CA MIM™ Resource Sharing for z/OS

Programming Guide

Release 12.0

Second Edition

CA Technologies Product References

This document references the following CA Technologies products:

- CA ASTEX™ Performance
- CA Common Services for z/OS (CCS)
- CA Easytrieve® Report Generator (CA Easytrieve)
- CA MIA Tape Sharing (CA MIA)
- CA MIC Message Sharing (CA MIC)
- CA MII Data Sharing (CA MII)
- CA MIM™ Resource Sharing (CA MIM)
- CA OPS/MVS® Event Management and Automation (CA OPS/MVS)
- CA Vantage™ Storage Resource Manager (CA Vantage SRM)

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

Note: In PDF format, page references identify the first page of the topic in which a change was made. The actual change may appear on a later page.

The following list details the changes made since the previous release of this guide:

For Release 12.0 Second Edition:

- Updated the Plan the MIMINIT Member (see page 18) section.
- Added the Hyperstar (see page 30) section.
- Updated the How You Activate New Features with ACTIVATE COMPATLEVEL (see page 116) section.
- Updated the How You Use ACTIVATE COMPATLEVEL with Checkpoint Files (see page 118) section.
- Added the How You Activate New Features with ACTIVATE FEATURE (see page 119) section.
- Added the How You Use ACTIVATE FEATURE with Checkpoint Files (see page 120) section.
- Added the How to DEACTIVATE a FEATURE (see page 120) section.

For Release 12.0:

- Added the How to Dynamically Change CA MIM Communication Methods (see page 76) section.
- Added the The CA MIM Logging Facility (see page 144) section.
- Updated the Plan the MIMINIT Member (see page 18) section.
- Updated the Communication Methods (see page 26) section.
- Updated the Physical Control File Communication Methods (see page 26) section.
- Updated the DASD Control File Communication (see page 27) section.
- Updated the Coupling Facility Structure Control Files (XES) (see page 28) section.
- Updated the Virtual Control File Communication Methods (see page 29) section.
- Updated the Initial Communication Methods Overview (see page 31) section.
- Updated the CTCDASD as the Initial Communication Method (see page 34) section.
- Updated the CTCONLY as the Initial Communications Method (see page 36) section.
- Updated the XCF as the Initial Communication Method (see page 38) section.
- Updated the NONE as the Communication Method (see page 40) section.
- Updated the ICMF Communication Method (see page 41) section.

- Updated the How You Define Your VCF Master and Error Recovery Environment (see page 41) section.
- Updated the How You Define CTC Devices for Use in VCF Environments (see page 44) section.
- Updated the How CA MIM Formats Control and Checkpoint Files (see page 63) section.
- Updated the How You Migrate to a New Control File (see page 64) section.
- Updated the Migrate to a DASD Control File (see page 65) section.
- Updated the Migrate to a Coupling Facility Structure Control File (see page 66) section.
- Updated the Migrate to a Virtual Control File (see page 67) section.
- Updated the Local Shutdown with Restart Manager (see page 108) section.
- Updated the How You Free Inactive Systems (see page 109) section.
- Updated the How You Activate Features of New Releases Dynamically section.
- Updated the Plan the MIMCMNDS Member (see page 21) to include Execution Attributes.
- Updated the How You Use the Dynamic Compatibility Feature with Checkpoint Files section.
- Updated the Performance Considerations (see page 121) section.
- Updated the How to Set a Preferred Medium for VCF Communication (see page 130) section.
- Updated the How You Alter System Definitions While CA MIM Executes (see page 150) section.
- Updated the How You Send Trace Data to an Output File (see page 165) section.
- Updated the Message Table Syntax Rules (see page 195) section.
- Updated the API Rules (see page 222) section.
- Updated the Report Generation for CA MIM (see page 198) section.

Contents

Chapter 1: Introduction

This section contains the following topics:

Intended Audience

This book is intended for system programmers and operators responsible for the installation, customization, and day-to-day operation of CA MIM.

Introduction to CA MIM

CA MIM is the industry standard for sharing DASD, tape, and console resources safely and efficiently in z/OS and z/VM multiple-image environments. The product streamlines and automates many of the procedures involved in sharing resources and enables multiple-image sites to share data center resources across as many as 32 system images.

Components and Facilities

CA MIM is comprised of three product components and a driver, which manages communications among mainframe systems.

CA MIA Tape Sharing

CA MIA enables z/OS sites, z/VM sites, and mixed z/OS and z/VM sites with CMS users and z/OS guests, to share tape devices automatically and safely. CA MIA provides integrity for data that resides on tape and eliminates the necessity for manual commands typically associated with tape device sharing.

CA MIA consists of the following facilities:

Global Tape Allocation Facility (GTAF)

Prevents jobs on different systems from simultaneously allocating the same tape, and prevents jobs from allocating devices already in use on another system.

Tape Preferencing and Control Facility (TPCF)

Lets you influence device selection during the z/OS allocation process. TPCF also responds automatically to the messages z/OS issues when a job cannot allocate a suitable online device.

CA MIC Message Sharing

CA MIC provides cross-system command routing from any z/OS or z/VM console and allows messages to be imported from external systems and routed to local consoles.

CA MIC facilitates global console management by allowing operations to control and customize the flow of console information so that systems can be monitored from a single point with conveniently accessible console output. This improves operations productivity and performance, and ensures that all systems are monitored consistently.

CA MIC consists of the following facilities:

Global Command and Message Facility (GCMF)

Allows you to route messages and commands to any or all systems in a complex.

Intersystem Communication Facility (ICMF)

Allows you to route cross-system commands and messages using an interface with the CA-L-Serv product.

CA MII Data Sharing

CA MII Data Sharing (CA MII) protects z/OS data integrity automatically, speeds resolution of resource conflicts in shared DASD environments, and adds additional integrity at a local system level.

CA MII consists of the following facilities:

Global Data Integrity Facility (GDIF)

Prevents simultaneous updates that occur when requests for resources are not communicated to all systems.

Enqueue Conflict Management Facility (ECMF)

Helps Time-Sharing Option (TSO) users and system operators identify and resolve conflicting requests for resources

Enhanced Data Set Integrity Facility (EDIF)

Prevents the most common sources of data set damage, such as damage to attributes or due to DISP=SHR updates

CA MIM Driver

The CA MIM Driver manages global activity of the product components by routing transactions across mainframe images through a common control file.

Global activity is managed by the routing of transactions across system images through either a small control file residing on a shared DASD volume or a virtual control file residing in private storage in the address space of a product on a selected master system.

The virtual control file architecture uses CTC devices to pass transaction data between z/OS and z/VM systems.

In a parallel sysplex environment with a CA MIM complex that is equal to or is a subset of the parallel sysplex complex, the control file can be placed in the coupling facility. This provides a significant performance enhancement through the reduction of I/O transfer times as compared to CTC, cached DASD, and non-cached DASD I/O operations.

CA MIM lets you define backup communication methods. This capability furnishes data centers with the redundancy needed to guarantee uninterrupted resource integrity as the operating environment changes or during hardware outages. While the product is running, migrations can be initiated between DASD control files, between virtual control files, or between DASD and virtual control files.

The CA MIM transaction processing architecture is based on a star configuration. With this architecture, every image needs only a single access to the control file to determine the global status of all managed resources. Frequency of access to the control file is based on the amount of resource activity on a particular image.

The CA MIM Driver can be defined as CA MIM address space control code, which supervises the activities of the CA MIM address space, regardless of which CA MIM facilities are activated.

Chapter 2: Planning Initial Settings

This section contains the following topics:

Overview

The MIMPARMS data set contains parameter values that CA MIM uses as input during product initialization to define the characteristics of the MIMplex. You specify the MIMPARMS data set on the //MIMPARMS DD statement in the JCL procedure used to start CA MIM. A sample MIMPARMS data set is installed with the CA MIM product into data set CAI.CBTDPARM.

You need to set initial values for statements and commands in the following CA MIM MIMPARMS parmlib members:

- MIMINIT
- MIMCMNDS
- MIMSYNCH

Note: For detailed information on each statement and command, see the *Statement and Command Reference Guide*.

General Statements

The following general statements can be used in any of the CA MIM parmlib members:

- IFSYS
- ENDIF
- INCLUDE
- LOG
- NOLOG

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

Plan the MIMINIT Member

The MIMINIT member for the parameter data set contains statements that specify initialization values for CA MIM. This member is pointed to in the PROCMIM member on the PROC MEMBER=MIMINIT statement. The initialization members specific to CA MIM do the following:

- Define control and checkpoint files data sets. You can direct CA MIM to allocate dynamically these types of data sets. Use the following parameters:
  - ALLOCATE DDNAME
  - ALLOCATE DSNAME
  - ALLOCATE XESFILEID
  - ALLOCATE STRNAME
- Define the CTC device addresses connecting systems. Use the following parameters:
  - CTCPATH ADDRESS
  - CTCPATH FROMSYSTEM
  - CTCPATH TOSYSTEM
- Identify the systems participating in the MIMplex. Use the following statement:
  - DEFSYS
- Define Virtual Control File (VCF) master system and recovery options. Use the following parameters:
  - GLOBALVALUE ANYELIGIBLE
  - GLOBALVALUE MOSTPREFERRED
  - GLOBALVALUE NOMASTER
  - GLOBALVALUE VCFMASTER

- Define general initialization values that affect the entire address space. The MIMINIT statement is used to define the following:
  - Communication method. Use the following parameter:
    - MIMINIT COMMUNICATION
  - Facilities to activate. Use the following parameters:
    - MIMINIT ECMF
    - MIMINIT EDIF
    - MIMINIT GCMF
    - MIMINIT GDIF
    - MIMINIT GTAF
    - MIMINIT ICMF
    - MIMINIT TPCF

    Facilities are activated on the MIMINIT statement by specifying ON or OFF. For example,

    ```
    MIMINIT ECMF=ON, GDIF=ON, EDIF=ON,   /* ALL MII FACILITIES
                     GTAF=ON, TPCF=ON,   /* ALL MIA FACILITIES
                     GCMF=ON, ICMF=ON    /* ALL MIC FACILITIES
    ```

    Once a facility has been activated on the MIMINIT statement, facility-specific initialization statements are eligible for processing. Use the following parameters:
    - GDIINIT
    - EDIINIT
    - GTAINIT
    - GCMINIT
    - ICMINIT
    - TPCINIT
  - Selectable features to activate. Use the following parameter:
    - MIMINIT FEATURE=name

      Multiple features can be specified on a single MIMINIT statement, or a separate MIMINIT statement can be used for each feature. For example:

      MIMINIT FEATURE=Hyperstar
  - I/O buffer sizes. Use the following parameters:
    - MIMINIT BLKSIZE
    - MIMINIT VCFBUFFERSIZE
    - MIMINIT VCFMAXBLOCKS

- SAF command security options. Use the following parameters:
  - MIMINIT SAFCMDAUTH
  - MIMINIT SAFPREFIX
- Component-specific PDS members to use. Use the following parameter:
  - MIMINIT MEMBER
- Names of systems and subsystems. Use the following parameters:
  - MIMINIT SUBNAME
  - MIMINIT MIMPLEX
  - MIMINIT SYSID
- Message handling. Use the following parameters:
  - MIMINIT LOGPARAMETERS
  - MIMINIT MSGPREFIX
  - MIMINIT MSGTEXT
  - MIMINIT SIGNON
  - MIMINIT SUPPRESSRESP
- Execution attributes. Use the following parameters:
  - MIMINIT BATCHJOB
  - MIMINIT CANCEL
  - MIMINIT PAGEFIX
- Control files settings. Use the following parameters:
  - MIMINIT COMPATLEVEL
  - MIMINIT FORMAT
  - MIMINIT MODE
- Command settings. Use the following parameters:
  - MIMINIT CMDPREFIX
  - MIMINIT COMMANDS

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

Plan the MIMCMNDS Member

You use the commands found in the MIMCMNDS member to define facility-specific variables unique to the processing requirements at your site. This member name is pointed to in the PROCMIM member by the PROC CMNDS=MIMCMNDS statement.

See the sample MIMCMNDS member in the CAI.CBTDPARM data set. The commands found in this member are executed after the MIMINIT member statements and before synchronization completes. In general, this member contains SETOPTION commands for each facility being activated.

You can define the following settings in the MIMCMNDS parmlib member:

- Define command aliases for CA MIM commands using the DEFALIAS command. You can substitute a command alias for the full command specification (including operands) when you want to issue that command.

  You can use command aliases for these purposes:

  - To issue a shorter version of a command (for example, in place of a display command that has several parameters)
  - To issue a command in a way that is easier to remember or more consistent with commands that you already know

  You can append additional parameters to a command alias. For example, if CONF was defined as DISPLAY CONFLICTS, you can append the QNAMES parameter to the CONF command alias like this:

  ```
  CONF QNAMES
  ```

  This tells CA MIM to execute the DISPLAY CONFLICTS QNAMES command.

  You cannot specify the MCS L parameter in this list of parameters if you are issuing the DEFALIAS command from a console or from a TSO session.

  Use the command name, rather than the alias, when looking up information about the command or when contacting CA Technical Support. Also, because of changes in command format or function, you may need to redefine existing aliases when you upgrade to a new release of CA MIM.

- Disable a CA MIM command. Any command can be disabled, and individual operands can be disabled on the DISPLAY, DUMP, GLOBALVALUE, SETOPTION, and SETTRACE commands. Disabled commands and operands cannot be restored by any command, but they become available again once CA MIM is restarted. CA MIM parameters are command and command [facility] operands.

  The DISABLE command is typically used to prevent system operators or TSO users from issuing specific CA MIM commands. Use the DISABLE Command.

- Define operating values for the following general-purpose CA MIM functions:

  - Control file. Use the following parameters:

    - SETOPTION MIM CFSIZEWARN

    - SETOPTION MIM CYCLES
    - SETOPTION MIM HIBERNATE
    - SETOPTION MIM INTERVAL
    - SETOPTION MIM LOCKOUT
    - SETOPTION MIM MARGIN
    - SETOPTION MIM MODE
- Commands. Use the following parameters:
    - SETOPTION MIM CMDPREFIX
    - SETOPTION MIM CMDRESPMAX
    - SETOPTION MIM CMDTIMEOUT
- Execution Attributes. Use the following parameter:
    - SETOPTION MIM ZIIP
- Tracing. Use the following parameters:
    - SETOPTION MIM CELLTRACE
    - SETOPTION MIM RESETPRINT
    - SETOPTION MIM RESETTRACE
    - SETOPTION MIM SETPRINT
    - SETOPTION MIM SETTRACE
    - SETOPTION MIM TRACE
- Automated Free. Use the following parameter:
    - SETOPTION MIM DOWNSYS
- SMF. Use the following parameters:
    - SETOPTION MIM STATCOLLECT
    - SETOPTION MIM STATCYCLE
    - SETOPTION MIM STATINTERVAL
- Termination. Use the following parameters:
    - SETOPTION MIM LOCALSTOP
    - SETOPTION MIM SHUTDOWN
- Virtual control file. Use the following parameters:
    - SETOPTION MIM VCFFORCE
    - SETOPTION MIM VCFMAXDELAY
    - SETOPTION MIM VCFMINDORM
    - SETOPTION MIM VCFRECOVERY

Plan the MIMSYNCH Member

The MIMSYNCH member of the CA MIM parameter data set contains CA MIM, z/OS, and z/OS subsystem commands you want to execute when CA MIM synchronizes with other systems in the MIMplex.

The MIMSYNCH member is specified on the SYNCH= statement in the PROCMIM startup procedure. Commands found in this member execute after MIMINIT statements and MIMCMNDS commands have executed, and when all CA MIM address spaces in the complex have synchronized.

See the sample MIMSYNCH and PROCMIM members in the CAI.CBTDPARM data set. The commands found in this member can be CA MIM, z/OS, or any z/OS subsystem commands you want to execute, once synchronization completes. We recommend that VARY commands be placed in this member when you are running CA MIA.

Plan the Message Facility Members

The CA MIM message facility includes the following members:

- EDIMSGS
- ICMMSGS
- MIAMSGS
- MICMSGS
- MIIMSGS
- MIMMSGS
- MIMMSGX

The message facility provides the ability to customize messages issued by CA MIM. You can use the message facility to customize message text and routing attributes such as routing and descriptor codes. The default message tables are located in the CAI.CBTDMSEN data set.

This facility can be used only to customize CA MIM messages. Only those messages found in the default message tables can be customized.

The message facility also has the ability to provide non-English releases of CA MIM for messages. While this release of CA MIM does not provide default non-English message tables, you can create customized non-English message tables. The message facility will be enhanced in future releases of the CA MIM product.

Chapter 3: Advanced Topics

This section contains the following topics:

Communication Methods

Your managed resources are represented in transactions routed to all systems in the MIMplex. CA MIM has many transaction types including:

- Device allocations
- Resource access
- Cross-system messages and commands
- Systems status changes.

Transactions are transported using a common control file. CA MIM lets you pick from a number of control file architectures and storage mediums, and dynamically change communications methods during the execution.

The type of CA MIM cross-system communication method that is best suited for your environment depends on several factors. Before you proceed, we recommend that you read Performance Considerations (see page 121) in this chapter for a better understanding of CA MIM control file internals and externals.

The following sections provide an overview of the two primary communication methods available to CA MIM: physical and virtual.

Physical Control File Communication Methods

CA MIM supports DASD files and coupling facility list structures for the communication method. The CA MIM control file is kept on the shared physical file. Individual CA MIM systems access the control file when it required global communication.

You can, at any given time, have 100 control files that are allocated for use by CA MIM. However, only the current control file is used for global communications. During a hardware or software failure, CA MIM automatically migrates to an alternate physical control file.

DASD Control File Communication

A DASD control file is a data set that resides on a shared DASD volume. The control file is used to communicate CA MIM transactions to all systems within the MIMplex. To use a DASD file for communication, the file must reside on a volume accessible by all systems participating in the MIMplex.

DASD control files must be predefined. A sample control file allocation job is provided with the CA MIM installation. The size of the DASD control file is specified in the JCL and the DASD volume that is to contain the data set.

The CA MIM address space on each system contends for access to the DASD control file by issuing hardware reserve requests. Access to the control file is serialized, because only one system can reserve the control file at any given moment.

Use the MIMINIT ALLOCATE statement to allocate a DASD control file to CA MIM during the initialization. In addition, you can issue the ALLOCATE command dynamically while CA MIM is executing. Once CA MIM allocates a DASD control file , you can migrate to it to begin using DASD control file communications.

Coupling Facility Structure Control Files (XES)

A coupling facility list structure control file is a list type of structure that resides in a coupling facility. A coupling facility can be a stand-alone device, for example, a 9674, or it can be a PR/SM LPAR running the coupling facility control code. To use a coupling facility list structure for communication, all CA MIM systems must reside in the same sysplex.

Coupling facility list structures must be predefined. Define coupling facility structure control files in the Coupling Facility Resource Management (CFRM) policy couple data set:

- Allocate and format the CFRM couple data set using the IBM utility IXCL1DSU, if one is not already in use in the parallel sysplex.
- Update the policy information in the CFRM couple data set using the IBM utility IXCMIAPU.
- Activate the coupling facility policies using the following z/OS operator command:

  ```
  SETXCF START,POLICY
  ```

Note: For detailed information, see the IBM publication, *Setting up a Sysplex*.

You can identify coupling facility structure control files using the following methods:

- Using the ALLOCATE statements in the MIMINIT member of the CAI.CBTDPARM data set
- Dynamically issuing the ALLOCATE command

The XESFILEID keyword and the STRNAME keyword of the ALLOCATE statement uniquely specify a control file identifier and a structure name for the control file.

From an operational standpoint, a coupling facility structure control file functions similarly to a DASD control file. CA MIM address spaces on each system of the MIMplex contend for access to the list structure. However, instead of a physical hardware reserve, the list header lock is used to serialize cross-system access to the list structure.

Virtual Control File Communication Methods

A virtual control file (VCF) is an area of virtual storage that is located in a pre-selected CA MIM address space. The CA MIM address space that is selected to manage the VCF is known as the master system. A system is master eligible if it has connectivity to all systems participating in the MIMplex.

When client systems must propagate global information, they must request the VCF through one of these methods:

- The CTC devices
- The z/OS Cross System Coupling Facility (XCF)

Once the master system receives the request, it transports the VCF back to the requesting client system. Only the master system knows which system has the VCF at any given moment. Once the client system receives the VCF, it is updated and transported back to the master. The master system then sends the VCF to the next waiting requestor (client system). The master system services VCF access requests on a first in, first out (FIFO) basis. This philosophy ensures that the systems with the most global activity receive the best possible service.

In a VCF environment, CA MIM supports a simultaneous mix of CTC devices and XCF for MIMplex communications. CA MIM systems within a sysplex can use XCF communication, while systems outside the sysplex must use CTC devices.

Use the VCFPREFERENCE command to dictate what type of communication path a client system should use when requesting the control file from the master system. If only one type of path is available (XCF or CTC device), then it is used regardless of your preference setting. In addition, the ability to use both XCF and CTC devices can be useful in recovery situations. This ability lets CA MIM fall back to any communication path that is available, if the current path is in error. Define the CTC devices to CA MIM using either of the following methods:

- The CTCPATH statement in the CA MIM initialization parameter member
- By dynamically issuing the CTCPATH command while CA MIM is executing

CA MIM automatically discovers the XCF paths during initialization and as systems join the CA MIM XCF group.

Hyperstar

For a MIMplex that has 3 or more active systems, performance of Virtual Control Files may be improved by activating the Hyperstar feature. When this feature is active, a client system does not always transfer the VCF back to the master system after completing its processing. Instead, CA MIM examines a ‘look ahead list’ of systems waiting for the VCF. If the list is not empty, and the client has a CTC or XCF path to the next system in the list, the client transfers the VCF directly to that system instead of the master. When the ‘look ahead list’ becomes empty, or when there is no path to the next system in the list, the client transfers the VCF back to the master.

The primary effect of Hyperstar is to reduce the amount of VCF I/O performed by the VCF master system, in many cases by more than 50 percent; this in turn reduces the amount of CPU time that is used on the master system, and can potentially reduce the control file access times for all systems in MIMplex.

Initial Communication Methods Overview

The MIMINIT COMMUNICATION parameter allows you to specify which communications method you want CA MIM to use upon startup.

The options include:

1. DASDONLY (Physical) – Instructs CA MIM to use a physical control file (DASD or coupling facility list structure) upon startup. The DASDONLY communication provides the control file on a shared DASD volume or a coupling facility list structure.
   - Benefits
     - Allows the MIMplex systems to be any combination of sysplexed and non-sysplexed images.
     - Solid-State DASD provides a high performance equivalent to CTC communication options.
     - Coupling facility list structures provide high performance.
   - Requirements
     - A 3380/3390-type DASD volume or a coupling facility list structure accessible by all systems participating in the MIMplex.
     - If using coupling facility structure control files are used, all MIMplex systems must be in the same parallel sysplex.
   - Recommendations
     - At least ten cylinders of DASD must be available for a DASD CA MIM control file.
2. CTCDASD (VCF/Physical) – Instructs CA MIM to use the communications method specified by the INITIAL sub-keyword; either CTC or DASD. However, by specifying CTCDASD as the initial communications method, the DASD file is always treated as a backup and recovery will result in a migration to the DASD file.
   - Benefits
     - Allows the MIMplex systems to be any combination of sysplexed and non-sysplexed images.
     - Provides high performance, especially for CA MII sites.
     - Provides backup master system capability in a master system failure.
     - Defines backup DASD or coupling facility structure control file in a CTC failure or if there are no available backup masters.
     - Is not subject to contention delays associated with DASD I/O when VCF is in use.
   - Requirements

- A 3380/3390-type DASD volume or coupling facility list structure accessible by all systems participating in the MIMplex.
- 3088 or extended mode ESCON CTC devices connecting at least one system to all others in the MIMplex.
- At least one CA MIM address space defined to manage the virtual control file (VCF).

