notification	8
11	Metric value	12
12	Exception warning limit	8
13	Exception problem limit	8
14	Exception rule type	6

Example: TCP/IP Event Notification

This example format shows the first line of a TCP/IP event notification API rule.

```
)API CAGSVN0001
```

The available REXX variables for TCP/IP event notification are:

REXX Variable	Value
API.APPLICATION	SYSVIEW
API.VERSION	Current release
API.LEVEL	Service Pack/Build
API.ID	CAGSVN0001
API.COLOR	Alert color ■ 'x00' - Default ■ 'x01' - Green ■ 'x02' - Blue ■ 'x03' - Red ■ 'x04' - White ■ 'x05' - Pink ■ 'x06' - Yellow ■ 'x07' - Turquoise
API.GROUP	Variable metric group name
API.NAME	Variable metric name
API.DESCRIPTION	Variable metric description
API.TCPID	TCP/IP jobname of stack
API.RESOURCE	Associated resource

REXX Variable	Value
API.STATUS	Current status
API.VALUE	Current value
API.ELAPSED	Elapsed time after notification
API.TEXT	Exception message text

API.TEXT has the following format:

Parameter	Contents	Max Length
01	System name	8
02	Subsystem - 'TCP'	8
03	TCP/IP jobname of stack	8
04	Metric group	8
05	Metric subgroup	8
06	Metric name	8
07	Resource	8
08	Old metric status	8
09	New metric status	8
10	Elapsed time since notification	8
11	Metric value	12
12	Exception warning limit	8
13	Exception problem limit	8
14	Exception rule type	6

Example: WebSphere MQ Event Notification

This example format shows the first line of a WebSphere MQ event notification API rule.

```
)API CAGSVS0001
```

The available REXX variables for WebSphere MQ event notification are:

REXX Variable	Value
API.APPLICATION	SYSVIEW

REXX Variable	Value
API.VERSION	Current release
API.LEVEL	Service Pack/Build
API.ID	CAGSVS0001
API.COLOR	Alert color ■ x'00' - Default ■ x'01' - Green ■ x'02' - Blue ■ x'03' - Red ■ x'04' - White ■ x'05' - Pink ■ x'06' - Yellow ■ x'07' - Turquoise
API.GROUP	Variable metric group name
API.NAME	Variable metric name
API.DESCRIPTION	Variable metric description
API.QMGR	Queue manager
API.RESOURCE	Associated resource
API.STATUS	Current status
API.VALUE	Current value
API.ELAPSED	Elapsed time after notification
API.TEXT	Exception message text

API.TEXT has the following format:

Parameter	Contents	Max Length
01	System name	8
02	Subsystem - 'MQSeries'	8
03	Queue manager	4
04	Metric group	8
05	Metric subgroup	8
06	Metric name	8

Parameter	Contents	Max Length
07	Resource	64
08	Old metric status	8
09	New metric status	8
10	Elapsed time since notification	8
11	Metric value	12
12	Exception warning limit	8
13	Exception problem limit	8
14	Exception rule type	6

Integration with CA Service Desk

CA SERVICE DESK Integration for z/OS (CAISDI) is a part of CA Common Services used by CA SYSVIEW for interfacing with CA SERVICE DESK. This integration lets CA SYSVIEW automatically open CA SERVICE DESK requests for unexpected product ABENDs. This integration provides your organization with an instantly recorded notification of the problem. This notification lets you quickly address the problem before causing more serious problems with CA SYSVIEW that could impact your ability to monitor system performance efficiently.

For information about customizing CAISDI to integrate with CA SYSVIEW, see the CA Common Services for z/OS CA Service Desk *Integration Guide*.