3. CTCONLY (VCF) – Instructs CA MIM to use CTC devices for communications upon startup. The CTCONLY communication method has the control file located in virtual storage in a master CA MIM address space and passed from the master to client systems through CTC devices.
   - Benefits
     - Allows the MIMplex systems to be any combination of sysplexed and non-sysplexed images.
     - High performance, especially for CA MII sites.
     - Requires no shared DASD control file.
     - Requires no coupling facility structure control file.
     - Provides backup master system capability in case of master system failure.
     - Is not subject to contention delays associated with DASD I/O.
   - Requirements
     - A 3088 or extended mode ESCON CTC devices connecting at least one system to all others in the MIMplex.
     - At least one CA MIM address space that is defined to manage the virtual control file (VCF).
     - At least one checkpoint file defined on each system in the MIMplex.
4. XCF (VCF) – Instructs CA MIM to use XCF communication upon startup. The z/OS Cross System Coupling Facility (XCF) communication method is similar to CTCONLY; however, the virtual control file (VCF) is passed from the master system to client systems through the XCF.
   - Benefits
     - Provides high performance, especially for CA MII sites.
     - Provides RMF reporting for optimizing XCF signaling and messaging
     - Allows for a backup master system in case of a master system failure.
     - Is not subject to contention delays associated with DASD I/O.
     - Requires no coupling facility structure control file.
     - Requires no shared DASD control file.
     - Automatically discovers connections between CA MIM systems, requiring no additional configuration.

- Requirements
  - All systems in the MIMplex must be in a single sysplex.
  - At least one CA MIM address space defined to manage the virtual control file (VCF).
  - At least one checkpoint file must be defined on each system in the MIMplex.
  - CA Common Services for z/OS.

5. ICMF - This is a special communication method that can be used to transport transactions to part of the MIMplex. ICMF is used only with the CA MIC component. This method can be used to transmit console messages between z/OS systems that do not communicate with each other through the control file.
6. NONE – This option allows you to start a CA MIM task on a single system without involving any cross-system communication media. This option is used primarily in new installations for pre-production testing.

Although you must specify an initial communications method on the MIMINIT COMMUNICATION parameter, you are not constricted to using that communication method during execution (except for COMMUNICATION=NONE or COMMUNICATION=ICMF). At any time, you can dynamically change communication methods, after correctly configuring an alternate method. For more information, refer to the following scenario documentation: *How to Dynamically Change CA MIM Communication Methods*.

CTCDASD as the Initial Communication Method

The CTCDASD initial communication method is a combination of physical control files and VCF methods using the DASD file or a coupling facility structure as a backup.

The MIMplex can consist of sysplexed and non-sysplexed images. This method is actually a combination of the CTCONLY and DASDONLY communication methods.

A DASD control file or a coupling facility structure control file is used as the base communication method during initialization and recovery situations. During initialization, all systems synchronize on the DASD or coupling facility structure control file. If all VCF communication requirements are met, CA MIM then automatically migrates from the DASD or coupling facility structure control file to the virtual control file (VCF) on the selected master system.

When VCF synchronization has completed, the VCF is passed on demand from the master system to client systems. The VCF is passed between systems using 3088 or ESCON CTC devices allocated to the CA MIM address space on each system. The CTC devices to be used by CA MIM are identified on CTCPATH statements.

Conceptually, a CTC device connects an I/O address on one processor to an I/O address on another processor. VCF data that is sent from one side is received on the other side, so every transmission consists of two operations:

- An outbound write operation from one side
- An inbound read operation on the other side

Data can be transmitted in either direction over a CTC path, but it travels only in one direction at any one moment.

When using the CTCDASD communication method, it is important to remember that the DASD or coupling facility structure control file is the base communication medium. Migration to the DASD or coupling facility structure control file occurs any time a resynchronization of the MIMplex is required. If possible, CA MIM automatically migrates back to using the VCF when resynchronization completes.

Note: Although you may have specified CTCDASD as your initial communication method, CA MIM will automatically join an XCF group. CA MIM will build XCF communication paths for all other active members of the CA MIM XCF group. You can instruct each client system to use an XCF path to the master system (if available) by setting your VCFPREFERENCE.

Implement CTCDASD Communication

To implement the CTCDASD communication method, you customize certain parameters in the MIMINIT member.

Follow these steps:

1. Define the systems participating in the MIMplex using the following DEFSYS statements:

   ```
   DEFSYS(sysa,aa,sysa)
   DEFSYS(sysb,bb,sysb)
   ```

2. To specify preferred master systems and recovery options use the GLOBALVALUE statement.

3. Define the CTC devices for CA MIM communication using the following CTCPATH statements:

   ```
   CTCPATH FROMSYSTEM=sysa ADDRESS=deviceaddress TOSYSTEM=sysb
   CTCPATH FROMSYSTEM=sysb ADDRESS=deviceaddress TOSYSTEM=sysa
   ```

4. Specify CTCDASD as the initial communications method using the MIMINIT COMMUNICATION statement:

   ```
   MIMINIT COMMUNICATION=(CTCDASD,INITIAL=[CTC|DASD])
   ```

   Note: The INITIAL parameter is unique to the CTCDASD communication method. It indicates your preferred communication method. After the synchronization has completed on the DASD or the coupling facility structure control file, this value determines which communication method to employ:

   - INITIAL=CTC indicates CA MIM is to automatically migrate to a selected VCF master.
   - INITIAL=DASD indicates CA MIM is to continue using the DASD or coupling facility structure control file. If desired, the MIGRATE command can be issued later to migrate to a master VCF system.

   This value is checked after the initial DASD or coupling facility structure control file synchronization, and after any event causing a DASD or coupling facility structure control file resynchronization.

5. Define the control files CA MIM will use using the ALLOCATE statement.

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

CTCONLY as the Initial Communications Method

The CTCONLY initial communication method uses a control file located in virtual storage (VCF) of the master CA MIM address space. The control file is passed from master to client using CTC devices.

When using CTC devices to communicate, the MIMplex can consist of sysplexed and non-sysplexed images. No shared DASD or coupling facility structure control file is required.

The CTCONLY communication method passes a virtual control file (VCF) on demand from a selected master system to client systems. The VCF is passed between systems using 3088 or ESCON CTC devices that are allocated to the CA MIM address space on each system. You can define the CTC devices using either of the following methods:

- Place CTCPATH statements in your MIMINIT member
- Issue the CTCPATH command dynamically while CA MIM is executing

Conceptually, a CTC device connects an I/O address on one processor to an I/O address on another processor. VCF data that is sent from one side is received on the other side, so every transmission consists of two operations:

- An outbound write operation from one side
- An inbound read operation on the other side

Data can be transmitted in either direction over a CTC path, but it travels only in one direction at any one moment.

We recommend that sites using the CTCONLY communication method are configured symmetrically. For example, every system in the MIMplex should have a CTC path available to every other system, so that multiple systems are eligible as master systems. This redundancy provides the best recovery options.

Note: Although you may have specified CTCONLY as your initial communication method, CA MIM will automatically join an XCF group. CA MIM will build XCF communication paths for all other active members of the CA MIM XCF group. You can instruct each client system to use an XCF path to the master system (if available) by setting your VCFPREFERENCE.

Implement CTCONLY Communication

To implement CTCONLY as the initial communication method, you customize certain parameters in the MIMINIT member.

Follow these steps:

1. Define the systems participating in the MIMplex using DEFSYS statements:

   ```
   DEFSYS(sysa,aa,sysa)
   DEFSYS(sysb,bb,sysb)
   ```

2. To specify preferred master systems and recovery options use the GLOBALVALUE statement.

3. Define CTC devices for CA MIM communication using CTCPATH statements:

   ```
   CTCPATH FROMSYSTEM=sysa ADDRESS=deviceaddress TOSYSTEM=sysb
   CTCPATH FROMSYSTEM=sysb ADDRESS=deviceaddress TOSYSTEM=sysa
   ```

4. Specify CTCONLY as the initial communications method using the MIMINIT COMMUNICATION statement.

5. Define checkpoint files for each system using the ALLOCATE statement. In your shared CA MIM initialization member use the IFSYS statement to direct commands to specific systems:

   ```
   IFSYS sysa
     ALLOCATE DDNAME=MIMCKP00,DSNAME=sysa.primckpt
     ALLOCATE DDNAME=MIMCKP01,DSNAME=sysa.altckpt
   ENDIF
   IFSYS sysb
     ALLOCATE DDNAME=MIMCKP00,DSNAME=sysb.primckpt
     ALLOCATE DDNAME=MIMCKP01,DSNAME=sysb.altckpt
   ENDIF
   ```

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

XCF as the Initial Communication Method

The XCF initial communications method is similar to CTCONLY; however, the VCF is passed from the master to the client systems through the z/OS Cross System Coupling Facility (XCF).

The XCF communication method is available in sites that are configured in a sysplex. XCF is not available on z/VM systems.

The XCF component of z/OS provides multisystem management services in a sysplex. CA MIM uses the XCF component of z/OS to transport the virtual control file from the master system to the client systems.

CA MIM uses the CA XCF Standard Component (LXCF) module LXCFMAIN to interface with XCF on its behalf. LXCF uses standard IBM XCF macros to exploit XCF services. The LXCFMAIN module is distributed with CA Common Services for z/OS.

XCF is a sysplex-only transport service. Therefore, the CA MIM XCF communication option is available only to those sites that have MIMplex systems in the defined sysplex. If the MIMplex includes systems outside of the sysplex, you can add the CTCPATH statements for those systems. CA MIM supports a simultaneous mixture of XCF and CTC devices for the VCF communication.

Signaling describes the mechanism through which the XCF component of z/OS transmits application data across systems in the sysplex. XCF component signaling is accomplished using 3088 or ESCON CTC devices or coupling facility list structures. The signaling methods used by the XCF component are transparent to the application exploiting XCF services. While the method of signaling used by the XCF component is transparent to CA MIM, the performance of the signaling service has a direct impact on the performance of CA MIM.

Note: Although you may have specified XCF as your initial communication method, CA MIM allows the simultaneous use of CTC and XCF communication paths. You can place CTCPATH definitions in your MIMINIT member or issue the CTCPATH command dynamically to define CTC devices for use by CA MIM.

Implement XCF Communication

To implement XCF as the initial communication method, you customize certain parameters in the MIMINIT member.

Follow these steps:

1. Define the systems participating in the MIMplex using the following DEFSYS statements:

   ```
   DEFSYS(sysa,aa,sysa)
   DEFSYS(sysb,bb,sysb)
   ```

2. To specify preferred master systems and recovery options use the GLOBALVALUE statement.

3. Specify XCF as the initial communications method using the MIMINIT COMMUNICATION statement.

4. Define checkpoint files for each system using the ALLOCATE statement. In your shared CA MIM initialization member use the IFSYS statement to direct commands to specific systems:

   ```
   IFSYS sysa
     ALLOCATE DDNAME=MIMCKP00,DSNAME=sysa.primckpt
     ALLOCATE DDNAME=MIMCKP01,DSNAME=sysa.altckpt
   ENDIF
   IFSYS sysb
     ALLOCATE DDNAME=MIMCKP00,DSNAME=sysb.primckpt
     ALLOCATE DDNAME=MIMCKP01,DSNAME=sysb.altckpt
   ENDIF
   ```

5. To provide addition redundancy, you can also define CTC devices for CA MIM communication using CTCPATH statements:

   ```
   CTCPATH FROMSYSTEM=sysa ADDRESS=deviceaddress TOSYSTEM=sysb
   CTCPATH FROMSYSTEM=sysb ADDRESS=deviceaddress TOSYSTEM=sysa
   ```

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

NONE as the Communication Method

Setting COMMUINICATION=NONE option lets you test start CA MIM without any cross-system communication capabilities. This method is used for pre-production testing and you cannot dynamically switch to alternative communication methods.

When you specify MIMINIT COMMUNICATION=NONE, no I/O operations are performed to DASD or coupling facility structure or CTC devices. Therefore, neither the Global Data Set Integrity Facility (GDIF) nor the Global Tape Allocation Facility (GTAF) can be activated. These are global facilities that require a cross-system communication method.

MIMINIT COMMUNCATION=NONE is generally used with the ICMF communication method between two geographically remote systems. Because DASD or coupling facility structures and CTC devices cannot usually be connected between geographically remote systems, no data can be transported using these traditional methods. Use MIMINIT COMMUNICATION=NONE when ICMF is activated to connect a geographically remote system to the MIMplex through CA L Serv.

MIMINIT COMMUNICATION=NONE can also be used to test certain CA MIM facilities on one system. While CA MIM is primarily used to manage global resources, several of its facilities have single-system features and responsibilities. With COMMUNICATION=NONE, you can test the following CA MIM facilities:

- EDIF
- ECMF
- TPCF

Implement COMMUNICATION=NONE

To implement COMMUNICATION=NONE, you customize certain parameters in the MIMINIT member.

Follow these steps:

1. Define the systems participating in the MIMplex using the following DEFSYS statements:

   ```
   DEFSYS(SYSA,01,SYSA)
   ```

2. Specify NONE as the communications method using the MIMINIT COMMUNICATION parameter.
3. If you are planning to activate ECMF, you must also include the ALLOCATE statements for checkpoint files.

ICMF Communication Method

The ICMF method is a special communication method that lets CA MIC transport messages and commands through the CA L Serv product. This communication method is available for use with CA MIC only, and must be used with one of the other communication methods.

You can also use this communication method between systems that cannot use shared DASD, coupling facility structure, XCF, or CTC methods. Using an interface to the CA L Serv product, ICMF can work between geographically remote systems. CA L Serv uses VTAM cross-domain communications to transmit data between registered applications such as CA MIC. The ICMF communication method can be used concurrently with the other CA MIM communication methods.

How You Define Your VCF Master and Error Recovery Environment

You can use the GLOBALVALUE statement to define VCF recovery and migration options. The following information describes how the GLOBALVALUE statement works:

- The GLOBALVALUE statement is placed in the MIMINIT member.
- The GLOBALVALUE statement in the MIMINIT member must be the same on all systems in the MIMplex.
- GLOBALVALUE statement options can be changed any time after CA MIM synchronization completes by issuing the GLOBALVALUE command. The changes are communicated across the complex.

How You Select Master Systems

The master CA MIM address space manages the VCF. Therefore, the master system uses more storage and performs more I/O operations than client CA MIM address spaces.

To define the master candidate list, issue the following statement:

```
GLOBALVALUE VCFMASTER=(sysa,sysb,sysc)
```

The master candidate list is examined from left to right during initialization to determine which CA MIM address space is to be the master system. The master candidate list is also examined in recovery situations when the current master becomes unavailable due to hardware or system-related errors. If the current master is unable to continue managing the VCF, management responsibility is passed to the next candidate in the VCFMASTER list.

You can change the master candidate list after synchronization by issuing the GLOBALVALUE command. You can issue the command from any active CA MIM system. The GLOBALVALUE parameter changes are routed to each system in the MIMplex.

The GLOBALVALUE statement has the following parameters:

- GLOBALVALUE VCFMASTER – The VCFMASTER parameter defines your preferred master candidate list. However, internally, CA MIM builds an eligible master list. The master candidate list and the eligible master list may or may not include the same systems. For example, in a four-system MIMplex (SYSA, SYSB, SYSC, and SYSD) where the CA MIM address space on each system has CTC or XCF connectivity to each of the other systems, all systems are eligible to become master. Therefore, all systems are included in the internally built eligible master list. However, you may have defined GLOBALVALUE VCFMASTER=(SYSA,SYSB). In this case, the eligible master list consists of SYSA, SYSB, SYSC, and SYSD, while the defined master candidate list consists of only SYSA and SYSB.
- GLOBALVALUE ANYELIGIBLE=[YES|NO] – In recovery situations, the ANYELIGIBLE parameter dictates whether CA MIM selects any master eligible system or only the specified systems in the VCFMASTER candidate list to begin managing the VCF.
- GLOBALVALUE MOSTPREFERRED=[YES|NO] - The most preferred master is always the leftmost system defined in the VCFMASTER candidate list. The most preferred master may or may not be the currently active master. If a more preferred system joins a currently executing MIMplex and MOSTPREFERRED=YES, CA MIM will automatically migrate master systems to the more preferred system.

  Consider the following example where a more preferred system joins a currently executing MIMplex:

  - A four-system MIMplex is defined including systems SYSA, SYSB, SYSC, and SYSD
  - GLOBALVALUE VCFMASTER=(SYSA,SYSB)
  - GLOBALVALUE MOSTPREFERRED=YES

- SYSA is down and FREED
- The currently active master system is SYSB
- Systems SYSB, SYSC, and SYSD are synchronized

When SYSA joins the MIMplex, CA MIM evaluates the MOSTPREFERRED parameter. In this scenario, MOSTPREFERRED is YES; therefore, CA MIM automatically transfers master responsibility to SYSA. If MOSTPREFFERED were NO, SYSB would have retained its master status.

- GLOBALVALUE NOMASTER=[WAIT|TERMINATE] - The NOMASTER parameter defines how client systems react when the currently active master CA MIM address space terminates or becomes unresponsive and there are no other eligible masters available in the MIMplex. This parameter determines whether the client CA MIM address spaces should terminate or wait for an eligible master to rejoin the MIMplex. However, if your initial communications method was CTCDASD, an automatic migration to your physical control file occurs when the master system becomes unavailable.
- GLOBALVALUE VCFMASTER = (sysa, sysb, ...sysn) - The VCFMASTER parameter defines your master candidate list. The leftmost system in the list is your most preferred master. An eligible master has connectivity to each system in the MIMplex. If a system has connectivity to each system in the MIMplex, then that system is added to the eligible master list. If a VCFMASTER parameter is defined having systems that are not eligible to become master systems, CA MIM attempts to select a master based on the eligible master list built internally.

As mentioned previously, master eligibility is determined internally. CA MIM systems evaluate their master eligibility dynamically during initialization, as VCF communication paths become available, and as dynamically added systems join the executing MIMplex. For a system to be master eligible, it must meet at least one the following criteria:

- Fully defined and connected CTCPATH statements in MIMINIT. In other words, a system must have a CTCPATH statement to every defined CA MIM system and every CA MIM system must have a CTCPATH statement defined to that system. However, if communication paths are unusable, then the system is not master eligible.
- A CA MIM system ‘discovers’ that it can communicate with every other defined system using a mix of CTC devices and XCF paths.
- If ANYELIGIBLE=YES is specified, then any system that can communicate with all currently active CA MIM systems is considered master eligible if all non-active systems are FREED or DISABLED. For example, assume you have ANYELIGBLE=YES and you have defined your systems with DEFSYS INITIAL=FREED. Starting CA MIM with FORMAT=BOTH or FORMAT=CF will cause the local system to assume master eligibility.
- If ANYELIGIBLE=NO is specified, then any system listed in the VCFMASTER candidate list that can communicate with all active systems is considered master eligible if all non-active systems are FREED or DISABLED.

Note: When running CA MII, you can gain the most benefit by selecting the system with the most managed enqueue activity to be the master system. The master system reads the VCF directly from its own storage, therefore it incurs no device I/O overhead, providing the best enqueue response time to the master system. You determine the number of managed enqueue requests by issuing the CA MII DISPLAY SERVICE command.

How You Define CTC Devices for Use in VCF Environments

The CTCPATH statement identifies the CTC device addresses used to transport the VCF from the master system to the client systems. As of Version 12.0, the CTCPATH statements are valid for all communication methods. Activating COMPATLEVEL=12.0 lets other communication methods use CTCPATH statements.

Conceptually, a CTC device connects an I/O address on one processor to an I/O address on another processor. VCF data that is sent from one side is received on the other side, so every transmission consists of two operations: an outbound write operation from one side, and an inbound read operation on the other side. Data may be transmitted in either direction over a CTC path, but it travels in only one direction at any one moment.

The following information describes how you implement the CTCPATH statement:

- Place the CTCPATH statements in the MIMINIT member of the CA MIM parmlib.
- Make the CTCPATH statements in the MIMINIT member the same on all systems.
- Do not place the CTCPATH statements in IFSYS/ENDIF blocks.

  During the initialization, the defined device addresses are allocated and made available for use exclusively by CA MIM.

The following information describes how you implement the CTCPATH command:

- Issue the CTCPATH command dynamically as CA MIM is executing.
- Issue the CTCPATH command before or after CA MIM is synchronized.
- You can use the CTCPATH command to make a system master eligible. From the master eligible system, add paths to every known system and from every known system add paths to the master eligible system.

The CTCPATH statement has the following parameters:

- ADDRESS = (*primaddr, altaddr1, ... altaddrn*)

  The ADDRESS parameter defines the CTC device addresses used to communicate in an outbound fashion from the system indicated in the FROMSYSTEM parameter. At least one primary address must be specified. The alternate addresses are used in recovery situations.

- FROMSYSTEM = *system name*

  (Optional) Identifies the originating system for the path. This system sends the VCF in an outbound fashion on this path.

  Note: FROMSYSTEM is an optional parameter that defaults to the local system.

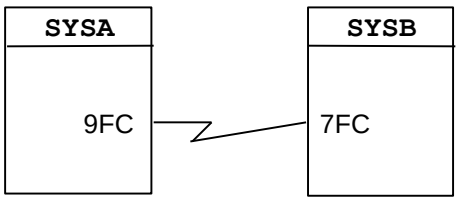
- TOSYSTEM = *system name*

  (Optional) Identifies the destination system for the path. This system receives the VCF in an inbound fashion on this path.

  Note: TOSYSTEM is an optional parameter. Omitting TOSYSTEM causes CA MIM to discover connections as the initial communication completes.

Examples

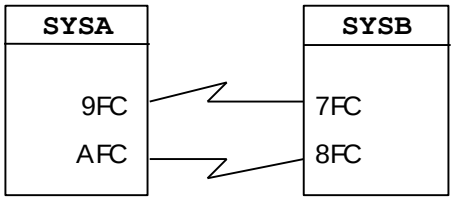
- CA MIM sends data in both directions from the master to each client system. One path is required from the master to each client system. Two CTCPATH statements are used to define the path from the master to the client, and from the client to the master. The following figure illustrates this concept:




```
CTCPATH FROMSYSTEM=SYSA ADDRESS=9FC TOSYSTEM=SYSB
CTCPATH FROMSYSTEM=SYSB ADDRESS=7FC TOSYSTEM=SYSA
```

You would place the preceding CTCPATH statements in the MIMINIT member to represent this configuration.

- The following figure illustrates the use of alternate CTC addresses for use in recovery situations:

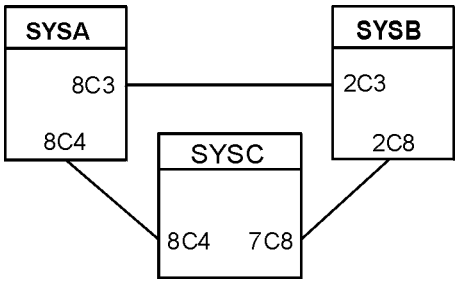



```
CTCPATH FROMSYSTEM=SYSA ADDRESS=(9FC,AFC) TOSYSTEM=SYSB
CTCPATH FROMSYSTEM=SYSB ADDRESS=(7FC,8FC) TOSYSTEM=SYSA
```

  You would place the previous CTCPATH statements in the MIMINIT member to represent this configuration.

  In this example, 9FC/7FC is the primary path, and AFC/8FC is the alternate path. The alternate path is used only when a problem is detected with the primary path. The alternate paths do not provide any performance advantages in terms of load sharing. They are for backup purposes only. Up to 15 alternate paths can be defined between any two systems. The total number of required CTCPATH statements depends on the number of defined systems in the MIMplex and whether the configuration is symmetrical or asymmetrical. We recommend symmetrical configurations because full redundancy provides the best recovery options. Master system recoverability is important in CTCONLY environments.

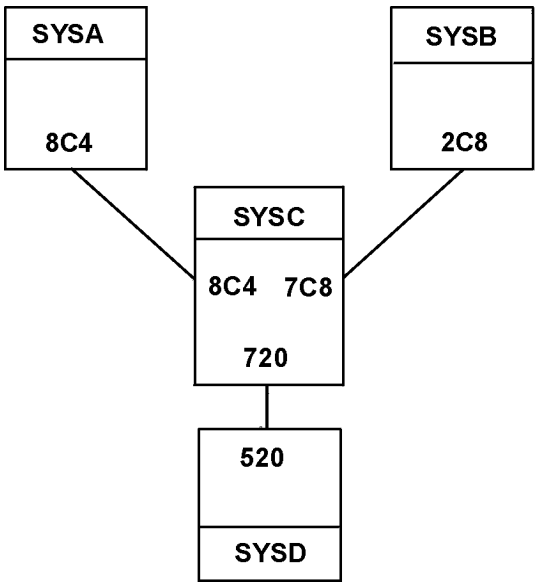
- A symmetrical configuration provides greater flexibility because the systems are connected to each other. Then, all systems are considered eligible masters. The following figure illustrates a symmetrical configuration in which all three systems are considered eligible masters:



```
CTCPATH FROMSYSTEM=SYSA ADDRESS=8C3 TOSYSTEM=SYSB
CTCPATH FROMSYSTEM=SYSB ADDRESS=2C3 TOSYSTEM=SYSA
CTCPATH FROMSYSTEM=SYSB ADDRESS=2C8 TOSYSTEM=SYSC
CTCPATH FROMSYSTEM=SYSC ADDRESS=7C8 TOSYSTEM=SYSB
CTCPATH FROMSYSTEM=SYSC ADDRESS=8C4 TOSYSTEM=SYSA
CTCPATH FROMSYSTEM=SYSA ADDRESS=8C4 TOSYSTEM=SYSC
```

You would place the preceding CTCPATH statements in the MIMINIT member to represent this configuration.

- In an asymmetrical configuration, systems are not connected to each other. At least one system is connected to every other system in the MIMplex. The following figure illustrates an asymmetrical configuration in which SYSC is the only eligible master.



```
CTCPATH FROMSYSTEM=SYSA ADDRRSS=8C4 TOSYSTEM=SYSC
CTCPATH FROMSYSTEM=SYSC ADDRRSS=8C4 TOSYSTEM=SYSA
CTCPATH FROMSYSTEM=SYSB ADDRRSS=2C8 TOSYSTEM=SYSC
CTCPATH FROMSYSTEM=SYSC ADDRRSS=7C8 TOSYSTEM=SYSB
CTCPATH FROMSYSTEM=SYSD ADDRRSS=520 TOSYSTEM=SYSC
CTCPATH FROMSYSTEM=SYSC ADDRRSS=720 TOSYSTEM=SYSD
```

The preceding CTCPATH statements are placed in the MIMINIT member to represent this configuration.

- You can omit the TOSYSTEM operand on the CTCPATH statement, which lets CA MIM discover connections as the initial communication occurs.

  Note: By omitting the TOSYSTEM operand, master eligibility is unable to be determined at initialization time. Without TOSYSTEM specified, CA MIM cannot determine connections. However, as the initial communication occurs, CA MIM reevaluates the master eligibility and notifies you of the changes.

  Complete CTCPATH Statements:

  ```
  CTCPATH FROMSYSTEM=SYSA ADDRESS=8C3 TOSYSTEM=SYSB
  CTCPATH FROMSYSTEM=SYSB ADDRESS=2C3 TOSYSTEM=SYSA
  CTCPATH FROMSYSTEM=SYSB ADDRESS=2C8 TOSYSTEM=SYSC
  CTCPATH FROMSYSTEM=SYSC ADDRESS=7C8 TOSYSTEM=SYSB
  CTCPATH FROMSYSTEM=SYSC ADDRESS=8C4 TOSYSTEM=SYSA
  CTCPATH FROMSYSTEM=SYSA ADDRESS=8C4 TOSYSTEM=SYSC
  ```

  Removing the TOSYSTEM operand reduces the CTCPATH statements:

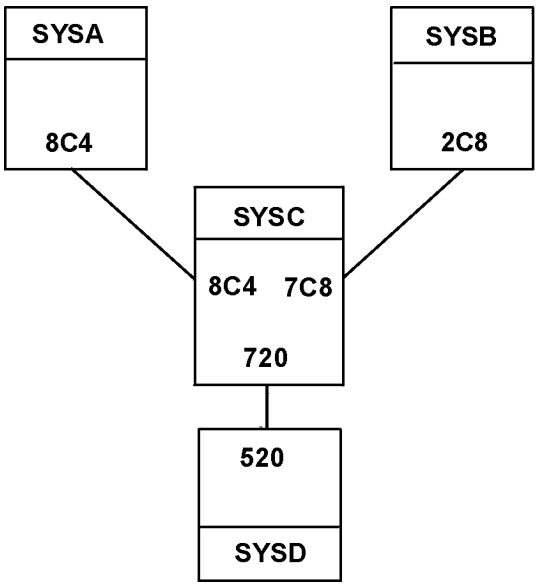
  ```
  CTCPATH FROMSYSTEM=SYSA ADDRESS=(8C3,8C4)
  CTCPATH FROMSYSTEM=SYSB ADDRESS=(2C3,2C8)
  CTCPATH FROMSYSTEM=SYSC ADDRESS=(7C8,8C4)
  ```

  During the initialization, CA MIM initiates the communication on every locally defined CTC. For example, on SYSA CA MIM initiates communications on devices 8C3 and 8C4. As external systems respond to the communications on 8C3 and 8C4, CA MIM completes its internal representation of that path by filling in the destination systems name. As the connections are completed, CA MIM reevaluates the local systems master eligibility and notifies external systems if the local system becomes master eligible.

  Note: CA MIM cannot synchronize until an eligible master is available. By removing the TOSYSTEM operand, CA MIM is unaware of eligible masters before initial communications complete. Therefore, CA MIM cannot synchronize and remains in a PENDING state. Issue the DISPLAY SYSTEMS command to identify any systems that are preventing CA MIM from starting. If NOPATH is specified, issue a FREE command to clear the NOPATH status and CA MIM then ignores that system during master eligibility evaluations. However, when doing a format start, CA MIM always honors the FREE status that is specified on the DEFSYS statements.

- You can issue the dynamic CTCPATH command to add new connections between systems or add alternate CTC devices for existing connections.

The master eligibility can change when you add connections. For example, assume SYSA, SYSB, SYSC, and SYSD are connected in the following fashion:



Dynamically adding paths from SYSA to SYSB and SYSD makes SYSA the master eligible connection.

Note: Dynamically adding a path on SYSA and SYSB creates a fully functional connection.

Virtual Control File Sizing Considerations

The VCF size is a mirror image of the primary backup DASD control file when the CA MIM complex executes in a mixed CTC and DASD environment.

The VCF image contains the same number of blocks (of size MIMINIT BLKSIZE), as the primary backup DASD control file. The primary backup DASD control file is defined as the DASD control file that was in use when the migration to the VCF took place. CA MIM supports up to 100 DASD control files. We recommend that alternate DASD control files be progressively larger in size than the primary control file. When the primary DASD control file or the VCF becomes full, a migration to a larger alternate DASD control file is successful. The larger alternate DASD control file yields an equally larger size VCF when compared to the original DASD control file size.

When running in a VCF-only environment (that is, MIMINIT COMMUNICATION=CTCONLY or MIMINIT COMMUNICATION=XCF), the VCF image size is dynamic.

VCF sizing:

- Determine the initial VCF size by multiplying the MIMINIT BLKSIZE specification by the MIMINIT VCFMAXBLOCKS specification.
- When the VCF image size needs increased, the detecting system expands the VCF dynamically.
- As each external system accesses the VCF, the system detects that the VCF has been expanded and begins operating with the increased file size.
- Depending upon the MIMINIT BLKSIZE specification, the VCF image can expand to a maximum of 2 GB.

Configure CTCPATH Statements for Disaster Recovery

Some data centers want to IPL a given system on two different CPUs. For example, consider a MIMplex with three systems – GK03, GK13, and GK62.

- CPU1 contains GK03 and GK13
- CPU2 contains GK62.

However, if CPU1 fails, then you start GK03 and GK13 on CPU2. Conversely, if CPU2 fails, then start GK62 on CPU1.

The client can code all necessary CTC device addresses in any order, as long as COMPATLEVEL=12.0 or higher.

Follow these steps:

1. Code the CTC device addresses on a single CTCPATH statement.

```
CTCPATH FROMSYS=GK03 TOSYS=GK13 ADDR=(183A,154A,B3A,C3A)
CTCPATH FROMSYS=GK03 TOSYS=GK62 ADDR=(184A,180A,B4A,C4A)

CTCPATH FROMSYS=GK13 TOSYS=GK03 ADDR=(183A,154A,B3A,C3A)
CTCPATH FROMSYS=GK13 TOSYS=GK62 ADDR=(187A,BFA,B7A,C7A)

CTCPATH FROMSYS=GK62 TOSYS=GK03 ADDR=(184A,180A,B4A,C4A)
CTCPATH FROMSYS=GK62 TOSYS=GK13 ADDR=(187A,BFA,B7A,C7A)
```

2. Review the CTCPATH statement between GK03 and GK62.
   - The first two addresses(184A,180A) are used when GK03 is on CPU1.
   - The remaining two(B4A,C4A) are used when GK03 is started on CPU2.

   The same is true for the CTCPATH statement between GK62 and GK03.
   - The first two addresses(184A,180A) are used when GK03 is on CPU2.
   - The remaining two(B4A,C4A) are used when GK03 is started on CPU1.
3. Code the CTC devices in any order.

```
CTCPATH FROMSYS=GK03 TOSYS=GK62 ADDR=(184A,180A,B4A,C4A)
CTCPATH FROMSYS=GK62 TOSYS=GK03 ADDR=(C4A,B4A,180A,184A)
```

As you can see, device #1 on GK03 is connected to device #4 on GK62. You can code the CTC devices in any order.

Control and Checkpoint File Considerations

Checkpoint files differ from control files in several ways:

- Control file data set names are specified on DD statements in the startup JCL procedure. Alternatively, control file data set names can be specified on ALLOCATE commands placed in the MIMINIT member to direct CA MIM to dynamically allocate the named control files. Checkpoint file names are constructed from a root name that you supply on the MIMINIT CHKPTDSN statement in the MIMINIT member. Alternatively, checkpoint files can be allocated by specific name using the ALLOCATE commands placed in the MIMINIT member.

  Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

  Control files are identified by a ddname of MIMTBL*nn*, where *nn* is a numeric value from 00 to 99, inclusive. Checkpoint files are identified by a ddname of MIMCKP*nn*, where *nn* is a numeric value from 00 to 99, inclusive.

- Checkpoint files may never be shared between systems, but control files must be shared. Use the IFSYS and ENDIF statements in the MIMINIT member to define unique MIMINIT CHKPTDSN statements or unique ALLOCATE commands for each system in your complex.

- If the MIMINIT CHKPTDSN statement is used to define checkpoint files, then the data sets must be named sequentially, beginning at suffix 00, and increasing contiguously (01, 02, 03, and so on). However, if ALLOCATE commands placed in the MIMINIT member are used to allocate checkpoint files, the data set names can be in any sequence. Control files may be named in any sequence, whether specified through DD card images in the startup JCL procedure, or specified through ALLOCATE commands in the MIMINIT member.

Important: CA MIM does not support control files or checkpoint files on the high portion of the Extended Address Volume (EAV). If you use EAV, make sure you allocate data sets with EATTR=NO.

Comparison of DASD Control Files and Coupling Facility Structure Control Files

CA MIM can use either DASD control files or coupling facility structure control files interchangeably in a DASDONLY or a CTCDASD communication environment. Operationally, the same commands, statements, and messages that apply to DASD control files function similarly with coupling facility structure control files.

CA MIM supports up to 100 control files in any combination of DASD or coupling facility structure. Either type of control file can be dynamically added using the ALLOCATE command, or dynamically removed using the DEALLOCATE command. The MIGRATE command can be used to switch between control file types while CA MIM is active.

However, there are differences in initial set up of the respective control file media types.

DASD Control Files

DASD control files are pre-allocated with a batch job.

Specify the size of the DASD control file and the DASD volume that is to contain the data set in the JCL. Data set placement is critical because CA MIM uses hardware reserves to serialize cross-system access to the primary DASD control file.

You can specify DASD control files in two ways:

- Through DD statements in the CA MIM JCL PROC,
- Using ALLOCATE initialization statements in the MIMINIT member of the MIMPARMS data set.

DASD control files are identified by a ddname of MIMTBL*nn*, where *nn* is a numeric value from 00 to 99.

Coupling Facility Structure Control Files

You define coupling facility structure control files in the Coupling Facility Resource Management (CFRM) policy couple data set.

- Use the IBM utility, IXCL1DSU, to allocate and format the CFRM couple data set, if one is not already in use in the parallel sysplex.
- Use the IBM utility, IXCMIAPU, to update the policy information in the CFRM couple data set.
- Issue the following z/OS command to activate the coupling facility policies:

  ```
  SETXCF START,POLICY
  ```

Note: For more information, see the IBM publication *Setting Up a Sysplex*.

You can identify coupling facility structure control files by using ALLOCATE statements in the MIMINIT member of the MIMPARMS data set. The XESFILEID keyword and the STRNAME keyword of the ALLOCATE statement uniquely specify a control file identifier and a structure name for the control file.

Create Checkpoint Files

If you want to create a checkpoint file, you must create a separate file on each system. These are not shared files like DASD control files.

To create a checkpoint file

1. Customize and submit the sample JCL found in the ALLOCKPT member of the CAI.CBTDJCL data set.
2. Specify the MIMINIT CHKPTDSN statement in the MIMINIT member to define the LOCAL checkpoint file prefix, or define the specific checkpoint file data set names using ALLOCATE commands in the MIMINIT member.

The checkpoint file is especially important when you are using the CTCONLY or XCF communication method. We recommend you allocate a primary and an alternate checkpoint file on each system in the complex. A recovery migration to the alternate checkpoint file occurs if the primary file become unusable due to hardware error or file space shortage problems. We recommend that you allocate the alternate checkpoint file to a size slightly larger than the primary checkpoint file.

To define checkpoint file prefixes

You use multiple MIMINIT CHKPTDSN statements in the MIMINIT member to define checkpoint files. The following example illustrates a two-system complex in which you have allocated primary and alternate checkpoint file names:

```
MIM.SYSA.CKPT00     (primary)
MIM.SYSA.CKPT01     (alternate)
MIM.SYSB.CKPT00     (primary)
MIM.SYSB.CKPT01     (alternate)
```

You would define the preceding checkpoint files in a shared MIMINIT member as follows:

```
IF SYSA
  MIMINIT CHKPTDSN=MIM.SYSA.CKPT
ENDIF
IF SYSB
  MIMINIT CHKPTDSN=MIM.SYSB.CKPT
ENDIF
```

To define specific checkpoint files

As an alternative to defining checkpoint prefixes, you can use multiple ALLOCATE commands in the MIMINIT member, qualified by IF/ENDIF statements, to define specifically named checkpoint files. The following illustrates a two-system complex in which you have allocated primary and alternate checkpoint file names:

```
MIM.SYSA.PRIMCKPT (primary)
MIM.SYSA.ALTCKPT (alternate)
MIM.SYSB.PRIMCKPT (primary)
MIM.SYSB.ALTCKPT (alternate)
```

You would define the preceding checkpoint files in a shared MIMINIT member as follows:

```
IF SYSA
   ALLOCATE DDNAME=MIMCKP00 DSNAME=MIM.SYSA.PRIMCKPT
   ALLOCATE DDNAME=MIMCKP01 DSNAME=MIM.SYSA.ALTCKPT
ENDIF
IF SYSB
   ALLOCATE DDNAME=MIMCKP00 DSNAME=MIM.SYSB.PRIMCKPT
   ALLOCATE DDNAME=MIMCKP01 DSNAME=MIM.SYSB.ALTCKPT
ENDIF
```

How You Allocate Control and Checkpoint Files

You need to allocate DASD control files only if you are planning to use either DASDONLY or CTCDASD as your communication method.

You can use coupling facility structure control files in addition to or in place of DASD control files in communication environments that require DASD control files, such as the DASDONLY or CTCDASD communication methods.

You also need to allocate DASD checkpoint files, if either one of the following is true:

- You will be using the CTCONLY or XCF communication methods. These environments use the checkpoint file to track system status information.
- You will be using the REQUEUE feature of the ENQ Conflict Management Facility (ECMF). ECMF uses the checkpoint file to track job requeues across system starts and stops.

DASD Control File Placement

Where you place the CA MIM DASD control file can impact the performance of CA MIM, and subsequently, the performance of your entire complex. Each CA MIM address space serializes access to the control file volume through z/OS RESERVE/RELEASE processing.

Use the following guidelines when deciding on which DASD unit to place CA MIM Control Files:

- Do select a device that has no other data sets on it.
- Do select a device that is accessible by all systems in the MIMplex.
- Do select a device that is currently supported by IBM or other hardware vendors.
- Do select a device that is connected by four to eight channel paths from each MIMplex LPAR.
- Do select a device that is connected via FICON channel paths, rather than ESCON channel paths.
- Do select a device whose channel paths are not overly busy.
- Do select a device that has been excluded from CA ASTEX Performance I/O monitoring.
- Do select a device that has been excluded from CA Vantage SRM housekeeping VTOC scans.
- Do select a device that has been excluded from full-volume backup utility services.
- Do select a device that has been excluded from synchronous or asynchronous I/O mirroring.
- Do not select a device that is subject to dynamic PAVing and Multiple Allegiance processes in both of the following circumstances:
  - The MIMplex spans multiple sysplexes and
  - The device is defined to a single logical control unit (LCU)

Placing the CA MIM control file on DASD units that do not meet these criteria can negatively impact CA MIM performance and degrade MIMplex-wide system throughput.

Allocate DASD Control Files

Allocate the DASD control file by customizing the sample JCL in the ALLOCCF member of the CAI.CBTDJCL data set. Do not use any other method to allocate CA MIM control files.

Keep the following in mind when allocating DASD control files:

- Allocate control files as contiguous cylinders and on cylinder boundaries.
- The *primary* control file should be a minimum of 10 cylinders in size.

- Allocate at least one *alternate* control file to be used as a backup in case the primary control file becomes unusable. You may allocate up to 99 alternate control files.
- Allocate alternate control files to a size slightly larger than the primary control file.
- Do not place alternate control files on the same device as the primary control file.
- If possible, allocate the file in a location that requires minimal or no head movement.

Follow these steps:

1. Issue the following command to determine the next available control file number:

   ```
   DISPLAY FILES
   ```

2. Create a new control file by customizing the sample JCL in the ALLOCCF member of the CAI.CBTDJCL data set.
3. Issue the following command on all systems in the MIMplex:

   ```
   ALLOCATE DSNAME=newdsn,DDNAME=MIMTBLnn
   ```

   nn

   Specifies the number of the new control file.

4. To verify that the new file was added to the control file list, reissue the following command:

   ```
   DISPLAY FILES
   ```

Allocate Checkpoint Files

Checkpoint files are not shared like DASD control files so you must allocate a unique checkpoint file for each system. You can allocate checkpoint files on non-shared DASD volumes and place them on volumes with other data sets. Checkpoint files are accessed much less frequently than control files.

You use the ALLOCATE command to allocate new or relocated DASD checkpoint files. Checkpoint files are not shared files like DASD control files, so you must allocate a unique checkpoint file for each system.

To allocate a checkpoint file

1. Issue the following command to determine the next available checkpoint file number identified by the checkpoint file prefix (MIM.SYSx. MIMCKP):

   ```
   DISPLAY FILES
   ```

2. Customize the sample JCL found in the ALLOCKPT member of the CAI.CBTDJCL data set to create a new checkpoint file.

3. Issue the following command:

   ```
   ALLOCATE DSNAME=newdsn,DDNAME=MIMCKPnn
   ```

   nn

   Specifies the number of the new checkpoint file.

4. To confirm that the new file was added to the checkpoint file list, reissue the following command:

   ```
   DISPLAY FILES
   ```

Allocate XES Control File Placement

You use the XESFILEID and STRNAME parameters on the ALLOCATE command to allocate any new coupling facility structure control file for the use of CA MIM. Use the XESFILEID parameter on the DEALLOCATE command to remove a coupling facility structure control file from the use of CA MIM.

To allocate a XES control file

1. Issue the following command to determine the next available control file number:

   ```
   DISPLAY FILES
   ```

   See the following note regarding pre-defining a coupling facility list structure.

2. Issue the following command on all systems in the MIMplex:

   ```
   ALLOCATE XESFILEID=nn,STRNAME=newstructure
   ```

   nn

   Specifies the number of the new control file.

   newstructure

   Specifies the name of the coupling facility structure to be used as a CA MIM control file

3. Reissue the following command to confirm that the new file was added to the control file list.

   ```
   DISPLAY FILES
   ```

Notes:

- You must have defined the coupling facility structure, MIMGR#TABLE02, in a policy in the CFRM (Coupling Facility Resource Management) couple data set, and the policy must have been previously activated.
- For detailed information regarding the activation of a CFRM policy, see the IBM publication *Setting Up a Sysplex*.

How You Use XES Control Files

After you have allocated coupling facility structure control files, you need to update the Coupling Facility Resource Management policy to define the names and attributes of the control file structures.

For detailed information on this process, see the IBM publication *Setting Up A Sysplex*.

- To determine the actual coupling facility resources available to the system, issue the following command:

  ```
  z/OS DISPLAY XCF,CF,CFNAME=ALL
  ```

- To determine the active CFRM couple data, issue the following command:

  ```
  z/OS DISPLAY XCF,COUPLE,TYPE=CFRM
  ```

- To determine the active CFRM policy, issue the following command:

  ```
  z/OS DISPLAY XCF,POLICY,TYPE=CFRM
  ```

You may need all of the information in the preceding displays to update the sample CFRM policy update job.

When defining structure control files in the CFRM policy, keep the following in mind:

- The primary structure control file should be at least 10 MB.
- The alternate structure control file should be slightly larger than the primary structure control file.
- The primary structure control file should not be allocated in the same coupling facility as the alternate structure control file.
- Structure control file names can be one to sixteen characters long. The first character must be alphabetic and the remaining characters can be alphanumeric or national (@, #, $) characters. CA MIM does not support the underscore (_) character in structure control file names.

To add a policy for the coupling facility structure control files

1. Update the CFRM couple data set by customizing and running the sample JCL in the ALLOCSTR member of the CAI.CBTDJCL data set. Note that the JCL is a sample. You may need to refer to your active CFRM policy to be able to build a complete new CFRM policy that includes the CA MIM structure control files.
2. Define the coupling facility structure control files by place the appropriate ALLOCATE initialization statements in the MIMINIT member of the MIMPARMS data set.