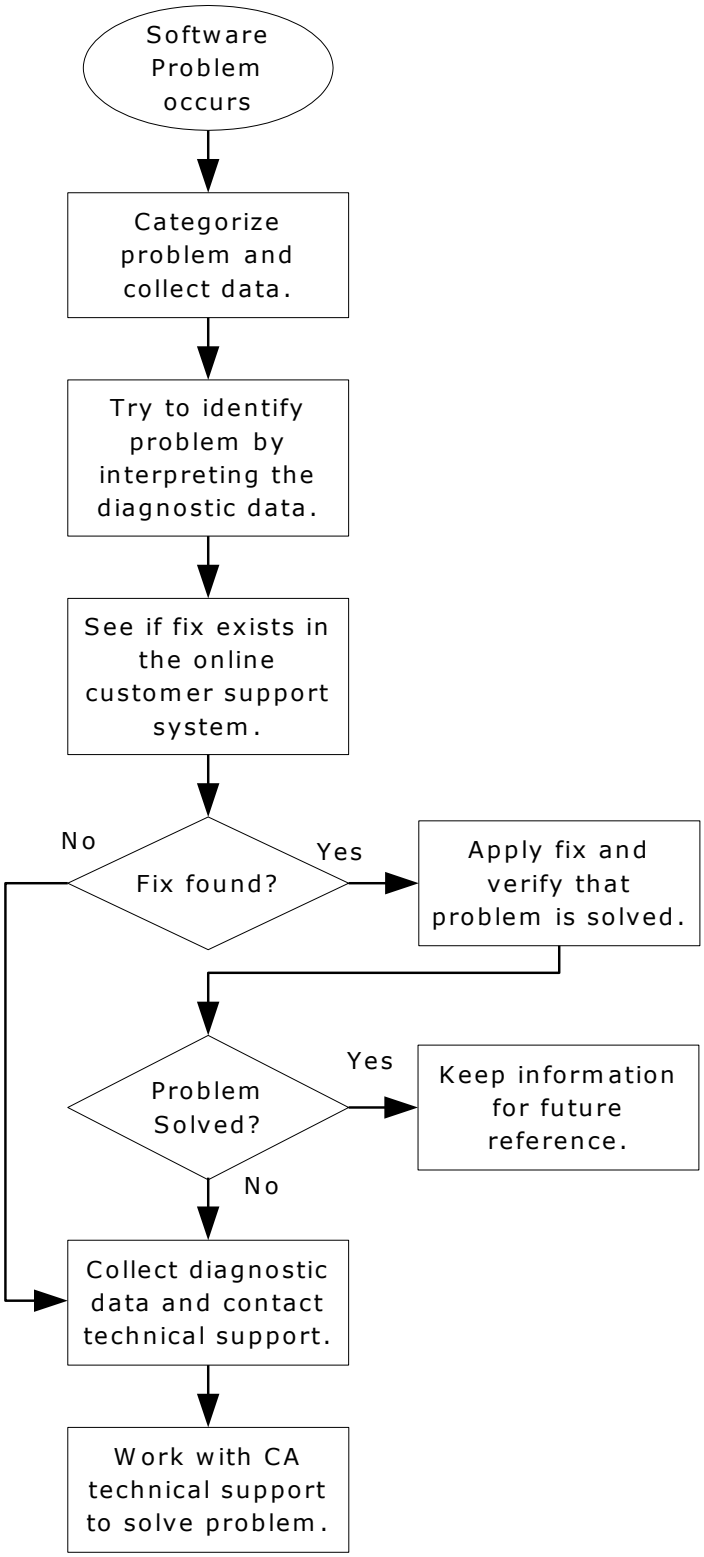
Appendix E: Contact Technical Support

This section contains information about identifying and resolving problems and contacting Technical Support.

This section contains the following topics:

Diagnostic Process

The following chart shows the process to follow when you have a problem with a CA software product. Each of these procedures is detailed on the following pages.


Software Problem occurs
Categorize problem and collect data.
Try to identify problem by interpreting the diagnostic data.
See if fix exists in the online customer support system.
Fix found?
No
Yes
Apply fix and verify that problem is solved.
Problem Solved?
Yes
Keep information for future reference.
No
Collect diagnostic data and contact technical support.
Work with CA technical support to solve problem.