For example, to specify the coupling facility structure control files defined in the preceding example, place the following ALLOCATE statements in the MIMINIT member:

```
ALLOCATE XESFILEID=00 STRNAME=MIMGR#TABLE00
ALLOCATE XESFILEID=01 STRNAME=MIMGR#TABLE01
```

If you already have DASD control files specified, and you want to add coupling facility control files in addition to the existing DASD control file, consider the following example placed in the MIMINIT member of the MIMPARMS data set:

```
ALLOCATE DDNAME=MIMTBL00 DSNAME=MIMGR.TABLE00
ALLOCATE DDNAME=MIMTBL01 DSNAME=MIMGR.TABLE01
ALLOCATE XESFILEID=02 STRNAME=MIMGR#TABLE00
ALLOCATE XESFILEID=03 STRNAME=MIMGR#TABLE01
```

Note: The MIMTBL*nn* value and the XESFILEID*nn* value cannot be duplicated, because these values are used to identify specific control files in commands and messages during the operation of CA MIM. For this reason, any duplication of the file identifier *nn* is treated as an error.

Control File Size Considerations

In a synchronized MIMplex, only the first one or two blocks of the control file are used. The largest usage of control file space usually occurs during MIMplex transition states, such as when a new CA MIM address space is joining an existing MIMplex or when a control file is being migrated. During these transition states, CA MIM may generate a large volume of transactions to communicate the current status of the resources managed by each component to the systems requiring this information.

Note: For more information, see Control File Blocks and the Global-Copy Process (see page 125).

The CA MIM control file must be large enough to accommodate the number of transactions generated during these transition states. The amount of space required during these transition states depends on the components active in the MIMplex, and also may depend on workload.

- For CA MIA and CA MIC, a primary control file size of 10 MB is usually sufficient.
- For CA MII component, the space required during these transition periods depends on the number of concurrently held managed ENQ resources. This number varies widely by installation and can be a changeable value.

To help you determine the correct size of the control file for MIMplexes running CA MII, issue the following command to calculate the global copy space required based on the peak number of concurrently held ENQ resources since product startup:

```
DISPLAY GDIF CFSIZE
```

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

Change the Size of the Virtual Control File Transfer Buffer

The virtual control files use a default transfer buffer of 32,768 bytes to accommodate cross-system transactions.

To change the size of the VCF transfer buffer

1. Specify a value for the VCFBUFFERSIZE parameter on either of the following:
   - The MIMINIT statement on the PARM parameter of the startup procedure
   - The z/OS START command for CA MIM.
2. Specify the same value on all systems.

The lowest possible size is seven bytes more than the current block size. The current block size is displayed with the DISPLAY IO command. The highest possible size is 62,464 bytes. The value for the VCFBUFFERSIZE parameter should be large enough to accommodate the average number of blocks read per cycle.

Example: VCF transfer buffer size

To increase the size of this buffer to 35,000 bytes, issue the following statement:

```
MIMINIT VCFBUFFERSIZE=35000
```

Important! When the buffer size is less than the average number of bytes read per service cycle, CA MIM transmits cross-system transactions too slowly. If you specify too large a buffer size, an excessive amount of real storage is used. To see how many blocks are read per service cycle, issue the following command:

```
DISPLAY IO
```

This command also displays other types of I/O statistics for the CA MIM control file.

How CA MIM Formats Control and Checkpoint Files

The CA MIM control and checkpoint files are allocated without specifying RECFM, LRECL, BLKSIZE, or the DSORG attributes. This allocation is intentional, because CA MIM does not use standard access methods (for example, VSAM or QSAM) to perform I/O operations to these files. Instead, it provides a special formatting routine, which prepares the file for access by CA MIM channel programs. CA MIM builds its own channel programs to perform I/O operations to DASD and checkpoint files. The same is true for CTC I/O operations. CA MIM uses Cross System Extended Services (XES) to access coupling facility structure control files.

In addition to structuring the files, the formatting logic creates a special record that is called the Global Management Record (GMR). The GMR is built primarily on DEFSYS, GLOBALVALUE, and the MIMINIT values and describes the systems and recovery options that define the MIMplex. The basic components of this record are created only during the formatting process.

The GMR resides in the DASD control file or coupling facility structure control file when you use the DASDONLY or CTCDASD communication methods. The GMR resides in the checkpoint file when you use the CTCONLY or XCF communication methods.

A FORMAT startup is required under the following circumstances:

- You are planning to reference new, unformatted DASD or coupling facility structure, or checkpoint files for the first time
- You are planning changes to the DEFSYS statement
- You are planning changes to the MIMINIT statement using these parameters:
  - BLKSIZE
  - COMMUNICATION
  - GCMF
  - GDIF
  - GTAF
  - ICMF
  - VCFMAXBLOCKS

How You Migrate to a New Control File

Control file migration is the process by which CA MIM leaves its current control file and begins using another control file to communicate globally.

Any system in a synchronized MIMplex can initiate migration to an alternate file. Migrations may be initiated by command or because of a recovery situation.

While CA MIM is executing, you can migrate between physical and virtual control files, provided you meet the following criteria:

- At least one physical (DASD or coupling facility list structure) control file is allocated to CA MIM. You can specify ALLOCATE statements in your MIMINIT member, or you can issue the ALLOCATE command dynamically while CA MIM is executing.
- At least one system is master eligible. A master eligible system has XCF or CTC connectivity to every other system in the MIMplex. You can dynamically add CTC devices using the CTCPATH command to complete connectivity while CA MIM executes.

CA MIM may automatically migrate to a new control file as part of error recovery in the following cases:

- If the current control file becomes unusable.
- When migrating from one virtual control file to another in a CTCDASD environment, CA MIM migrates to the DASD control file or coupling facility structure control file for a few seconds while re-synchronization takes place. After synchronization, the virtual control file automatically migrates to the new virtual control file.

Migrate to a DASD Control File

While CA MIM is executing you can migrate to any properly formatted DASD control file that is allocated to all systems in your MIMplex. Migration to a DASD control file occurs in one of several ways:

- From a virtual control file to a DASD control file
- From one DASD control file to another DASD control file
- From a coupling facility structure control file to a DASD control file

To migrate to a DASD control file:

1. Issue the following command to view the list of available control files:

   ```
   DISPLAY FILES
   ```

2. Issue the following command to initiate the migration to a specific file:

   ```
   MIGRATE CONTROLFILE=nn
   ```

   nn

   Specifies the control file ID of the DASD control file

To migrate CA MIM to the next available control file

1. **Issue the following command:**

   ```
   MIGRATE CF
   ```

Note: When you migrate from a virtual control file to a DASD control file, the virtual control file is deactivated automatically.

Migrate to a Coupling Facility Structure Control File

You can migrate to a coupling facility structure control file in the same way that you migrate to a DASD control file. To view the list of available control file IDs, issue the DISPLAY FILES=CF command. Migration to a coupling facility structure control file occurs in one of these ways:

- From a virtual control file to a coupling facility structure control file
- From one coupling facility structure control file to another coupling facility structure control file
- From a DASD control file to a coupling facility structure control file

To migrate to a coupling facility structure control file:

1. Issue the following command to view the list of available control files:

   ```
   DISPLAY FILES
   ```

2. Issue the following command to initiate the migration to a specific file:

   ```
   MIGRATE CONTROLFILE=nn
   ```

   nn

   Specifies the control file ID of the coupling facility structure control file

You do not have to migrate to the first or next control file identified in the startup procedure or listed in the DISPLAY FILES=CF command response. You can migrate to any usable control file, even if the ID of the control file is lower than the ID for the current control file.

To migrate to the next available control file issue the following command:

```
MIGRATE CF
```

Note: When you migrate from a virtual control file to a coupling facility structure control file, the virtual control file is automatically deactivated.

Migrate to a Virtual Control File

You can use the MIGRATE command to migrate from a physical control file to a virtual control file. Virtual control file communication is available if there is an active, fully connected, eligible master system in your MIMplex.

To migrate to a virtual control file:

1. Issue the following command to view the list of eligible master systems:

   ```
   DISPLAY GLOBALVALUE
   ```

2. Issue the following command to migrate to using a virtual control file:

   ```
   MIGRATE MASTER=sysid
   ```

   sysid

   Specifies the system ID (sysid) of the new master system as specified on a DEFSYS statement.

To migrate to the next preferred master system

Issue the following command:

```
MIGRATE MASTER
```

How You Replace Control Files

This section discusses replacing control files.

Replace DASD Control Files

To replace a DASD control file, de-allocate that file using the DEALLOCATE command and then allocate a new file using the ALLOCATE command. You must issue these commands on all systems in the complex.

For example, suppose that the control file identified on the //MIMTBL03 DD statement in your startup procedure becomes unusable, and you want to replace it.

To replace DASD control files

1. De-allocate the control file by issuing the following command:

   ```
   DEALLOCATE DDNAME=MIMTBL03
   ```

2. Physically allocate a new control file by customizing and submitting the sample JCL in the ALLOCCF member of the CAI.CBTDJCL data set.

3. To make file MIM.NEW03 available to the CA MIM address space, issue the following command:

   ```
   ALLOCATE DDNAME=MIMTBL03 DSNAME=MIM.NEW03
   ```

4. Repeat Step 3. on all systems, using the new DDNAME and DSNAME.

The ALLOCATE command formats the control file automatically, so you do not need to restart CA MIM.

Notes:

- You cannot replace a DASD control file that you are currently using. You need to migrate to a new control file first and then replace the control file that you abandoned.
- For more information about migrating between DASD control files, see Migrate to a DASD Control File (see page 65).

Replace Coupling Facility Structure Control Files

To replace a coupling facility structure control file, you deallocate that file using the DEALLOCATE command, then allocate a new file using the ALLOCATE command. You must issue these commands on all systems in the MIMplex.

For example, assume you want to remove a coupling facility list structure named MIMGR#TABLE02, with a control file ID of 02, from the use of CA MIM because you want to replace it with a new structure in a different coupling facility.

To replace the coupling facility structure control files in this example

1. Deallocate the control file by issuing the following command on all members in the MIMplex:

   ```
   DEALLOCATE XESFILEID=02
   ```

2. Define a new structure for CA MIM to use by updating the Coupling Facility Resource Management (CFRM) couple data set with a new policy and activate that policy.

   Note: For detailed information on the process, see the IBM publication *Setting Up a Sysplex*. You may want to see member ALLOCSTR in the CAI.CBTDJCL data set for sample JCL that can be used to update the CFRM policy.

3. Make the new coupling facility structure control file available to the CA MIM address space by allocating it to all members in the MIMplex. Assuming that the new structure name is MIMGR#TABLE03 and the file ID to be assigned to the control file is 03, issue the following command on all members of the MIMplex:

   ```
   ALLOCATE XESFILEID=03 STRNAME=MIMGR#TABLE03
   ```

You must specify both the file ID and the structure name of the new control file on the ALLOCATE command on all systems in the MIMplex. The ALLOCATE command formats the control file automatically, so you do not need to restart CA MIM.

Note: To replace a coupling facility structure control file that is currently the active control file, first you need to migrate to a new control file and then deallocate the abandoned coupling facility structure control file from all members of the MIMplex. Next, issue the SETXCF FORCE,STRUCTURE command to delete the XES structure, as the DEALLOCATE XESFILED command does not automatically do so. Once this is complete, the replacement XES structure can be defined in the new CFRM policy and activated.

More information:

How You Migrate to a New Control File (see page 64)

Coupling Facility Structure System-managed Rebuild

CA MIM provides full support for system-managed rebuild of coupling facility structure control files. This feature is supported when CA MIM is executing on any z/OS operating system and all CA MIM MIMplex members are executing at a supported release of CA MIM. This means that CA MIM continues to operate while the system-managed rebuild process moves a coupling facility structure control file from one coupling facility to a new coupling facility.

You start this process by issuing the following command:

```
SETXCF START,REBUILD
```

This process does not require any special CA MIM commands or procedures preceding or following the system-managed rebuild operation.

Notes:

- For information regarding the system-managed rebuild function of a coupling facility, see the IBM publication *Setting Up a Sysplex*.
- IBM coupling facility system-managed rebuild is a planned, operator-initiated function for moving coupling facility resources. Do not invoke it for coupling facility hardware or coupling facility link types of failures. CA MIM continues to react to a coupling facility structure control file error by marking the file in error and initiating a control file migration in the case of an error on the IN-USE control file.
- You cannot use the MVS REBUILD command to change the size of a CA MIM list structure. Instead, use the DEALLOCATE and ALLOCATE commands.

Product Activation Considerations

At startup time, CA MIM examines the system IDs in its DEFSYS statement to see which systems will be joining the MIMplex. Synchronization does not complete until you start CA MIM on each system identified in the control file.

- To tell CA MIM to ignore a system ID for this execution because the system is not joining the complex, issue a FREE command for that system.
- To permanently remove the ID for a system that will not join the MIMplex again, issue the REMOVE command for that system.

Startup Activities

CA MIM is designed to run on two or more systems sharing a common control file, therefore, it will not synchronize unless at least two systems are communicating with the control file. An exception is made for cases where all eligible systems have been FREED from the control file, in which case, it is possible for a single system to start-up and synchronize.

If you want to start a single system for test purposes, be sure to include a DEFSYS statement in the initialization member, and define a second system on the DEFSYS statement. When the starting system reaches “PENDING” status, issue a FREE command (supplying the name of the second system for the SYSNAME parameter). If you use a REMOVE command instead of a FREE command, then the system will not reach synchronization, because it will appear that the second system was never defined.

CA MIA Startup Activities

By design, CA MIA serializes z/OS allocation across multiple systems. Therefore, we recommend that CA MIA should be started *before* any tape allocation is allowed to occur on any system in the MIMplex.

For CA MIA to successfully start, at least one device must be defined to the product. If no devices are defined to CA MIA, it issues a MIM2003 NO MANAGED DEVICES message when started, then abends with a U0040 RC=66 at startup.

GTAF and TPCF do not vary devices online or change the status of a device unless you specified VARY commands in the parameter data set and synchronization has completed.

GTAF delays tape allocations on an active system until synchronization is complete.

CA MIC Startup Activities

CA MIC allocates all subsystem consoles specified on the GCMINIT CONSLIST statement.

CA MII Startup Activities

We strongly recommend that GDIF be started *before* any other products or subsystems that issue GDIF- managed ENQ requests because:

- GDIF propagates all ENQ requests outstanding at startup time, but it cannot eliminate outstanding hardware reserves.
- Neither GDIF nor EDIF can prevent damage from an outstanding integrity exposure.
- We recommend that GDIF be started using the Early Start Mechanism which is outlined in the *CA MIM Installation Guide*.

How You Override Initialization Values at Startup Time

To override values defined on the MIMINIT statement or on the PARM parameter of the EXEC statement of the startup procedure, specify new values on the z/OS START command. All of the parameters that you can specify on the z/OS START command are found on the MIMINIT statement.

You can override a parameter only if you defined a symbolic variable for that parameter on the PARM parameter of the startup procedure, and you specified that parameter on the PROC statement of the startup procedure. Also, you must specify the symbolic variable as a parameter on the z/OS START command, rather than specifying name of the parameter (unless the name of the parameter is the same as its symbolic variable).

For example, suppose you use the symbolic variable CMNDS to represent the COMMANDS parameter on the PROC statement in your startup procedure. To make CA MIM use the member named NEWCMNDS during this execution (instead of the member identified on the CMNDS variable in the start-up procedure), you can specify CMNDS=NEWCMNDS on the z/OS START command:

```
S MIMGR,CMNDS=NEWCMNDS
```

CMNDS=NEWCMNDS overrides the variable for the CMNDS parameter on the PROC statement because the PROC statement recognizes CMNDS as a symbolic variable. If you specify COMMANDS=&CMNDS in the PROC statement, then the PROC statement substitutes the value COMMANDS=NEWCMNDS.

The SYSID Override Parameter

Each z/OS or z/VM image in your CA MIM complex has a name, an index, and an alias, all of which are used in CA MIM displays and for other CA MIM purposes. The MIMINIT statement defines the name, index, and alias. The statement must name all systems in the complex, because the order of the names in the statement determines the index value.

The MIMINIT statement typically uses the SMF ID (in z/OS systems) or the SYSID (in z/VM systems) to determine which system name, index, and alias it should assign. For example, the statement:

```
MIMINIT DEFSYS (FIRST,FF,SY22) (SECOND,SS,SY23)
```

assigns name=FIRST, alias=FF and index=1 to the system having SMF ID SY22, and SECOND, SS and index=2 to the system having SMF ID SY23.

The CA MIM system name is usually made equal to the SMF ID. The following statement makes the CA MIM ID and the SMF ID the same for SY22 and SY23:

```
MIMINIT DEFSYS (SY22,FF,SY22) (SY23,SS,SY23)
```

This is a convenient and recommended convention.

It is possible to override the SMF ID, for DEFSYS and IFSYS purposes, by using the MIMINIT SYSID statement. If used, the MIMINIT SYSID statement must be positioned before the MIMINIT DEFSYS statement in the INIT member. We recommend you use SYSID only for testing purposes. If you choose to use SYSID, then make sure you do not accidentally define the same system twice, either by sharing the INIT member, or copying the INIT member to other systems.

Having two or more systems in your CA MIM complex specify the same SYSID causes control file errors, and CA MIM will abend. Consider temporarily using a SYSID statement in the following circumstances:

- You need to override the system name for a single execution of CA MIM
- You did not specify a DEFSYS statement, and you want to define a system name that is different from the SMF ID of the system
- The SMF ID of the system is incorrect due to an error during the IPL procedure

Note: For more information about using the SYSID parameter on the MIMINIT statement, see the discussion of the MIMINIT Statement in the *Statement and Command Reference Guide*

Suppose you are running CA MIM on two systems and instead of using the SMF ID of the system, you want to use the system name SYSA for the local system. To do this, specify SYSID=SYSA on the PARM parameter of the startup procedure for the local system. Or, issue the following z/OS START command from the local system (if the name of the started task is MIMGR):

```
S MIMGR,SYSID=SYSA
```

Note: When you override the name of a system, you should check all CA MIM parmlib members for statements or commands that reference system names. Here are some examples:

```
IFSYS
CTCPATH
GLOBALVALUE
LINK
COLLECT
```

If the CTCPATH and GLOBALVALUE statements are incorrect, then they can prevent CA MIM from initializing at startup. If the system names are not correct, then you can issue the LINK and COLLECT commands to create new dynamic linkages and collection sets.

The FORMAT Override Parameter

A FORMAT parameter is defined in the MIMPROC on the PROC statement. The default value specified in the MIMPROC is NONE. The following scope parameters are valid for FORMAT:

BOTH

Formats all control files and the local checkpoint files.

CF

Formats all control files.

CKPT

Formats only the checkpoint information in the checkpoint file. You can use the abbreviation CKPT for this operand.

NONE

No formatting is done.

Again, when control files, checkpoint files, or both are formatted, CA MIM acquires information from the parmlib and creates a GMR representing various aspects of the MIMplex. The location of the GMR depends on the CA MIM communication method selected. Therefore, proper formatting depends on the communication method selected and whether checkpoint files exist.

Format DASDONLY or CTCDASD

If the following are true:

- The GMR resides in the DASD control file or coupling facility structure.
- Checkpoint files are not required, and you are using the DASDONLY or CTCDASD communication method with the REQUEUE feature of ECMF not active.

Then do the following:

1. Stop CA MIM on all systems in the MIMplex by issuing the following command:

   ```
   @SHUTDOWN,GLOBAL
   ```

2. Start CA MIM on the first system by issuing the following command:

   ```
   S MIMGR,FORMAT=CF
   ```

3. Start CA MIM on all other systems normally by issuing the following command on each system:

   ```
   S MIMGR
   ```

If checkpoint files are required, and you are using the DASDONLY or CTCDASD communication method with the REQUEUE feature of ECMF active, us the following procedure:

1. Stop CA MIM on all systems in the MIMplex by issuing the following command:

   ```
   @SHUTDOWN,GLOBAL
   ```

2. Start CA MIM on the first system by issuing the following command:

   ```
   S MIMGR,FORMAT=BOTH
   ```

3. Start CA MIM on all other systems normally by issuing the following command on each system:

   ```
   S MIMGR,FORMAT=CKPT
   ```

Format CTCONLY or XCF

If the following items are true:

- The GMR resides in the checkpoint files.
- Checkpoint files are required, as no DASD control files exist.

Then, use the following procedure:

1. Stop CA MIM on all systems in the MIMplex:

   ```
   @SHUTDOWN GLOBAL
   ```

2. Start CA MIM on the first system and format both the virtual control file image, as well as the checkpoint file:

   ```
   S MIMGR,FORMAT=BOTH
   ```

   The following message is issued on the first system:

   ```
   MIM0359 VCF FORMAT COMPLETE - MIM CAN BE STARTED ON OTHER SYSTEMS
   ```

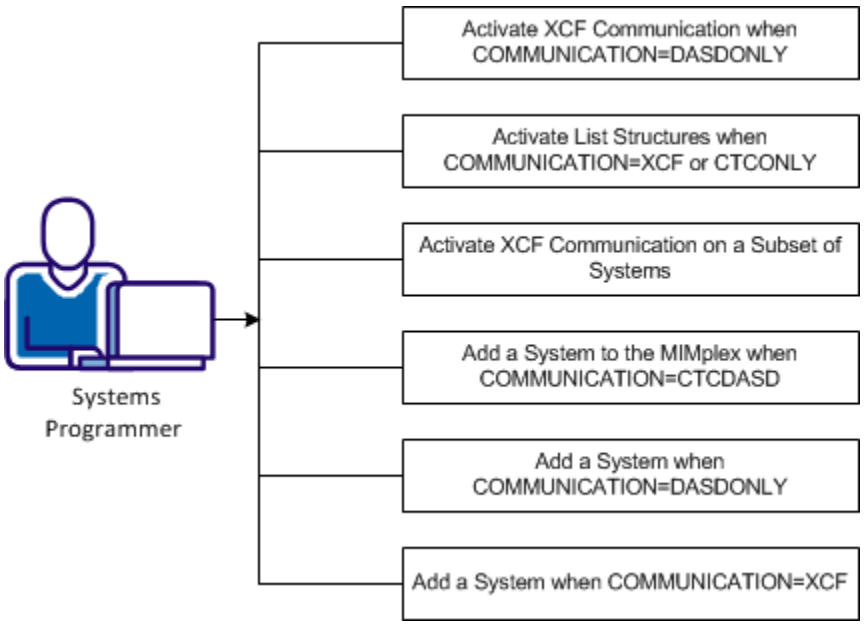
3. Start CA MIM on all other systems and format their checkpoint files:

   ```
   S MIMGR,FORMAT=CHKPT
   ```

How You Dynamically Change CA MIM Communication Methods

Global ENQ response time is dependent on the CA MIM communication method. As a z/OS systems programmer, using an alternate method can improve response time. By supporting dynamic configuration changes, you can more easily try different configurations and measure the results.

Equation 1: How you dynamically change CA MIM communication methods



The following is a list of supported dynamic procedures.

- Activate XCF Communication when COMMUNICATION=DASDONLY (see page 77)
  - Revert to the Original Communication Method (see page 79)
- Activate List Structures when COMMUNICATION=XCF or CTCONLY (see page 80)
  - Revert to the Original List Structures (see page 82)
- Activate XCF Communication on a Subset of Systems (see page 83)
  - Revert to CTC Communication (see page 88)
- Add a System to the MIMplex when COMMUNICATION=CTCDASD (see page 90)
  - Revert to the Original MIMplex (see page 94)
- Add a System when COMMUNICATION=DASDONLY (see page 95)
- Add a System when COMMUNICATION=XCF (see page 96)

Activate XCF Communication when COMMUNICATION=DASDONLY

This procedure dynamically activates the XCF communication and works with either DASD control files or XES list structures. The MIMplex is three systems using XES list structures with the MIM system names of GK03, GK13, and GK62.

Note: Be sure that your COMPATLEVEL is set to 12.0 or higher.

The following steps demonstrate how to dynamically activate the XCF communication when COMMUNICATION=DASDONLY.

Follow these steps:

1. Set up your XCF communication:
   - Confirm that all systems in the MIMplex are also within the same sysplex.
   - Set MIMINIT VCFBUFFERSIZE to 32768 or larger.
   - Review the XCF configuration and options in the SYS1.PARMLIB(COUPLExx) member.
   - You do not need to define a special transport class to CA MIM. However, we recommend at least one CLASSLEN of 32768 or larger to avoid possible performance problems. CLASSLEN is an XCF parameter and is coded in the SYS1.PARMLIB(COUPLExx) member.
   - If some of the LPARS are geographically remote, then IBM recommends that XCF use both CTC and list structures.

   Note: You do not need to create the Checkpoint files. However, CA MIM has no issues when you are already using them.
2. Activate your XCF communication.
   a. Verify that CA MIM automatically created your XCF paths on XE03:

```
@D PATH
MIM0067I Command DISPLAY 960
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  14:30:23 ON 2013.171
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 XCF  GK62     USABLE                  0        0                    0
 XCF  GK13     USABLE                  0        0                    0
 N/A   GK03    LOCAL                   0        0                    0
```

   b. Verify that CA MIM automatically created your XCF paths on GK13:

```
@D PATH
MIM0067I Command DISPLAY 444
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  14:30:45 ON 2013.171
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 XCF  GK62     USABLE                  0        0                    0
 N/A  GK13     LOCAL                   0        0                    0
```

```
 XCF  GK03     USABLE                  0        0                    0
```

c. Verify that CA MIM automatically created your XCF paths on GK62:

```
@D PATH
MIM0067I Command DISPLAY 837
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  14:30:53 ON 2013.171
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK62     LOCAL                   0        0                    0
 XCF  GK03     USABLE                  0        0                    0
 XCF  GK13     USABLE                  0        0                    0
```

CA MIM automatically creates a GLOBALVALUE, when not already coded in the INIT member:

```
@D GLOBALVALUE
MIM0067I Command DISPLAY 956
MIM0373I MIM GLOBALVALUE display:
   ANYELIGIBLE=YES       MOSTPREFERRED=NO      NOMASTER=WAIT
   VCFMASTER=NONE
   ELIGIBLE MASTER LIST: GK03, GK13, GK62
```

3. Order the VCFMASTER list from the busiest system to the least busy system.

   Performance is improved as it reduces the total amount of I/O.

4. Enter the DISPLAY IO command on all systems and pick the system with the largest RATE: CYC.

   In this example, we have determined that GK62 is the busiest, followed by GK13, followed by GK03.

5. Enter the GLOBALVALUE command on one system only:

```
@GLOBALVALUE VCFMASTER(GK62,GK13,GK03)
MIM0067I Command GLOBALVALUE 263
MIM0193I GLOBALVALUE command processing initiated
MIM0363I GLOBALVALUE command changes pending on external systems
MIM0362I GLOBALVALUE command processing complete
```

6. Verify the new VCFMASTER list:

```
@D GLOBALV
MIM0067I Command DISPLAY 273
MIM0373I MIM GLOBALVALUE display:
   ANYELIGIBLE=YES       MOSTPREFERRED=NO      NOMASTER=WAIT
   VCFMASTER=( GK62, GK13, GK03 )
   ELIGIBLE MASTER LIST: GK62, GK13, GK03
```

7. Update the initialization member with the new GLOBALVALUE statement.

8. Use the MIGRATE command and move the control file up to the VCF on GK62.

```
@MIGRATE MASTER=GK62
MIM0067I Command MIGRATE 307
MIM0232I VCF migration SCHEDULED   - MASTER=GK62
```

```
MIM0377I VCF migration IN PROGRESS - MASTER=GK62
MIM0342I system GK62 VCF activation      COMPLETE - MASTER=GK62
```

9. Confirm that the Virtual Control file is active on GK62:

```
@D SYS
MIM0067I Command DISPLAY 211
MIM0108I SYSTEMS DISPLAY
   INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
    01    03   GK03     EXTERNAL ACTIVE      zOS  2013.171 14:54:51.19
    02    13   GK13     EXTERNAL ACTIVE      zOS  2013.171 14:54:51.14
    03    62   GK62     LOCAL    MASTER      zOS  2013.171 14:54:51.22
```

The XCF communication is activated and GK62 is now the VCFMASTER.

Revert to the Original Communication Method

You can revert CA MIM to the original DASD file or XES list structure using the MIGRATE command.

Follow these steps:

1. Use the MIGRATE command:

```
@MIG CF=00
MIM0067I Command MIGRATE 531
MIM0227I Migration initiated to control file 00
MIM0088I migration to file 00 IN PROGRESS
MIM0242I system GK62 VCF deactivation    COMPLETE
MIM0080I migration to file 00 COMPLETE
MIM0638I Global Copy Received
```

2. Display the files and confirm that the original DASD file or XES structure is in-use:

```
@D FILES
MIM0067I Command DISPLAY 610
MIM0102I Control File Display:
   File Unit Status  Volser   Data Set
    00       IN-USE  CFCCMIM2 KERGL01#TABLE00
    01       USABLE  CFCCMIM1 KERGL01#TABLE01
```

You have reverted to the original DASD file or XES structure.

Activate List Structures when COMMUNICATION=XCF or CTCONLY

This procedure dynamically activates XES list structures when COMMUNICATION=XCF or CTCONLY. The CA MIM system names are GK03, GK13, and GK62.

Note: Be sure that your COMPATLEVEL is set to 12.0 or higher.

The following steps create a primary and a secondary list structure. The primary list structure is the size of the largest DISPLAY CFSIZE command. The secondary list structure is about 20 percent larger.

Follow these steps:

1. Set up your list structures:
   a. Confirm that all systems in the MIMplex are also within the same sysplex.
   b. Enter the DISPLAY CFSIZE command on all systems and determine the optimal size for the new XES list structures.

      ```
      @D CFSIZE
      MIM0067I Command DISPLAY 726
      MIM1154I GDIF CFSIZE Display:
      ```

      The recommended primary GDIF control file size is 10240 KB.
   c. Allocate the list structures using the sample JCL contained in the ALLOCSTR member of the MIM.CNTL library.
2. Activate the list structures.
   a. Allocate the new XES list structures to the MIM address space by using the ALLOCATE command:

      ```
      @ALLOC XESFILEID=00 STRNAME=KERGL01#TABLE00
      IXL014I IXLCONN REQUEST FOR STRUCTURE KERGL01#TABLE00 920
      WAS SUCCESSFUL.  JOBNAME: MIM12MIM ASID: 0088
      CONNECTOR NAME: MIM12MIMGK62 CFNAME: CFCCMIM2
      MIM0472I 2049 blocks formatted on KERGL01#TABLE00
      MIM0067I Command ALLOCATE 922
      MIM0153I ALLOCATION successful

      @ALLOC XESFILEID=01 STRNAME=KERGL01#TABLE01
      IXL014I IXLCONN REQUEST FOR STRUCTURE KERGL01#TABLE01 925
      WAS SUCCESSFUL.  JOBNAME: MIM12MIM ASID: 0088
      CONNECTOR NAME: MIM12MIMGK62 CFNAME: CFCCMIM1
      MIM0067I Command ALLOCATE 926
      MIM0153I ALLOCATION successful
      ```
   b. Enter the ALLOCATE commands on all systems.
   c. Confirm the successful allocation with the following DISPLAY FILES command on all systems:

      ```
      @D FILES
      ```

```
MIM0067I Command DISPLAY 937
MIM0102I Control File Display:
   File Unit Status  Volser   Data Set
    00       USABLE  CFCCMIM2 KERGL01#TABLE00
    01       USABLE  CFCCMIM1 KERGL01#TABLE01
```

d. Use the MIGRATE command for migrating to the XES list structure.

```
@MIG CF=00
MIM0067I Command MIGRATE 767
MIM0227I Migration initiated to control file 00
MIM0088I migration to file 00 IN PROGRESS
MIM0242I system GK62 VCF deactivation    COMPLETE
MIM0080I migration to file 00 COMPLETE
MIM0638I Global Copy Received
```

The migration is now complete.

3. Verify that CA MIM is using the proper file by issuing the following DISPLAY FILES command:

```
@D FILES
MIM0067I Command DISPLAY 780
MIM0102I Control File Display:
   File Unit Status  Volser   Data Set
    00       IN-USE  CFCCMIM2 KERGL01#TABLE00
    01       USABLE  CFCCMIM1 KERGL01#TABLE01
MIM0104I Checkpoint File Display:
   File Unit Status  Volser   Data Set
    00  2064 IN-USE  TSO23C   KERGL01.XE62.CKPT00
    01  2064 USABLE  TSO23C   KERGL01.XE62.CKPT01
```

The list structures are activated and file 00 is being used, which is correct.

Revert to the Original List Structures

You see a problem and want CA MIM to revert to use the XCF signaling(if COMMUNICATION=XCF) or the CTC communication (if COMMUNICATION=CTCONLY).

Follow these steps:

1. Issue the MIGRATE command:

```
@MIGRATE MASTER=GK03
MIM0067I Command MIGRATE 813
MIM0232I VCF migration SCHEDULED   - MASTER=GK03
MIM0377I VCF migration IN PROGRESS - MASTER=GK03
MIM0342I system GK62 VCF activation      COMPLETE - MASTER=GK03
MIM0241I VCF migration COMPLETE    - MASTER=GK03
MIM0637I Global Copy Sent to GK03     GK13
```

2. Confirm that GK03 is now the VCFMASTER by using the DISPLAY SYSTEMS command:

```
@D SYSTEMS
MIM0067I Command DISPLAY 825
MIM0108I SYSTEMS DISPLAY
   INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
    01    03   GK03     EXTERNAL MASTER      zOS  2013.172 11:39:35.68
    02    13   GK13     EXTERNAL ACTIVE      zOS  2013.172 11:39:35.61
    03    62   GK62     LOCAL    ACTIVE      zOS  2013.172 11:39:35.69
```

Activate XCF Communication on a Subset of the Systems

This procedure dynamically activates XCF communication on a subset of the MIMplex when COMMUNICATION=CTCDASD or CTCONLY. The MIMplex is four systems with system names of GK03, GK13, GK23, and GK62. GK03, GK13, and GK62 are in one sysplex. GK23 is in a different sysplex.

Note: Be sure that your COMPATLEVEL is set to 12.0 or higher.

Follow these steps:

1. Set up your XCF communication on a subset of systems:
   - Set MIMINIT VCFBUFFERSIZE to 32768 or larger.
   - Review the XCF configuration and options in the SYS1.PARMLIB(COUPLExx) member.
   - You do not need to define a special transport class to CA MIM. However, we recommend at lease one CLASSLEN of 32768 or larger to avoid possible performance problems. CLASSLEN is an XCF parameter and is coded in the SYS1.PARMLIB(COUPLExx) member.
   - If some of the LPARS are geographically remote, then IBM recommends that XCF use both CTC and list structures.

   Note: You do not need to create the Checkpoint files. However, if you are already using them CA MIM has no issues.
2. Confirm the current paths by entering the DISPLAY PATH command on all four systems.
   - On GK03:

```
@D PATH
MIM0067I Command DISPLAY 422
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  08:38:10 ON 2013.176
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 XCF  GK62     USABLE                  0        0                    0
 XCF  GK13     USABLE                  0        0                    0
 N/A  GK03     LOCAL                   0        0     0.000382   3,269
 184A GK62     USABLE  IDLE          3,268    3,268                   0
 180A GK62     USABLE  IDLE              0        0                   0
 0B4A GK62     USABLE  IDLE              0        0                   0
 186A GK23     USABLE  IDLE          3,268    3,268                   0
 182A GK23     USABLE  IDLE              0        0                   0
 0B6A GK23     USABLE  IDLE              0        0                   0
 183A GK13     USABLE  IDLE          3,254    3,254                   0
 154A GK13     USABLE  IDLE              0        0                   0
 0B3A GK13     USABLE  IDLE              0        0                   0
```

In the previous display, CA MIM automatically created the XCF paths to external systems that are in the same sysplex as the local system. No actions were required to accomplish this task. Also notice that CA MIM does not show the path currently IN-USE. The behavior of the DISPLAY PATH command is required to use the command on the client systems to determine the current path.

- On GK13:

```
@D PATH
MIM0067I Command DISPLAY 673
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  08:38:20 ON 2013.176
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 XCF  GK62     USABLE                   0        0                    0
 N/A  GK13     LOCAL                    0        0                    0
 187A GK62     USABLE  IDLE             0        0                    0
 0BFA GK62     USABLE  IDLE             0        0                    0
 0B7A GK62     USABLE  IDLE             0        0                    0
 185A GK23     USABLE  IDLE             0        0                    0
 181A GK23     USABLE  IDLE             0        0                    0
 0B5A GK23     USABLE  IDLE             0        0                    0
>183A GK03     IN-USE  BUSY         3,540    3,540     0.001925   3,540
                       RESERVE
 154A GK03     USABLE  IDLE             0        0                    0
 0B3A GK03     USABLE  IDLE             0        0                    0
 XCF  GK03     USABLE                   0        0                    0
```

Note the XCF paths to GK03 and GK62 in the previous display.

- On GK23:

```
@D PATH
MIM0067I Command DISPLAY 327
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  08:38:29 ON 2013.176
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK23     LOCAL                    0        0                    0
 183A GK62     USABLE  IDLE             0        0                    0
 154A GK62     USABLE  IDLE             0        0                    0
 0B3A GK62     USABLE  IDLE             0        0                    0
 185A GK13     USABLE  IDLE             0        0                    0
 181A GK13     USABLE  IDLE             0        0                    0
 0B5A GK13     USABLE  IDLE             0        0                    0
>186A GK03     IN-USE  IDLE         3,657    3,657     0.002053   3,657
 182A GK03     USABLE  IDLE             0        0                    0
 0B6A GK03     USABLE  IDLE             0        0                    0
```

The previous display shows no XCF paths to GK03, GK13, and GK62 because GK23 is in a different sysplex.

- On GK62:

```
@D PATH
MIM0067I Command DISPLAY 136
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  08:38:36 ON 2013.176
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK62     LOCAL                   0        0                    0
 183A GK23     USABLE  IDLE            0        0                    0
 154A GK23     USABLE  IDLE            0        0                    0
 0B3A GK23     USABLE  IDLE            0        0                    0
 187A GK13     USABLE  IDLE            0        0                    0
 0BFA GK13     USABLE  IDLE            0        0                    0
 0B7A GK13     USABLE  IDLE            0        0                    0
>184A GK03     IN-USE  IDLE        3,796    3,796     0.002015   3,796
 180A GK03     USABLE  IDLE            0        0                    0
 0B4A GK03     USABLE  IDLE            0        0                    0
 XCF  GK03     USABLE                  0        0                    0
 XCF  GK13     USABLE                  0        0                    0
```

So, all four systems are currently using CTCs to communicate, but systems GK03, GK13, and GK62 can now communicate through XCF. Use the SET MIM VCFPREFERENCE=CTC/XCF command for specifying whether CA MIM uses the CTC or XCF connection. The default is CTC when COMMUNICATION=CTCONLY or CTCDASD.

3. Activate your XCF communication

   - Issue the SET MIM VCFPREFERENCE=XCF command on GK13 and GK62 to switch from CTC to XCF. The command is not needed on the current VCFMASTER(GK03) because it detects the switch from the clients.
   - On GK13:

     ```
     @SET MIM VCFPREF=XCF
     MIM0067I Command SETOPTION 169
     MIM0052I SETOPTION MIM processing complete
     MIM0292I VCF switched to alternate VCF path XCF
     ```

   - On GK62:

     ```
     @SET MIM VCFPREF=XCF
     MIM0067I Command SETOPTION 169
     MIM0052I SETOPTION MIM processing complete
     MIM0292I VCF switched to alternate VCF path XCF
     ```

   The activation is now complete.

4. Update the commands member with:

   ```
   IFSYS GK03,GK13,GK62
   SET MIM VCFPREFERENCE=XCF
   ENDIF
   ```

   Note: GK23 was not included in the command as we want it to continue using CTC which is the default for COMMUNICATION=CTCDASD or CTCONLY.

5. Verify the current connectivity by entering the DISPLAY PATH command on all client systems.

   - On GK62:

```
@D PATH
MIM0067I Command DISPLAY 180
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  08:38:36 ON 2013.176
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK62     LOCAL                    0        0                    0
 183A GK23     USABLE  IDLE             0        0                    0
 154A GK23     USABLE  IDLE             0        0                    0
 0B3A GK23     USABLE  IDLE             0        0                    0
 187A GK13     USABLE  IDLE             0        0                    0
 0BFA GK13     USABLE  IDLE             0        0                    0
 0B7A GK13     USABLE  IDLE             0        0                    0
 184A GK03     USABLE  IDLE         4,769    4,769     0.001856   4,769
 180A GK03     USABLE  IDLE             0        0                    0
 0B4A GK03     USABLE  IDLE             0        0                    0
>XCF  GK03     IN-USE                  95       95     0.000408      95
 XCF  GK13     USABLE                   0        0                    0
```

     Notice how the XCF path is now being used between GK62 and GK03.

   - On GK13:

```
@D PATH
MIM0067I Command DISPLAY 776
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  08:38:20 ON 2013.176
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 XCF  GK62     USABLE                   0        0                    0
 N/A  GK13     LOCAL                    0        0                    0
 187A GK62     USABLE  IDLE             0        0                    0
 0BFA GK62     USABLE  IDLE             0        0                    0
 0B7A GK62     USABLE  IDLE             0        0                    0
 185A GK23     USABLE  IDLE             0        0                    0
 181A GK23     USABLE  IDLE             0        0                    0
 0B5A GK23     USABLE  IDLE             0        0                    0
 183A GK03     USABLE  IDLE         5,266    5,266     0.001632   5,266
 154A GK03     USABLE  IDLE             0        0                    0
 0B3A GK03     USABLE  IDLE             0        0                    0
>XCF  GK03     IN-USE                 105      105     0.001399     105
```

     Notice how the XCF path is now being used between GK13 and GK03.

   - On GK23:

```
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  08:38:29 ON 2013.176
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK23     LOCAL                    0        0                    0
```

```
 183A GK62     USABLE  IDLE            0       0                  0
 154A GK62     USABLE  IDLE            0       0                  0
 0B3A GK62     USABLE  IDLE            0       0                  0
 185A GK13     USABLE  IDLE            0       0                  0
 181A GK13     USABLE  IDLE            0       0                  0
 0B5A GK13     USABLE  IDLE            0       0                  0
>186A GK03     IN-USE  IDLE        3,657   3,657     0.002053   3,657
 182A GK03     USABLE  IDLE            0       0                  0
 0B6A GK03     USABLE  IDLE            0       0                  0
```

GK23 is still using a CTC connection to GK03 because there are no XCF paths between GK23 and GK03.

Revert to CTC Communication

You decide to revert CA MIM back to using the CTC communication.

Follow these steps:

1. Use the SET MIM VCFPREFERENCE=CTC command.

   - On GK13:

```
@SET MIM VCFPREFERENCE=CTC
MIM0067I Command SETOPTION 035
MIM0052I SETOPTION MIM processing complete
MIM0292I VCF switched to alternate VCF path 183A
@D PATH
MIM0067I Command DISPLAY 039
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  08:51:19 ON 2013.176
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK13     LOCAL                   0        0                    0
 187A GK62     USABLE  IDLE            0        0                    0
 0BFA GK62     USABLE  IDLE            0        0                    0
 0B7A GK62     USABLE  IDLE            0        0                    0
 185A GK23     USABLE  IDLE            0        0                    0
 181A GK23     USABLE  IDLE            0        0                    0
 0B5A GK23     USABLE  IDLE            0        0                    0
>183A GK03     IN-USE  IDLE           29       29     0.000692      29
 154A GK03     USABLE  IDLE            0        0                    0
 0B3A GK03     USABLE  IDLE            0        0                    0
 XCF  GK03     USABLE              46,770   46,770     0.002763  46,770
 XCF  GK62     USABLE                  0        0                    0
```

     In the previous display, you can see that GK13 is now using CTC address 183A to GK03.

   - On GK62:

```
@SET MIM VCFPREF=CTC
MIM0067I Command SETOPTION 635
0052I SETOPTION MIM processing complete
MIM0292I VCF switched to alternate VCF path 184A
@D PATH
MIM0067I Command DISPLAY 697
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  08:38:36 ON 2013.176
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK62     LOCAL                   0        0                    0
 183A GK23     USABLE  IDLE            0        0                    0
 154A GK23     USABLE  IDLE            0        0                    0
 0B3A GK23     USABLE  IDLE            0        0                    0
 187A GK13     USABLE  IDLE            0        0                    0
 0BFA GK13     USABLE  IDLE            0        0                    0
```

```
 0B7A GK13     USABLE  IDLE            0        0                    0
>184A GK03     IN-USE  IDLE        4,794    4,794     0.001852   4,794
 180A GK03     USABLE  IDLE            0        0                    0
 0B4A GK03     USABLE  IDLE            0        0                    0
 XCF  GK03     USABLE              60,408   60,408     0.003041  60,408
 XCF  GK13     USABLE                  0        0                    0
```

In the previous display, you can see that GK62 is now using CTC address 184A to GK03.

No commands are required for GK23 because it was already using CTCs. The command is not needed on the current VCFMASTER(GK03) because it detects the switch from the clients.

Add a System to the MIMplex when COMMUNICATION=CTCDASD

This procedure dynamically adds a system to an existing MIMplex when COMMUNICATION=CTCDASD or CTCONLY.

Note: Be sure that your COMPATLEVEL is set to 12.0 or higher.

Follow these steps:

1. Set up your new system.

   a. Review the current configuration.

      In this example, we have a two system MIMplex with systems GK03 and GK13. We are going to add system GK62.

      The current INIT member contains the following statements:

```
DEFSYS (GK03,03,XE03,INIT=FREED),
            (GK13,13,XE13,INIT=FREED)

GLOBALVALUE VCFMASTER=(GK03,GK13),
            MOSTP=YES,
            NOMASTER=WAIT,
            ANYELIG=YES
CTCPATH FROMSYS=GK03 TOSYS=GK13 ADDR=(183A,154A,B3A)
CTCPATH FROMSYS=GK13 TOSYS=GK03 ADDR=(183A,154A,B3A)
```

   b. Confirm the current configuration on GK13 using the DISPLAY command:

```
@D SYS
MIM0067I Command DISPLAY 355
MIM0108I SYSTEMS DISPLAY
   INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
    01    03   GK03     EXTERNAL MASTER      zOS  2013.175 14:40:17.11
    02    13   GK13     LOCAL    ACTIVE      zOS  2013.175 14:40:17.20

@D PATH
MIM0067I Command DISPLAY 359
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  14:39:35 ON 2013.175
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK13     LOCAL                    0        0                    0
>183A GK03     IN-USE  IDLE           305      305     0.001085     305
 154A GK03     USABLE  IDLE             0        0                    0
 0B3A GK03     USABLE  IDLE             0        0                    0

@D GLOBALV
MIM0067I Command DISPLAY 372
MIM0373I MIM GLOBALVALUE display:
   ANYELIGIBLE=YES       MOSTPREFERRED=YES     NOMASTER=WAIT
   VCFMASTER=( GK03, GK13 )
   ELIGIBLE MASTER LIST: GK03, GK13
```

c. Confirm the current configuration on GK03 using the DISPLAY command:

```
@D PATH
MIM0067I Command DISPLAY 237
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  14:39:23 ON 2013.175
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK03     LOCAL                    0        0     0.000024     624
 183A GK13     USABLE  IDLE           621      621                    0
 154A GK13     USABLE  IDLE             0        0                    0
 0B3A GK13     USABLE  IDLE             0        0                    0
```

2. Activate the new system GK62 using the DEFSYS command on one system only:

```
@DEFSYS (GK62,62,XE62)
MIM0067I Command DEFSYS 319
MIM0099I DEFSYS command is processing
MIM0408I System Add Successful - GK62
```

3. Confirm that GK62 has been added successfully:

```
@D SYS
MIM0067I Command DISPLAY 327
MIM0108I SYSTEMS DISPLAY
   INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
    01    03   GK03     LOCAL    MASTER      zOS  2013.175 14:44:03.55
    02    13   GK13     EXTERNAL ACTIVE      zOS  2013.175 14:44:03.55
    03    62   GK62     EXTERNAL FREED             <- new system
```

4. (Optional) Update the VCFMASTER parameter on the GLOBALVALUE statement. This list shows the systems from busiest to least busy (in terms of the global ENQ activity).