Collect Diagnostic Data

The following information is helpful in diagnosing problems that might occur:

- Control statements used to activate your product
- JCL used to install or activate your product
- Attach the output for the capture command of "CAPTURE DIAG0000"to the issue. This will aid the CA Technical Support in resolving the issue.
- Relevant system log or console listings
- Relevant system dumps or product dumps
- List of other IBM or third-party products that might be involved
- Manufacturer, model number, and capacity of your hardware
- Numbers and text of IBM or CA error messages associated with the problem
- Names of panels where the problem occurs
- Listings of all fixes applied to all relevant software, including:
  - The date's fixes were applied
  - Fix numbers
  - Names of components to which fixes were applied
- Short description of the problem

Interpret Diagnostic Data

When you have collected the specified diagnostic data, write down your answers to the following questions:

1. What was the sequence of events prior to the error condition?
2. What were the circumstances when the problem occurred and what action did you take?
3. Has this situation occurred before? What was different then?
4. Did the problem occur after a particular PTF was applied or after a new release of the software was installed?

5. Have you recently installed a new release of the operating system?
6. Has the hardware configuration (tape drives, disk drives, and so forth) changed?

From your response to these questions and the diagnostic data, try to identify the cause and resolve the problem.

Access the Online Client Support System

Support Online is an online product support and service system available on the Internet from CA. Support Online contains an extensive Knowledge Base that lets you retrieve many types of product-related information with a single search. Support Online also provides full product downloads and an easy-to-use Suggestion Box for you to submit enhancement ideas for your licensed products.

The features of Support Online include:

- Solution downloads
- Technical Support issue management
- License key downloads
- Virus signature downloads
- Product-specific FAQs
- Product documentation downloads
- Newsgroup open forums
- E-News newsletters

Requirements for Using Support Online

With a current release of a browser and without logging in, you have access to a considerable amount of information about the website. This access includes access to FAQs, published solutions (for channel products only), compatibilities, virus signatures, news, CA user group information, and support services, and to perform general Knowledge Base searches.

When you log in, you have for full access to all the services related to your licensed products. These services include published solutions, license keys, newsgroups, Hyper Subscriptions, product and documentation downloads, and issue management. These areas require that you are a registered Support Online user.

If you enrolled at AccountConnect.ca.com, you can log in to Support Online using your AccountConnect digital certificate rather than entering a login name and password. However, this works only on the PC where the digital certificate resides, and only if you are using Microsoft Internet Explorer 5.5 or later. To access Support Online from another PC, or if you are using Netscape Navigator, provide your login name and password.

CA TLC: Total License Care

Many CA software solutions use license keys or authorization codes to validate your hardware configuration. If you need assistance obtaining a license key or authorization code, contact the CA TLC: Total License Care group through Support Online.

Call CA Technical Support

For further technical assistance with this product, Technical Support is available 24 hours a day, seven days a week. You can obtain a complete list of CA locations and telephone numbers by product through Support Online at the Support Services link.

If you are unable to resolve the problem, have the following information ready before contacting CA Technical Support:

- All the diagnostic information described in Collect Diagnostic Data.
- Product name, release number, operating system, and genlevel.
- Product name and release number of any other software you suspect is involved.
- Release level and PUTLEVEL of the operating system.
- Your name, telephone number and extension (if any).
- Your company name.

- Your site ID.
- A severity code. This code is a number (from 1 to 4) that you assign to the problem. Use the following to determine the severity of the problem:

  1

  a "system down" or inoperative condition

  2

  a suspected high-impact condition associated with the product

  3

  a question concerning product performance or an intermittent low-impact condition associated with the product

  4

  a question concerning general product use or implementation

Product Releases and Maintenance

Customers are requested to operate only under currently supported versions of the product.

Customers with current maintenance agreements also receive ongoing product maintenance. When a new release of the system is available, a notice is sent to all current customers.

Request Enhancements

CA welcomes your suggestions for product enhancements. All suggestions are considered and acknowledged. You can use either of two methods to request enhancements:

- Enter your request through at http://ca.com, the CA web-based, interactive support system.
- Ask your Account Manager to initiate the request for you.

Index

D

E

F

P

R

S

T

U

V

W

X

Z