   In this example, GK62 has the least amount of activity, so we want to put this system last in the list:

```
@GLOBALVALUE VCFMASTER(GK03,GK13,GK62)
MIM0067I Command GLOBALVALUE 360
MIM0386W System GK62 is ineligible to become VCF MASTER
MIM0193I GLOBALVALUE command processing initiated
MIM0363I GLOBALVALUE command changes pending on external systems
MIM0362I GLOBALVALUE command processing complete
```

5. Confirm the change:

```
@D GLOBALV
MIM0067I Command DISPLAY 367
MIM0373I MIM GLOBALVALUE display:
   ANYELIGIBLE=YES       MOSTPREFERRED=YES     NOMASTER=WAIT
   VCFMASTER=( GK03, GK13, *GK62 )
   ELIGIBLE MASTER LIST: GK03, GK13
```

   GK62 becomes eligible when it is started.

Test the connectivity of the devices using the PROCCTC utility before you use the CTCPATH command.

6. Create the CTCPATH definitions on GK03 and GK13.

   - On GK03:

```
@CTCPATH FROMSYS=GK03 TOSYS=GK62 ADDR=(184A,180A,B4A)
MIM0067I Command CTCPATH 500
MIM0703I Path added for device 184A
MIM0703I Path added for device 180A
MIM0703I Path added for device 0B4A
```

     Confirm the new paths:

```
@D PATH
MIM0067I Command DISPLAY 520
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  14:39:23 ON 2013.175
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 0B4A GK62     USABLE  BUSY            0        0                   0
                       WRITHDS
 180A GK62     USABLE  BUSY            0        0                   0
                       WRITHDS
 184A GK62     USABLE  BUSY            0        0                   0
                       WRITHDS
 N/A  GK03     LOCAL                   0        0     0.000055   4,980
 183A GK13     USABLE  IDLE        4,969    4,969                   0
 154A GK13     USABLE  IDLE            0        0                   0
 0B3A GK13     USABLE  IDLE            0        0                   0
```

   - On GK13:

```
@CTCPATH FROMSYS=GK13 TOSYS=GK62 ADDR=(187A,BFA,B7A)
MIM0067I Command CTCPATH 685
MIM0703I Path added for device 187A
MIM0703I Path added for device 0BFA
MIM0703I Path added for device 0B7A
```

     Confirm the new paths:

```
@D PATH
MIM0067I Command DISPLAY 695
MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  14:39:35 ON 2013.175
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 0B7A GK62     USABLE  BUSY            0        0                   0
                       WRITHDS
 0BFA GK62     USABLE  BUSY            0        0                   0
                       WRITHDS
 187A GK62     USABLE  BUSY            0        0                   0
                       WRITHDS
 N/A  GK13     LOCAL                   0        0                   0
>183A GK03     IN-USE  IDLE        4,859    4,859     0.001026   4,859
```

```
     154A GK03     USABLE  IDLE            0       0                  0
     0B3A GK03     USABLE  IDLE            0       0                  0
```

7. Update the INIT member with the contents of the previous commands and add the CTCPATHs from GK62:

```
DEFSYS (GK03,03,XE03,INIT=FREED),
       (GK13,13,XE13,INIT=FREED),
       (GK62,62,XE62,INIT=FREED)

GLOBALVALUE VCFMASTER=(GK03,GK13,GK62),
            MOSTP=YES,
            NOMASTER=WAIT,

CTCPATH FROMSYS=GK03 TOSYS=GK13 ADDR=(183A,154A,B3A)
CTCPATH FROMSYS=GK03 TOSYS=GK62 ADDR=(184A,180A,B4A)

CTCPATH FROMSYS=GK13 TOSYS=GK03 ADDR=(183A,154A,B3A)
CTCPATH FROMSYS=GK13 TOSYS=GK62 ADDR=(187A,BFA,B7A)  <-new

CTCPATH FROMSYS=GK62 TOSYS=GK03 ADDR=(184A,180A,B4A)
CTCPATH FROMSYS=GK62 TOSYS=GK13 ADDR=(187A,BFA,B7A)  <-new
```

Note: If COMM=CTCDASD, ensure that the control files or list structures are allocated or cataloged on the new system.

8. Start MIM on GK62:

```
S MIM,FORMAT=CHKPT
```

The new system joins the existing MIMplex.

Note: COMMUNICATION=CTCDASD does not require a checkpoint file. However, if you have one allocated due to using ECMF REQUEUE, then format it.

9. Verify that the new system joined the MIMplex by entering some verification commands on the new system/GK62:

```
@DCSF
MIM0023I system GK62 in file VCF synchronization COMPLETE
MIM0067I Command DISPLAY 676
MIM0108I SYSTEMS DISPLAY
   INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
    01    03   GK03     EXTERNAL MASTER      zOS  2013.175 14:53:19.61
    02    13   GK13     EXTERNAL ACTIVE      zOS  2013.175 14:53:19.66
    03    62   GK62     LOCAL    ACTIVE      zOS  2013.175 14:53:19.67

GK62 is now ACTIVE.

MIM0373I MIM GLOBALVALUE display:
   ANYELIGIBLE=YES       MOSTPREFERRED=YES     NOMASTER=WAIT
   VCFMASTER=( GK03, GK13, GK62 )
   ELIGIBLE MASTER LIST: GK03, GK13, GK62
```

```
GK62 is now an eligible master.

MIM0176 MIM PATH DISPLAY:
LAST RESTART AT  14:52:58 ON 2013.175
 DEV  SYSTEM   STATE   IO-STAT     READS   WRITES  ERR TIME/RSV  CYCLES
 N/A  GK62     LOCAL                    0        0                    0
 187A GK13     USABLE  IDLE             0        0                    0
 0BFA GK13     USABLE  IDLE             0        0                    0
 0B7A GK13     USABLE  IDLE             0        0                    0
>184A GK03     IN-USE  IDLE             3        3     0.001479       3
 180A GK03     USABLE  IDLE             0        0                    0
 0B4A GK03     USABLE  IDLE             0        0                    0
```

All 6 CTC addresses have been allocated and 184A is the path currently IN-USE to the VCFMASTER(GK03).

Revert to the Original MIMplex

When the new system does not join the MIMplex:

1. Shut down CA MIM on that system.
2. Enter the FREE command from one of the active systems.

Once the system has been added with the DEFSYS command, there is no way to remove it. You may need to restore the INIT member to the original state, and perform a global shutdown and control file/checkpoint file format.

Add a System when COMMUNICATION=DASDONLY

You want a system dynamically added to the established MIMplex using COMMUNICATION=DASDONLY. The CA MIM name and alias are honored as coded on the DEFSYS statement.

Follow these steps:

1. Ensure that no systems have been REMOVED from the MIMPLEX by reviewing the DISPLAY SYSTEMS command output:

```
MIM0108I SYSTEMS DISPLAY
   INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
    01    03   GK03     EXTERNAL ACTIVE      zOS  2012.013 12:25:45.38
    02    13   GK13     LOCAL    ACTIVE      zOS  2012.013 12:25:45.91
    03    23   GK23     EXTERNAL FREED
```

   Verify that there are no gaps between the index numbers. If there are no gaps, then it is safe to proceed with the next step. Otherwise, update the DEFSYS statement, shut down globally, then restart globally with a control file format.

2. Update the DEFSYS statement in the CA MIMINIT member:

```
DEFSYS (GK03,03,XE03,INIT=FREED),
       (GK13,13,XE13,INIT=FREED),
       (GK23,23,XE23,INIT=FREED),
       (GK62,62,XE62,INIT=FREED)      /* new system
```

3. Start MIM on the new LPAR (GK62 in this example) with FORMAT=CHKPT or FORMAT=NONE if no checkpoints exist.

   The new system joins the established MIMplex and honors the CA MIM name and alias as coded on the DEFSYS statement.

4. Confirm the new system joined with the DISPLAY SYSTEMS command:

```
MIM0108I SYSTEMS DISPLAY
   INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
    01    03   GK03     EXTERNAL ACTIVE      zOS  2012.013 12:26:26.39
    02    13   GK13     EXTERNAL ACTIVE      zOS  2012.013 12:26:26.88
    03    23   GK23     EXTERNAL FREED
    04    62   GK62     LOCAL    ACTIVE      zOS  2012.013 12:26:27.36
```

The new system has been added to the MIMplex.

Add a System when COMMUNICATION=XCF

You want a system dynamically added to the established MIMplex using COMMUNICATION=XCF. The CA MIM name and alias are honored as coded on the DEFSYS statement.

Follow these steps:

1. Issue the DISPLAY INIT command to ensure that the MIMPLEX is running with COMMUNICATION=XCF and COMPATLEVEL= 11.9 or greater.
2. Issue the DISPLAY SYSTEMS command to verify that you have an active MIMPLEX. The new system cannot be the first active system in the MIMPLEX.

```
MIM0108I SYSTEMS DISPLAY
  INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
   01    03   MIMMC003   LOCAL  MASTER       zOS  2012.026 10:52:21.82
   02    13   MIMMC013   EXTERNAL FREED
```

3. Issue the DEFSYS command with your preferred parameters.

```
DEFSYS (MIMMC062,62,XE62)
```

   Verify that the following message appears:

```
MIM0408I – System Add Successful – MIMMC062
```

4. Issue the DISPLAY SYSTEMS command to ensure that the new system was added correctly.

```
INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
  01    03   MIMMC003 LOCAL    MASTER      zOS  2012.026 10:57:06.16
  02    13   MIMMC013 EXTERNAL FREED
  03    62   MIMMC062 EXTERNAL FREED
```

   For a new system to become a VCFMASTER system, update the GLOBALVALUE parameter.

5. Issue a DISPLAY GLOBALVALUE command. Note the list of eligible master systems.

```
MIM0373I MIM GLOBALVALUE display:
ANYELIGIBLE=YES       MOSTPREFERRED=YES     NOMASTER=WAIT
VCFMASTER=( MIMMC003, MIMMC013 )
ELIGIBLE MASTER LIST: MIMMC003, MIMMC013
```

6. Issue the following command on one system only:

```
GLOBALVALUE VCFMASTER={LIST}
```

   Note: Copy the VCFMASTER list from the prior command and add the desired system.

```
GLOBALVALUE VCFMASTER=( MIMMC003, MIMMC013 , MIMMC062)
```

7. Verify Output:

```
MIM0363I GLOBALVALUE command changes pending on external systems
GLOBALVALUE command processing complete
```

8. Issue a DISPLAY GLOBALVALUE command again to verify that the desired system was added to the VCFMASTER list.

```
MIM0373I MIM GLOBALVALUE display:
ANYELIGIBLE=YES       MOSTPREFERRED=YES     NOMASTER=WAIT
VCFMASTER=( MIMMC003, MIMMC013, MIMMC062)
ELIGIBLE MASTER LIST: MIMMC003, MIMMC013, MIMMC062
```

9. Update the GLOBALVALUE statement in the MIMINIT member to match the exact text of the operator command.

```
GLOBALVALUE VCFMASTER=(MIMMC003,MIMMC013, MIMMC062),
            ANYELIGIBLE=YES,
            NOMASTER=WAIT,
            MOSTPREFERRED=YES
```

10. Update the DEFSYS statement in the MIMINIT member. Be sure to code the new system last on the system list.

```
DEFSYS (MIMMC003,03,XE03,INITIAL=FREED),
       (MIMMC013,13,XE13,INITIAL=FREED),
       (MIMMC062,62,XE62)
```

11. Allocate checkpoint files for the new system. Defined them using the CHKPTDSN statement in the MIMINIT member.

```
MIMINIT CHKPTDSN=CHIMA14.MIMMC0&SYSCLONE..MIMCKP
```

12. Start the new system with FORMAT=CHKPT.

13. Confirm that the new system has been added by entering a DISPLAY SYSTEMS command.

```
MIM0108I SYSTEMS DISPLAY
  INDEX ALIAS SYSTEM   RELATION STATUS      OPSYS     LAST ACCESS
   01    03   MIMMC003 EXTERNAL MASTER      zOS  2012.026 11:05:55.14
   02    13   MIMMC013 EXTERNAL FREED
   03    62   MIMMC062 LOCAL    ACTIVE      zOS  2012.026 11:05:55.2
```

The new system has been added.

Format Start

On a format start, CA MIM will erase any existing information within the control or checkpoint file and revert to using defaults and settings provided by your initialization parameter members. Depending on your configuration, you may need to format the control file, the checkpoint file, or both.

Format only the control file if the following are true:

- You have specified DASDONLY or CTCDASD as the initial communications method
- You are not using the ECMF REQUEUE feature

To perform a format start when checkpoint files are not present:

1. Stop CA MIM on all systems in the MIMplex by issuing the following command:

   ```
   @SHUTDOWN,GLOBAL
   ```

2. Start CA MIM on the first system by issuing the following command:

   ```
   S MIMGR,FORMAT=CF
   ```

3. Start CA MIM on all other systems normally by issuing the following command on each system:

   ```
   S MIMGR
   ```

Format both the control file and the checkpoint file if the following are true:

- You have specified DASDONLY or CTCDASD as the initial communications method and you are using the ECMF REQUEUE feature. The ECMF REQUEUE feature requires checkpoint files.
- You have specified XCF or CTCONLY as the initial communications method.

To perform a format start when checkpoint files are required:

1. Stop CA MIM on all systems in the MIMplex by issuing the following command:

   ```
   @SHUTDOWN,GLOBAL
   ```

2. Start CA MIM on the first system by issuing the following command:

   ```
   S MIMGR,FORMAT=BOTH
   ```

3. Start CA MIM on all other systems normally by issuing the following command on each system:

   ```
   S MIMGR,FORMAT=CKPT
   ```

Note: When starting with XCF or CTCONLY as the initial communications method, the following message is displayed when the VCF is formatted:

MIM0359 VCF FORMAT COMPLETE - MIM CAN BE STARTED ON OTHER SYSTEMS

How You Define CA MIM in the Subsystem Name Table

This section explains how the CA MIM subsystem name value should be defined in the *IEFSSNxx* member of SYS1.PARMLIB and on the MIMINIT SUBNAME parameter of the MIMINIT member of the CA MIM parmlib.

This information is of value in environments where the order in which subsystem interface events are processed is important, specifically with tape allocations. For example, in an environment in which CA MIA coexists with the robotic tape drive software of other vendors, you would want CA MIA to process tape allocation requests after the products of other vendors do.

If you are concerned about the order in which subsystem interface events are processed, then you should define a subsystem name for CA MIM in the IEFSSN*xx* member of the SYS1.PARMLIB data set. Using the keyword format, place the CA MIM subsystem name definition in the appropriate order relative to other subsystem definitions. Proper placement of the CA MIM definition ensures that subsystem interface events are processed in the proper order by each subsystem.

The CA MIM subsystem naming convention for the IEFSSN*xx* member of SYS1.PARMLIB requires the first three characters to be “MIM” and the fourth character be alphanumeric or special. The fourth character cannot be *, D, or ?.

The following is an example of a valid IEFSSN*xx* definition for CA MIM:

```
SUBSYS SUBNAME(MIMA)     /* CA-Multi-image Allocation  */
```

After you update the IEFSSN*xx* member of SYS1.PARMLIB, you must then update the MIMINIT SUBNAME= parameter in the MIMINIT member of the CA MIM parmlib like this:

```
MIMINIT SUBNAME=MIMA
```

When the CA MIM address space is started, it processes this statement, then searches for and locates the matching subsystem definition, MIMA, that was established when the IEFSSN*xx* definitions were processed.

Note: If you do not define the subsystem name properly in both parmlibs, matching definitions will not be found, and subsystem events will not be processed based on your IEFSSN*xx* definitions.

If you do not have concerns about the order in which subsystem interface events are processed, you do not need to provide definitions for CA MIM in the IEFSSN*xx* member of SYS1.PARMLIB. Regardless of whether you place a CA MIM definition in IEFSSN*xx*, CA MIM will still dynamically create a subsystem definition for itself during product initialization. When CA MIM is started after an IPL, the MIMINIT SUBNAME parameter is processed and an attempt is made to locate a like-named subsystem. If you do not have a CA MIM definition in IEFSSN*xx*, then no matching name will be found, and CA MIM will dynamically define a subsystem using the name provided on the MIMINIT SUBNAME parameter.

When a CA MIM subsystem value has not been predefined in IEFSSN*xx*, the first three characters of the MIMINIT SUBNAME value must be “MIM” and the fourth character can be alphanumeric or special character, except for *, ?, or D.

The following is an example of this naming convention:

```
MIMINIT SUBNAME=MIM&
```

Note: You can use most EBCDIC characters when calling the IBM subsystem services to define a subsystem. However, you can use only alphanumeric and national characters when defining subsystems in the IEFSSN*xx* member of SYS1.PARMLIB.

If you do not provide a subsystem name value on the MIMINIT SUBNAME parameter, CA MIM uses a default name when dynamically creating a subsystem definition during product startup.

Note: For more information, see the *CA MIM Statement and Command Reference Guide* and the IBM guides *Initialization and Tuning Reference* and *Using the Subsystem Interface*.

How You Run Components in Separate Address Spaces

Many customers activate all CA MIM components in a single address space. That is a single CA MIM startup procedure references parmlib members that activate all CA MIM components. Other customers find it advantageous to activate CA MIA and CA MIC in one address space, and CA MII in a second address space. Other customers prefer running three address spaces, each having one CA MIM component activated.

Note: We recommend running CA MII in a separate address space from CA MIA and CA MIC.

Consider the following issues when running multiple address spaces:

- Each address space must be activated using a uniquely named startup procedure. Each procedure must reference unique MIMINIT, MIMCMNDS, and MIMSYNCH parmlib members, and unique control files.
- Each address space needs a unique command character as defined by the CMDPREFIX option of the SETOPTION MIM or MIMINIT commands.
- Severe problems affecting one CA MIM address space do not affect the other CA MIM address spaces running on the same system.
- If you run CA MIM with CA MIM for z/VM, we strongly recommend running CA MII separately from CA MIA and CA MIC.
- You can stop each address space individually using the assigned command prefix character. For example, you can stop CA MIA without affecting CA MII.
- Consider setting MIMINIT MSGPREFIX=CMDPREFIX when running CA MIM components in separate address spaces. This parameter lets each CA MIM address space prefix its messages with its unique command prefix character. These unique prefix characters help system operators and programmers easily identify the CA MIM address space from which a particular message originated.

Important! When running multiple address spaces, you can use the z/OS MODIFY (F) command, F MIM, to direct commands to all CA MIM address spaces. For display commands like F MIM, D FACILITIES, or F MIM, D SYSTEMS, all CA MIM address spaces respond.

However, exercise caution when using F MIM to issue commands. For example, the command F MIM,SHUTDOWN causes all CA MIM address spaces to terminate. Instead of using F MIM, use the command prefix character to direct commands to the CA MIM address space you want to control. (If you named your CA MII started task MIMDASD, specify F MIMDASD.)

The following are component-specific sample startup procedures in the CAI.CBTDPROC data set that reference component-specific sample parmlib members in the CAI.CBTDPARM data set:

- **Procedure PROCMIA** references parmlib members MIAINIT, MIACMNDS, MIASYNCH
- **Procedure PROCMIC** references parmlib members MICINIT, MICCMNDS, MICSYNCH
- **Procedure PROCMII** references parmlib members MIIINIT, MIICMNDS, MIISYNCH

Product Termination Considerations

You can use either the z/OS STOP command or the CA MIM SHUTDOWN command to stop CA MIM. Both let you stop it on all systems in the complex (global) or on a single system (local).

Global Shutdown Processing

CA MIM performs several actions to ensure the integrity of your resources at termination. When you stop CA MIM on all systems in your complex, components and facilities take the following actions:

CA MIA Activities During Global Shutdown

When CA MIA is running, we strongly recommend that you stop CA MIM on all systems, rather than on a subset of systems.

WARNING! Stopping CA MIM with a CA MIM SHUTDOWN LOCAL/FREE/RESERVE/DUMP (or z/OS STOP) command when GTAF is running may create data integrity exposures.

You can use either the z/OS STOP command or the CA MIM SHUTDOWN command to stop CA MIA. Both allow you to stop the product on the local system only or on all systems in the complex.

When CA MIA is not running, the operator may share tape devices manually by using z/OS VARY OFFLINE and VARY ONLINE commands, ensuring that each device is online to only one system at a time.

GTAF takes offline all unallocated devices on each system. This prevents z/OS from selecting these devices (because they could be in use on another system).

GTAF marks as PENDING OFFLINE all allocated devices on the system to which they are allocated. This causes the local z/OS operating system to take the devices offline as soon as they are deallocated. GTAF takes the devices offline on all other systems.

TPCF restores control to z/OS for any job that is in the allocation recovery process. TPCF replies with an invalid response to any outstanding z/OS IEF238D message, which restores control to z/OS and causes z/OS to prompt you for information.

CA recognizes that it may not be possible to quiesce all tape activity on a system. It is possible to safely stop CA MIM on one local system without quiescing all tape activity. However, it is not practical to document every situation that may occur under such circumstances.

We strongly recommend that you use caution if attempting to shut down a subset of the systems in the MIMplex. If you are not confident that you can safely avoid data integrity exposures, contact CA Technical Support for assistance.

More information:

CA MIA Activities During Local Shutdown (see page 106)

How You Automatically Free a System From the MIMplex

The auto FREE feature provides a means to automate the FREE command operation for a CA MIM member on a failed system in a sysplex. The auto FREE feature functions independently of the MIMplex control file communication methodology.

The auto FREE feature works by reacting to a notification event provided by the XCF/XES operating system component, which describes that a system has been removed from the active sysplex.

Note: For information on sysplex member removal, see the IBM MVS guide *Setting Up a Sysplex*.

The MIMplex can be a subset of a sysplex, or it can be comprised of more than one sysplex. As long as one member of a sysplex survives to provide a system down notification event, the auto FREE feature automatically FREEs the removed MIMplex member. A MIMplex member must be a member of a sysplex to be automatically freed with auto FREE. The system down notification event is provided by the XCF/XES operating system component, in a sysplex environment.

In addition to the MIMplex configuration relationship to the sysplex configuration, consider the following regarding the use of the auto FREE feature:

- The auto FREE feature is controlled by the SETOPTION MIM DOWNSYS command and can be enabled or disabled on an individual system basis in the MIMplex. DOWNSYS=AUTOFREE needs to be specified on the surviving sysplex member to react to the incoming system down event.
- To relate CA MIM system names with z/OS system names, the z/OS system name, rather than the z/OS SMF ID, must be specified on the DEFSYS statement. The z/OS system name is defined on the SYSNAME keyword of the IEASYSnn member of the z/OS SYS1.PARMLIB data set. The z/OS DISPLAY XCF,SYSPLEX,ALL command can be used to list all of the z/OS system names in a given sysplex. Alternately, the z/OS DISPLAY SYMBOLS command shows the local z/OS system name for the &SYSNAME variable name.

CA MIC Activities During Global Shutdown

- GCMF terminates any incomplete multiple-line messages.
- GCMF deletes all highlighted messages from other systems.
- GCMF deallocates or deactivates subsystems or extended consoles used.
- ICMF notifies CA-L-Serv that it is terminating.

CA MII Activities During Global Shutdown

- GDIF stops propagating new ENQ and RESERVE requests.
- GDIF stops eliminating hardware reserves for any new RESERVE requests.
- GDIF does not restore any of the hardware reserves that it has converted. However, GDIF performs quiesce processing for any outstanding RESERVE requests that it already has converted and any requests included in a global conflict.
- The ECMF REQUEUE feature handles requeued jobs and checkpoint information you have specified through the REQCKPT parameter on a SETOPTION command.

Global Shutdown Procedure

When GDIF or GTAF is running, we strongly recommend that you stop CA MIM on all systems rather than on a subset of the systems in your complex. To do this, issue a z/OS STOP or the CA MIM SHUTDOWN command. To use the z/OS STOP command, specify the name of the CA MIM started task as a parameter. For example, if MIMGR is the name of the started task that you want to stop, then issue this shortened form of the z/OS STOP command:

```
P MIMGR
```

CA MIM then issues message MIM0060, which asks what type of processing you want to perform at termination. Reply GLOBAL to stop on all systems and to have CA MIM quiesce on all systems at termination.

You can also use the SHUTDOWN GLOBAL command to stop CA MIM on all systems. For example, issue this SHUTDOWN command from any system:

```
SHUTDOWN GLOBAL
```

Note: GLOBAL is the initial default value of the command. You can change the default value of the command to DUMP, FREE, LOCAL, or RESERVE with the SETOPTION SHUTDOWN command.

Note: For a description of the SHUTDOWN command, see the *Statement and Command Reference Guide*.

When GLOBAL is selected, CA MIM does not stop with an abend code or request an abend dump.

Although GDIF does not propagate any new ENQ or RESERVE requests or eliminate new hardware reserves, GDIF does not terminate while a propagated or converted request is outstanding. GDIF does terminate only after tasks have issued DEQ requests to release these resources. This ensures resource integrity.

GTAF takes offline any device that is not allocated on each local system. This ensures integrity of tape resources. After GTAF terminates, each managed tape device is left online to, at most, one system in the complex.

Local Shutdown Processing

To stop CA MIM on the local system only, you can issue a z/OS STOP command for the CA MIM started task, and then reply LOCAL or DUMP to the MIM0060 message you receive. Alternatively, you can stop it locally by issuing a SHUTDOWN LOCAL, SHUTDOWN DUMP, or SHUTDOWN FREE command.

Your reply to the MIM0060 message or the parameter you use on the SHUTDOWN command depends on whether you want to receive an abend dump. When you specify LOCAL, CA MIM terminates but does not request an abend dump. When you specify DUMP, CA MIM terminates and requests an abend dump.

For example, to stop CA MIM on the local system and receive an abend dump, issue the following command from the local system:

```
SHUTDOWN DUMP
```

Other systems are not able to use resources and devices held by the system that you are stopping. This remains in effect until you restart that system or free resources and devices through the FREE command. This can also be done automatically by issuing the SHUTDOWN FREE command.

When CA MIM terminates after a SHUTDOWN LOCAL or SHUTDOWN FREE command, abend code U1222 appears if SETOPTION LOCALSTOP=ABEND.

CA MII Activities During Local Shutdown

WARNING! Stopping CA MIM with the CA MIM SHUTDOWN LOCAL/FREE/RESERVE/DUMP (or z/OS STOP) command when GDIF is running may create data integrity exposures if the system on which CA MIM is stopped will generate new RESERVE or ENQ requests.

We strongly recommend that you stop CA MIM on all systems, rather than a subset of systems in your complex, when GDIF is running. When CA MIM is stopped on one system, any new RESERVE and ENQ requests on that system will not be serialized with the rest of the systems in the complex where CA MIM is still running. No new RESERVE requests are converted. No new ENQ requests are propagated. Therefore, there may be data integrity exposures. It is safe to stop CA MIM on only one system if that system will not be generating any new RESERVE or ENQ requests.

Resources held by the local system at the time CA MIM is stopped locally are not available for use by systems in the rest of the complex where CA MIM is still running until CA MIM is restarted on the local system, or the local system is freed through the CA MIM FREE command. A system can be automatically freed at time of shutdown by issuing the CA MIM SHUTDOWN FREE command.

CA MIA Activities During Local Shutdown

WARNING! When GTAF is running, we recommend that you stop CA MIM on all systems, rather than on a subset of systems in your complex. When CA MIM is stopped on one system, any new tape device activity on that system (such as allocations, z/OS VARY commands, or device name replies to IEF238D WTORs) will not be serialized with the rest of the systems in the complex where CA MIM is still running, creating data integrity exposures.

Tape resources held by the local system at the time CA MIM is stopped locally are not available for use by systems in the rest of the complex where CA MIM is still running until CA MIM is restarted on the local system, or the local system is freed through the CA MIM FREE command. A system can be automatically freed at time of shutdown by issuing the CA MIM SHUTDOWN FREE command.

Local Shutdown Procedure

This section describes a step-by-step local shutdown procedure. To avoid data integrity exposures, we recommend the following procedure to safely stop CA MIA on one local system in the complex:

1. Quiesce tape activity on the local system.

   This process may include steps such as draining tape class initiators, quiescing online systems, and so on.

2. Issue the following command on the local system:

   ```
   SHUTDOWN LOCAL
   ```

3. Issue the following command for the local system on an external system:

   ```
   FREE
   ```

   Note: The SHUTDOWN FREE command would replace Steps 2 and 3.

4. Determine what tape devices are to be used by the local system when tape processing on that system is restarted.

5. Issue the following command on an external system for each device identified in Step 4:

   ```
   VARY NOTAVAILABLE GLOBAL
   ```

6. Wait for all VARY NOTAVAILABLE GLOBAL commands issued in Step 5 to fully complete.

   This includes waiting for any necessary deallocations to occur and each device to be taken offline on all systems.

7. Issue the following z/OS command on the local system for each device identified in Step 4:

   ```
   VARY ONLINE
   ```

8. Restart tape processing on the local system.

9. If any further devices are needed by the local system, repeat Steps 4 through 7 for each device.

WARNING! If you issue a z/OS VARY command for a device on the local system (Step 7) without previously issuing a VARY NOTAVAILABLE GLOBAL command for the device on an external system (Steps 5 and 6), it may create data integrity exposures. The same is true for replying with a device name to an IEF238D WTOR.

Local System Shutdown With or Without Abend Codes

You can adjust the result of certain SHUTDOWN commands so that CA MIM stops with or without abend codes. The SHUTDOWN FREE and SHUTDOWN LOCAL commands cause a U1222 abend, and the SHUTDOWN RESERVE command causes a U1223 abend. These abends are issued only when SET MIM LOCALSTOP=ABEND. You can prevent this by changing this parameter to NOABEND.

Specify one of the following values for the LOCALSTOP parameter on the SETOPTION command:

ABEND

CA MIM stops with abend codes

NOABEND

CA MIM stops without abend codes.

Default: ABEND

Note: Other operands of the SHUTDOWN command (DUMP and GLOBAL) are not affected by the SETOPTION LOCALSTOP command..

Local Shutdown with Restart Manager

This feature is available when running under the restart manager task and COMPATLEVEL=11.81 or higher. Restart manager provides the capability to restart CA MIM components automatically after a failure, or to perform a planned restart on a single system. To perform a planned restart on a single system, use the SHUTDOWN RESTART command. When used with CA MII, the restart manager maintains global resource integrity throughout the restart process. Local ENQ/DEQ activity is suspended for managed QNAMEs from the instant a failure is detected or a SHUTDOWN RESTART command is issued. Activity remains suspended until the restart is complete and normal global operation resumes.

Local Shutdown with JES3

When CA MIM is started with SUB=MSTR on a JES3 system and any of the CA MIM logs are allocated to SYSOUT, enter the following command:

```
F MIM,CLOSELOG MIMLOG
```

Enter this command before shutting down JES3.

How You Shut Down the System in a VCF Environment

If you enter a SHUTDOWN command in a VCF environment, then CA MIM responds in different ways depending on the system that is being shut down. The communication method used (CTCONLY, CTCDASD, or XCF) also affects the response of CA MIM.

If a client system is shut down in a CTCONLY or XCF environment, then follow the procedures outlined in Local Shutdown Processing in this chapter. If the master system is shut down, then CA MIM automatically migrates to the next most preferred master once the original master is freed. If no alternate master system is available, then the action taken by CA MIM depends on the NOMASTER parameter of the GLOBALVALUE statement.

Note: For more information, see the *CA MIM Statement and Command Reference Guide*. In a CTCDASD environment, CA MIM migrates to the physical DASD control file before migrating back to the VCF on the second most preferred master.

WARNING! If the master system is shut down and is not freed from the control file, then CA MIM is not synchronized. The master system has a status of MIGRATING and the client systems have a status of AWAKENING when the DISPLAY SYSTEMS command is issued. The master and client systems retain their status until the FREE command is issued or the system is restarted.

For all virtual control file communication methods, if MOSTPREFERRED=YES is specified, then the VCF migrates to the most preferred system when CA MIM is restarted.

To shut down CA MIM on all systems, enter the SHUTDOWN GLOBAL command.

How You Free Inactive Systems

If you stop CA MIM on a system or you lose the communication to a system, then the active CA MIM address spaces detect this situation. CA MIM then issues warning messages. The operator determines whether it is appropriate to free the inactive system from the MIMplex.

The FREE command is used to advise active CA MIM address spaces that a system has been temporarily removed from the MIMplex.

How You Free Resources and Devices

The FREE command tells CA MIM to free resources and devices held by a system because that system has become temporarily inactive. Use this command if a system has become inactive and you expect the system to rejoin your complex. This command is useful if you are: 1) not starting a system that CA MIM typically runs on, 2) stopping a system, or 3) if a system fails.

A MIM0116 message is issued in response to the FREE command to indicate that this command successfully updated the status of a system in the control file to FREED. A MIM0115 message is returned if a FREE command is issued for a system that is still considered active in the control file. CA MIM does not allow a system to be freed if it still has a status of ACTIVE in the control file.

When CA MIM is shut down on one system, CA MIM on the remaining systems considers resources held on the inactive system to be unavailable at the time of shutdown until a FREE command is issued for the inactive system. For example, if an enqueue that CA MII is managing is held on the system when CA MIM is shut down, CA MII on the remaining systems continues to consider that enqueue as held by the inactive system until a FREE command is issued. The same is true for CA MIA and any managed tape resources held by the system at the time CA MIA becomes inactive. Until the FREE command is issued for the inactive system, these resources are unavailable to the other systems.

Resources not held by the inactive system at the time of CA MIM shutdown are available for use by the remaining CA MIM systems before the FREE command is issued. Also, once the FREE command is issued, all managed resources are available for use by the remaining CA MIM systems. If enqueue and tape activity continues on the system where CA MIM is no longer active, then there is a risk of integrity exposures (corrupted data sets, dual allocations of tape drives) between the inactive system and the remaining CA MIM systems.

Warning! Issue the FREE command only during error recovery, if you are sure that a system has no enqueue activity, reserve activity, or tape activity. Otherwise, an integrity exposure or abend may occur.

If you receive a message about a system being inactive, then follow the procedures for responding to that message rather than simply issuing a FREE command. Procedures for responding to these messages are provided in the chapter “Troubleshooting.”

Example: Freeing resources and devices

To free resources and devices held by a system that has become inactive, or to inform CA MIM that a system will not be joining the complex, issue a FREE command from another system. Specify the system name, alias, or index number of the system that you are freeing on the FREE command. For example, to free system 01, issue this FREE command from another system on which CA MIM is running:

```
FREE 01
```

When you issue this command, any resources or devices held by system 01 are freed. All active systems ignore system 01 until you start CA MIM on that system or until you restart the product on another system and format the control files. If you format the control files at the next start-up, then all systems expect system 01 to join the complex. Therefore, reissue the FREE command if system 01 is still inactive at that time.

Quiesce and Restart CA MIM

When you need to temporarily deactivate CA MIM (for example, while you perform maintenance on a system), use the CA MIM QUIESCE command. Then, use the RESTART command to reactivate CA MIM.

Important! Quiesce CA MIM *only* when you are bringing z/OS down on that system.

Follow these steps:

1. Issue the QUIESCE command.

   CA MIM performs quiesce processing and then goes into a “wait state.” This means that it will not try to access its control file from that system.

   Important! Begin the QUIESCE command with the CA MIM command prefix character or a z/OS MODIFY command. Otherwise, you will quiesce z/OS. If you are concerned about errors, then use the CA MIM DISABLE command to disable the QUIESCE command.

   CA MIM issues the following messages as it quiesces:

   ```
   MIM0280I System sysid QUIESCE pending
   MIM0282I CA-Multi-image Manager system sysid is QUIESCED
   ```

   Other systems display MIM0061 messages because the quiesced system has stopped updating its time stamp in the control file. Do not respond to these messages. Eventually, these systems display MIM0063 messages to tell you that the quiesced system seems to be “asleep.” Do not issue the FREE command for the quiesced system. Otherwise, CA MIM issues a U0322 abend code and MIM0112 message when you restart the quiesced system.

2. Quiesce z/OS and perform any required tasks while your system is down.

3. Reactivate your hardware to restart z/OS.

4. Issue a RESTART command.

   This restarts CA MIM on your quiesced system. CA MIM removes highlighting for the MIM0282I messages and reactivates itself. When systems have synchronized and the restart is complete, the following message is issued:

   ```
   MIM0281I  system sysid RESTART complete
   ```

WAITSTATE and SHUTDOWN WAIT

To preserve data integrity, there are situations where you may need to shut down CA MIM in a disabled wait state. Two parameters have been added to CA MIM to control and define how CA MIM enters a disabled wait state:

- The MIMINIT WAITSTATE statement defines the conditions under which CA MIM will enter a disabled Wait state. You can also set the value for this keyword in the PARM field in the CA MIM startup JCL PROC sample.

  Important: Use this parameter with careful consideration. Specifying any value other than NEVER (default) puts CA MIM into a disabled state that abruptly halts the entire system. Under some circumstances, this is a good choice, because it is the only way to ensure the integrity of shared data.

  You should choose a value other than NEVER only when GDIF is activated. All other CA MIM components (TPCF, GTAF, EDIF, ECMF, and ICMF) can all be stopped safely, and do not require the wait state to ensure data integrity.

  Note: The SETOPTION SHUTDOWN command will NOT be honored when WAITSTATE is set to anything other than NEVER.

- The SHUTDOWN WAIT command sets the status of the local system to a freed state, then puts the system into a disabled wait state.

For more information about these keywords, see the *CA MIM Statement and Command Reference Guide*.

Command Security Considerations

CA MIM provides the capability to control access to its commands through command validation using the operating system security system software.

CA MIM command validation support is activated by specifying the statement MIMINIT SAFCMDAUTH=ON. If this feature is activated but the security system does not support SAF operator command validation, then CA MIM terminates during initialization.

Once the security system is activated, CA MIM extracts the user ID of the command issuer and sends a command authorization call to the security system software using the OPERCMDS class. CA MIM builds a two-level entity name comprised of a subsystem identifier prefix (the default is MIMGR), followed by the full name of the command verb.

For example, if you issue a DISPLAY command, then the entity name is MIMGR.DISPLAY. You can provide your own one- to eight-character prefix by specifying the MIMINIT SAFPREFIX statement. Optionally, you can request that CA MIM use the job name of its address space (user ID in z/VM) as the subsystem prefix value in the entity name. This can be useful if you run different CA MIM facilities in different address spaces, and you want to limit command access based on the facility.

CA MIM requires a user to have READ access to permit the execution of DISPLAY commands. All other commands require UPDATE access. CA MIM interprets a "no decision" return code from the security system (that is, no profile exists) as a denial for command access.

Note:

- For a list of CA MIM commands, the required SAF authorization to execute the commands, and the SAF entity name used to validate the authority of the command issuer, see the next section.

  CA MIM provides a user exit point in the command authorization processing prior to the system authorization call for the command. This installation exit called MIMATHXT, can unconditionally permit the command, unconditionally reject the command, or continue normally, in which case the command is conditionally executed based on the security system authorization decision.

More information:

User Exits (see page 167)

Commands, Access Authorities, and Entity Names

This table lists all CA MIM commands, the SAF authorization level that is required to execute the command, and the SAF entity name that is used to validate the command issuers UTOKEN (User Token).

Note: The *MIMGR* portion of the entity name is the default entity name prefix value. You can override the prefix value with the MIMINIT SAFPREFIX statement.

Command	SAF Authority	SAF Entity Name
ACTIVATE	UPDATE	*MIMGR*.ACTIVATE
ADDQNAME	UPDATE	*MIMGR*.ADDQNAME

Command	SAF Authority	SAF Entity Name
ALLOCATE	UPDATE	*MIMGR*.ALLOCATE
ALTER	UPDATE	*MIMGR*.ALTER
AUTHCHK	UPDATE	*MIMGR*.AUTHCHK
COLLECT	UPDATE	*MIMGR*.COLLECT
CP	UPDATE	*MIMGR*.CP
CTC	UPDATE	*MIMGR*.CTC
DEALLOCATE	UPDATE	*MIMGR*.DEALLOCATE
DEFALIAS	UPDATE	*MIMGR*.DEFALIAS
DELQNAME	UPDATE	*MIMGR*.DELQNAME
DEQJOB	UPDATE	*MIMGR*.DEQJOB
DIAGNOSE	UPDATE	*MIMGR*.DIAGNOSE
DISPLAY ECMF	READ	*MIMGR*.DISPLAY
DISPLAY EDIF	READ	*MIMGR*.DISPLAY
DISPLAY GCMF	READ	*MIMGR*.DISPLAY
DISPLAY GDIF	READ	*MIMGR*.DISPLAY
DISPLAY GTAF	READ	*MIMGR*.DISPLAY
DISPLAY ICMF	READ	*MIMGR*.DISPLAY
DISPLAY MIM	READ	*MIMGR*.DISPLAY
DISPLAY TPCF	READ	*MIMGR*.DISPLAY
DOM	UPDATE	*MIMGR*.DOM
DROPSYS	UPDATE	*MIMGR*.DROPSYS
DUMP GCMF	UPDATE	*MIMGR*.DUMP
DUMP GDIF	UPDATE	*MIMGR*.DUMP
DUMP GTAF	UPDATE	*MIMGR*.DUMP
DUMP ICMF	UPDATE	*MIMGR*.DUMP
DUMP MIM	UPDATE	*MIMGR*.DUMP
DUMP TPCF	UPDATE	*MIMGR*.DUMP
EDITEST	UPDATE	*MIMGR*.EDITEST
EXEMPT	UPDATE	*MIMGR*.EXEMPT
FREE	UPDATE	*MIMGR*.FREE

Command	SAF Authority	SAF Entity Name
FREECONS	UPDATE	*MIMGR*.FREECONS
GLOBALVALUE	UPDATE	*MIMGR*.GLOBALVALUE
ICMF	UPDATE	*MIMGR*.ICMF
IDEFSYS	UPDATE	*MIMGR*.IDEFSYS
LINK	UPDATE	*MIMGR*.LINK
MSGTABLE	UPDATE	*MIMGR*.MSGTABLE
MIGRATE	UPDATE	*MIMGR*.MIGRATE
QUIESCE	UPDATE	*MIMGR*.QUIESCE
REMOVE	UPDATE	*MIMGR*.REMOVE
RESTART	UPDATE	*MIMGR*.RESTART
RESYNCH	UPDATE	*MIMGR*.RESYNCH
SETOPTION ECMF	UPDATE	*MIMGR*.SETOPTION
SETOPTION EDIF	UPDATE	*MIMGR*.SETOPTION
SETOPTION GCMF	UPDATE	*MIMGR*.SETOPTION
SETOPTION GDIF	UPDATE	*MIMGR*.SETOPTION
SETOPTION ICMF	UPDATE	*MIMGR*.SETOPTION
SETOPTION MIM	UPDATE	*MIMGR*.SETOPTION
SETOPTION TPCF	UPDATE	*MIMGR*.SETOPTION
SHUTDOWN	UPDATE	*MIMGR*.SHUTDOWN
SYSDUMP	UPDATE	*MIMGR*.SYSDUMP
USERDATA	UPDATE	*MIMGR*.USERDATA
VARY	UPDATE	*MIMGR*.VARY
VCF	UPDATE	*MIMGR*.VCF

How You Activate New Features with ACTIVATE COMPATLEVEL

Normally all systems in a CA MIM complex run at the same release level. However, when new CA MIM releases are announced it can be necessary to run multiple releases of CA MIM.

When a complex is running different releases of CA MIM, COMPATLEVEL controls the activation of selected features. The compatibility level specification permits different releases of CA MIM to coexist in the same complex. COMPATLEVEL controls the activation or suppression of features.

Specify the compatibility level using:

- The MIMINIT COMPATLEVEL statement, which you specify in the MIMINIT member of the MIMPARMS data set.
- The ACTIVATE COMPATLEVEL command, which you issue to change the COMPATLEVEL of the complex.

The MIMINIT COMPATLEVEL statement denotes the level at which the CA MIM complex operates, regardless of the releases on the individual systems that make up the complex. The COMPATLEVEL value must be the same on all systems in the CA MIM complex.

Example 1: Using COMPATLEVEL with active systems

Starting CA MIM r12.0 on a system joining an existing CA MIM r11.9 MIMplex currently running COMPATLEVEL=11.9, assume:

SYSA

Running CA MIM r11.9 at COMPATLEVEL=11.9

SYSB

Running CA MIM r11.9 at COMPATLEVEL=11.9

SYSC

Running CA MIM r11.9 at COMPATLEVEL=11.9

Assuming you want to start CA MIM r12.0 on SYSC, the process would be as follows:

1. Stop the CA MIM r11.9 task on SYSC.
2. Start CA MIM r12.0 task on SYSC with COMPATLEVEL=11.9 specified in MIMINIT.

CA MIM r12.0 on SYSC then joins the existing COMPATLEVEL=11.9 MIMplex (SYSA and SYSB) without any errors.

In order to roll out CA MIM r12.0 to SYSA and SYSB:

1. Stop CA MIM r11.9 task on SYSB.

2. Start CA MIM r12.0 task on SYSB with COMPATLEVEL=11.9 specified in MIMINIT.
3. Stop CA MIM r11.9 task on SYSA.
4. Start CA MIM r12.0 task on SYSA with COMPATLEVEL=11.9 specified in MIMINIT.

CA MIM r12.0 is now running on all three systems at COMPATLEVEL=11.9. To move to COMPATLEVEL=12.0 (without formatting MIM control or checkpoint files, which would require a global shutdown):

1. Issue the following MIM command on any single system in the MIMplex: ACTIVATE COMPATLEVEL=12.0
2. Code COMPATLEVEL=12.0 in the MIMINIT member within the MIM parmlib.

Note: COMPATLEVEL=12.0 cannot be specified on any CA MIM statement or command until all systems in the MIMplex are running CA MIM r12.0.

Example 2: Using COMPATLEVEL with inactive systems

In this example:

- CA MIM r12.0 is up and running on SYSA and SYSB and at COMPATLEVEL=11.9
- SYSC is currently down and FREED.

The ACTIVATE COMPATEVEL command expects all systems that are coded on the DEFSYS statement to be active. Use the FORCE option on the ACTIVATE command.

For example, if you entered the ACTIVATE COMPATLEVEL command without the FORCE option, you would see the following messages:

```
F MIM,ACTIVATE COMPATLEVEL=12.0
MIM0067 COMMAND ACTIVATE
MIM0628E ACTIVATE command aborted; all defined systems not active
```

CA MIM rejected the command, because CA MIM is not active on SYSC. To avoid the rejection, append the FORCE operand:

```
F MIM,ACTIVATE COMPATLEVEL=(12.0,FORCE)
MIM0067 COMMAND ACTIVATE
MIM0616I ACTIVATE COMPATLEVEL=12.0 has been scheduled
MIM0619I COMPATLEVEL 12.0 accepted
MIM0620I COMPATLEVEL 12.0 activation in progress
MIM0621I COMPATLEVEL 12.0 activation complete
```

The next time CA MIM is started on SYSC, start CA MIM r12.0 with COMPATLEVEL=12.0.

How You Use ACTIVATE COMPATLEVEL with Checkpoint Files

When using the dynamic compatibility level feature with checkpoint files, format the local checkpoint file to start CA MIM if either of the following conditions is true:

- The system on which you start CA MIM was never included in the activate process.

  For example, you are running CA MIM r11.9 on three systems (SYSA, SYSB, and SYSC). The current COMPATLEVEL specification is 11.9. On SYSC, CA MIM is not active and is therefore currently freed.

  Shut down CA MIM r11.9 one system at a time and bring up CA MIM r12.0 with COMPATLEVEL=11.9 (SYSA and SYSB). You do not start CA MIM on SYSC (which is in a freed state). You then issue the command:

  ```
  ACTIVATE COMPATLEVEL=(12.0,FORCE)
  ```

  The FORCE option tells CA MIM to ignore the freed systems (SYSC in this case). Now you have SYSA and SYSB running at the CA MIM r12.0 level. After issuing the ACTIVATE command update the MIMPARMS data set with MIMINIT COMPATLEVEL=12.0. If you decide to bring CA MIM r12.0 up on SYSC, format the local checkpoint file using FORMAT=CHKPT on the startup command.

- You have issued a successful ACTIVATE COMPATLEVEL command and decide to revert to the previous release using a control file format.

  For example, you are running CA MIM r12.0 on three systems using COMPATLEVEL 12.0. You decide to revert to release 11.9 by a global shutdown and a control file format.

  In this case, the proper procedure is to update the MIMPARMS data set with COMPATLEVEL=11.9. Shut down the systems globally, and start with a FORMAT=BOTH on the first system, and FORMAT=CHKPT on all subsequent systems.

How You Activate New Features with ACTIVATE FEATURE

New features can be dynamically activated using the ACTIVATE FEATURE command. This command prevents the need for a global shutdown and control file format. The ACTIVATE FEATURE command works similar to the ACTIVATE COMPATLEVEL command. The biggest difference is that new FEATURES are not tied to a release boundary and their activation is optional.

Specify the compatibility level using:

- The MIMINIT FEATURE statement, which you specify in the MIMINIT member of the MIMPARMS data set.
- The ACTIVATE FEATURE command, which you issue to activate a new feature.

The MIMINIT FEATURE statement denotes which features should be activated at startup. The FEATURE value must be the same on all systems in the CA MIM complex.

Example 1: Using FEATURE with active systems

1. Ensure that CA MIM has all necessary PTFs or Releases installed containing the Feature you wish to activate. This must be true on all systems.
2. Enter F MIM,ACTIVATE FEATURE=featurename (on one system only)

   ```
   MIM0684I ACTIVATE FEATURE=featurename has been scheduled
   MIM0619I ACTIVATE FEATURE=featurename accepted
   MIM0620I ACTIVATE FEATURE=featurename in progress
   MIM0621I ACTIVATE FEATURE=featurename complete
   ```
3. Update the MIMINIT statement in the init member with FEATURE=featurename

Example 2: Using ACTIVATE FEATURE with inactive systems

In this example:

- CA MIM is up and running on SYSA and SYSB with no Features active.
- SYSC is currently down and FREED.

The ACTIVATE FEATURE command expects all systems that are coded on the DEFSYS statement to be active. Use the FORCE option on the ACTIVATE command.

For example, if you entered the ACTIVATE FEATURE command without the FORCE option, you would see the following messages:

```
F MIM,ACTIVATE FEATURE=featurename
MIM0067 COMMAND ACTIVATE
MIM0628E ACTIVATE command aborted; all defined systems not active
```

CA MIM rejected the command, because CA MIM is not active on SYSC. To avoid the rejection, append the FORCE operand:

```
F MIM,ACTIVATE FEATURE=(HYPERSTAR,FORCE)
Command ACTIVATE
MIM0684I ACTIVATE FEATURE=featurename has been scheduled
MIM0619I ACTIVATE FEATURE=featurename accepted
MIM0620I ACTIVATE FEATURE=featurename in progress
MIM0621I ACTIVATE FEATURE=featurename complete
```

The next time CA MIM is started on SYSC, start CA MIM with MIMINIT FEATURE=featurename.

How You Use ACTIVATE FEATURE with Checkpoint Files

When using the ACTIVATE FEATURE command with checkpoint files, format the local checkpoint file when starting CA MIM if either of the following conditions is true:

- The system on which you start CA MIM was not active at the time of the ACTIVATE FEATURE command.

How to DEACTIVATE a FEATURE

If you decide to deactivate a Feature, simply use the DEACTIVATE command. In this example there are external systems in a FREED state, so the FORCE operand is necessary:

```
F MIM,DEACTIVATE FEATURE=(featurename,FORCE)
MIM0067I Command DEACTIVATE
MIM0684I DEACTIVATE FEATURE=featurename has been scheduled
MIM0619I DEACTIVATE FEATURE=featurename accepted
MIM0620I DEACTIVATE FEATURE=featurename in progress
MIM0621I DEACTIVATE FEATURE=featurename complete
```

Sysplex Considerations

Each CA MIM component has specific considerations for operating in a sysplex. For details, see the appropriate *Programming Guide* for each component.

Performance Considerations

CA MIM provides resource serialization services among multiple z/OS operating systems. CA MII lets sites share DASD resources, CA MIA lets sites share tape devices. CA MIC allows console data (messages and commands) to be routed across systems. A number of factors, both internal and external to the CA MIM address space, can cause CA MIM to provide less than optimal global ENQ service times.

External factors, like dispatching priority have a direct impact on the performance of the CA MIM address space. These factors can degrade CA MIM ENQ service times.

The following CA MIM internal parameters can also affect performance:

- Type of cross-system communication method selected
- CA MIM features activated
- Type and volume of managed workload
- Page fixing CA MIM control block storage
- Offloading CA MIM work to zIIP engines

This section examines the internal and external factors that can affect CA MIM performance in your environment.

CA MIM Driver

The CA MIM Driver is the primary control program in the CA MIM address space. The CA MIM Driver supervises the overall activities of the CA MIM address space, regardless of which CA MIM components or facilities are activated. CA MIM Driver code in a given CA MIM address space provides the following:

- Initialization, synchronization, and termination processes
- Control file error recovery and migration processes
- Control file serialization and I/O operations
- Command and message processing
- Virtual storage management
- Subtask management
- Lock management
- Diagnostic tracing
- SMF recording

Control File Internals

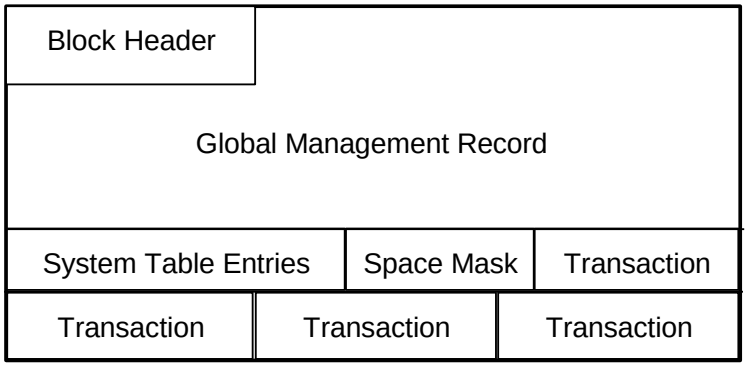
There are two primary structures found in all CA MIM control files. The first structure is the *Global Management Record (GMR),* which contains information about the status of every CA MIM address space in the MIMplex. And secondly, CA MIM *transactions* are routed through the CA MIM control file and represent managed resources activity. The Global Management Record (GMR) and CA MIM transactions in this chapter discuss these internal CA MIM control file structures in detail.

Global Management Record (GMR)

The internal structure of the control file is the same regardless of the type of control file medium (DASD, coupling facility, CTC, or XCF) being used. The internal structure of a CA MIM control file always consists of a GMR followed by transactions. The GMR is created as part of the control file formatting process and contains information regarding the following:

- Identity of each system in the MIMplex (system name, alias, index number)
- Current status of the CA MIM address space on each system: active, freed, migrating, and so on
- Time of last control file access for every CA MIM address space
- Control file structural information (BLKSIZE, COMPATLEVEL, and so on)
- Control file statistics (access rates, transaction counts)

The following illustration depicts the internal structure of a CA MIM control file.



The first block of a control file contains the GMR and a number of CA MIM transactions. All subsequent blocks in the CA MIM control file are used to temporarily store CA MIM transactions. Transactions are deleted from the control file as soon as all CA MIM address spaces have read and processed them.

Control File Cycle

The term *control file cycle* refers to the process of an individual CA MIM address space obtaining serialization to the control file, reading, processing, and writing data to and from the control file, and then finally releasing the control file. A single control file cycle is all that is required by a CA MIM address space to update its “picture” of all outstanding resources in the MIMplex. The *star topology* has been employed by CA MIM for over 20 years and is currently used with both the *physical* and *virtual* control file architectures.

The internal CA MIM processes involved in every control file cycle is illustrated here:

```
RESERVE the control file

 Read and copy the first block (GMR) into local storage

 Check the state of the MIMplex by interrogating the GMR

 Read and direct incoming transactions to each component driver (MIA, MII, MIC)

 Receive outbound transactions from each component driver

 Update the GMR residing in private storage

 Write new GMR and transactions out to the control file

RELEASE the control file
```

The duration of a single control file cycle depends on a number of factors. The more transactions there are to process, the longer the duration of a control file cycle. The volume of transactions is governed by the amount of resource activity on a system combined with whether CA MIM is managing those resources. CA MIM parameters tell CA MIM what type of resources it should be managing. The more resources you tell CA MIM to manage, the more CA MIM transactions that are generated, and the longer each control file cycle.

One way to optimize CA MIM is to ensure your CA MIM parameters involving transaction processing are defined correctly. There are a number of CA MIM parameter definitions that may cause unnecessary CA MIM transactions to be created, or may cause unnecessary CA MIM address space overhead, which elongates the control file cycle times.

You can also expect elongated CA MIM control file cycle times on systems that are CPU constrained, real storage constrained, or I/O bound.

Another aspect of control file cycle processing that affects the cycle completion time is that the frequency of control file cycles is different on every system in the MIMplex. Control file cycles are driven by either of two events:

- Managed workload activity
- A timer expiration

CA MIM on a system with a large amount of managed enqueue requests will access the control file more frequently than CA MIM on a system that has very little managed enqueue requests. If you have a MIMplex comprised of systems running a range of heavy to light enqueue workloads, then CA MIM address spaces on those systems will be accessing the control file at different frequencies. In other words, CA MIM on a production system may perform control file cycles at a rate of 25 times per second, while CA MIM on a test system may perform a control file cycle only one time per second. Severely disproportionate control file access rates by CA MIM on different systems will elongate individual control file cycle times and, in turn, degrade CA MIM transaction processing times.

CA MIM Transactions

The other main structures found in CA MIM control files are CA MIM transactions. There are over 28 different CA MIM transactions that represent changes to managed resources. Each CA MIM component has transactions unique to that component. The following table illustrates some of the transaction types associated with each CA MIM component:

Component	Transaction Types
CA MIM	GLOBALVALUE changes
CA MIA	Tape device status changes, preferencing
CA MII	Enqueues, dequeues, conflicts, job requeue

The size of a particular CA MIM transaction is variable in length. For example, the size of a CA MIC cross-system message transaction is dependent on the length of the message text. Likewise, the size of a CA MII enqueue transaction is dependent on the resource name length (RNAME): the longer the data set name, the longer the CA MII transaction. In its simplest form, an individual CA MIM transaction is mapped something like this:

Transaction Map
Size
Type
Flags
Routing Mask
Transaction-specific Data

The routing mask field is used to direct the transaction to specific systems. For example, if a user on SYSA is granted access to a particular data set, then CA MII on SYSB and SYSC needs to be informed of this fact. In this case, CA MII on SYSA creates a transaction having a routing mask for SYSB and SYSC, and writes the transaction to the CA MIM control file. Once CA MIM on SYSB and SYSC have read and processed this transaction, it is deleted from the control file. Remember that CA MIM transactions are in the control file for a brief period of time-ideally for less than one second. The more often each CA MIM address space accesses the control file, the faster CA MIM transactions are processed and deleted from the control file, so the complex-wide throughput time will be better.

Control File Blocks and the Global-Copy Process

The number of control file blocks in-use at any moment in time is very fluid. In a typical synchronized MIMplex, only one or two control file blocks are used from control file cycle to control file cycle. The largest usage of control file blocks occurs during MIMplex transition states such as when a new CA MIM address space is joining an existing MIMplex, or when a control file migration occurs. During these transition states, CA MIM generates a large volume of transactions to represent all outstanding resources at the moment the transition state began. This allows the newly joining system (or re-synchronizing systems, in the case of a migration) to immediately understand the global resource environment at that moment in time.

The currently active system that detects the newly joining system (and has the current global picture of all managed resources) will initiate a process known as a *global*-copy. The volume of transactions generated during a given global-copy operation is totally dependent on the amount of managed resource activity at the time the global-copy was initiated. Once all global-copy transactions have been placed in the control file, the newly joining CA MIM address space will read, process, and remove them from the control file. At that point, the newly joining system is synchronized with the rest of the MIMplex. Once all global-copy transactions are removed from the control file, the MIMplex resumes to using only one or two control file blocks.

The CA MIM control file must be large enough to handle the volume of transactions generated as part of a global-copy operation. If the control file is too small, then CA MIM attempts to use an alternate control file. If all alternate control files are too small, then CA MIM ends abnormally with a U0095 RC=55. To avoid this abend, you must provide a control file large enough to handle a global-copy operations should it be initiated during peak enqueue processing times. The maximum number of blocks used during a global-copy operation in your environment can be determined by examining the MAX USED field shown with the DISPLAY IO command. CA MIM also provides an early warning feature that provides an alert message when you exceed 50% of the total number of control file blocks.

Note: For more information, see the details on the SETOPTION CFSIZEWARN parameter in the *Statement and Command Reference Guide*.

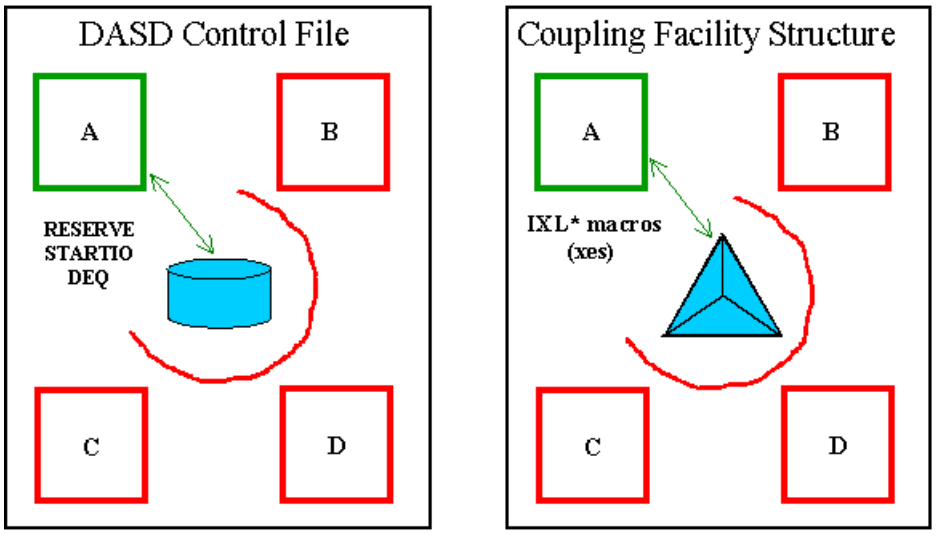
Control File Externals

This section discusses the two fundamental, high-level architectures of cross-system communication used by CA MIM. The two primary types of CA MIM control file architectures are referred to as the *physical* architecture and the *virtual* architecture. While the internal structures of physical and virtual control files are exactly the same, differences do exist as to how CA MIM Driver serializes access and performs I/O operation with each type of control file. This section takes a look at the differences between the CA MIM physical and virtual control file architectures.

Physical Control Files

Physical control files reside on hardware devices that are accessible by all systems in the enterprise. Examples of physical control files would be 1) a control file residing on shared DASD hardware and 2) a list structure residing in a coupling facility. With DASD control files, serializing access is achieved using RESERVE/DEQ macros and I/O operations are performed using the STARTIO macro. With coupling facility control files, serialization is achieved and I/O operations are performed using Cross System Extended Services (XES) IXL* macros. For this reason, the CA MIM coupling facility control files are commonly referred to as *XES control files*.

The following illustration shows these concepts:

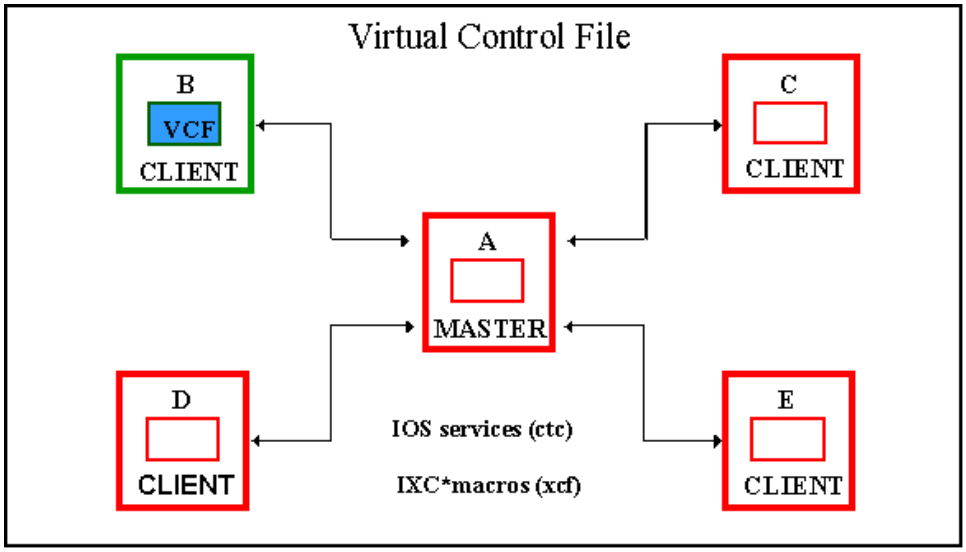


Virtual Control Files

The second type of CA MIM control file is known as a virtual control file (VCF). A VCF resides in the data space of a selected CA MIM address space. The CA MIM address space that controls the VCF is known as the *master* CA MIM address space. The CA MIM master address space is charged with managing or serializing access to the VCF found in its data space. The other systems in the enterprise are knows as CA MIM *client* systems. Client systems request the VCF by sending requests to the CA MIM master address space. The CA MIM master queues and services these requests on a first in, first out basis. When a client request reaches the top of the queue, the CA MIM master will transfer the VCF from its data space to the CA MIM client address space using CA MIM-allocated CTC devices, or through the Cross System Coupling Facility (XCF) component of z/OS. The client CA MIM address space will then read and update the VCF and then transfer the updated copy back to the master. The master will then service the next VCF requestor, which may be another CA MIM client or the CA MIM master itself.

Serialization of the VCF is achieved by the VCF management code running in the master CA MIM address space. When the VCF is being transported using CA MIM-allocated CTC devices, I/O operations are performed using low-level IOS services.

If you chose the CA MIM XCF communication option, VCF I/O operations are performed using XCF IXC macros. When CA MIM presents the VCF to XCF for cross-system transport, XCF may use XCF-allocated CTC devices, or XCF-allocated coupling facility list structures to actually perform the cross-system transfer on the behalf of CA MIM. The transport vehicle employed by XCF is transparent to CA MIM as CA MIM is simply registered with XCF as an application. The following illustration shows these concepts:



Control File Mediums and Performance

The hardware medium that is used to *store* the CA MIM Physical Control File (DASD or a coupling facility) or *transmit* the CA MIM virtual control file (CTC or XCF) has a direct impact on the ability of CA MIM to process work efficiently. The faster CA MIM can read and write transactions to the CA MIM control file, the faster the CA MIM service requestors are dispatched and, in turn, greater system throughput is achieved.

One of the measurements used to gauge control file performance is the Average Cycle Time displayed with the CA MIM DISPLAY IO command. The Average Cycle Time reflects how long a control file cycle takes to complete. Control file cycle time begins at the moment the control file reserve is requested, includes the time it takes to obtain the reserve and read and write transaction data to the control file, and ends when the control file is released. This performance measurement is accurate for both physical and virtual control files.

The following table shows reasonable Average Cycle Times for various CA MIM control file mediums and architectures. This table is intended to show comparative rates between the various mediums and architectures.

Average Cycle Time	Control File Type	Control File Medium	CA MIM address spaces
25 ms	physical	ESCON DASD	all
15 ms	physical	FICON DASD	all
10 ms	virtual	CTC	client
10 ms	virtual	XCF	client
6 ms	virtual	CTC	master
6 ms	virtual	XCF	master
4 ms	physical	XES	all

Due to numerous environmental factors, the Average Cycle Times in a particular enterprise may be slightly higher or lower than those values shown here. You can see from this table that the type of medium used to store or transmit the CA MIM control file has a direct impact on the performance of the product. Again, the faster the control file cycle, the faster CA MIM transactions are processed, and the better the complex-wide enqueue throughput.

How to Set a Preferred Medium for VCF Communication

You can use a combination of XCF and CTC paths at any given time when you use the default value of COMPATLEVEL=12.0.

Set your medium for VCF as follows:

- Display the available VCF paths using the following command:

  ```
  DISPLAY PATH
  ```

- Before you start CA MIM:

  Specify the preferred medium for the VCF communication on the MIMINIT member using the following statement:

  ```
  MIMINIT VCFPREFERENCE statement
  ```

- While CA MIM is executing:

  Change the VCFPREFERENCE using the following command:

  ```
  SETOPT VCFPREFERENCE
  ```

  Note: VCFPREFERENCE is enforced when CA MIM initiates the VCF communication from a client system to the master system. Setting VCFPREFERENCE has no immediate effect on the master system.

If CA MIM currently uses the VCF communication, it automatically selects a path according to the current VCFPREFERENCE setting. However, if a path of the preferred medium is not available, CA MIM uses any available path.

Note: See the *Statement and Command Reference Guide* for more information about VCFPREFERENCE.

How You Tune Control File Access Rates

Control file cycles are driven by either of the following two events:

- Managed workload activity
- Timer expiration

Systems having higher managed workloads cause CA MIM Driver to access the control file more frequently than systems with little managed workload. By default, CA MIM accesses its control file one time per second. It is therefore possible to have CA MIM on some systems accessing the control file 20 or 30 times per second, while CA MIM on other systems is accessing the control file only one time per second. Environments generating such disproportionate control file access rates will not achieve the best possible transaction processing times.

The objective of control file access tuning is to provide a global (MIMplex-wide) balance of control file access rates; this ensures efficient transaction processing and, in turn, optimizes throughput on all systems in the MIMplex.

Tuning control file access rates is most critical for sites using the CA MII component of CA MIM. CA MII transaction processing has the lowest delay tolerance of any of the CA MIM component transactions. In other words, a delay in CA MII transaction processing has a more direct impact on system throughput than CA MIA or CA MIC transactions. If a CA MIA (tape device status) or CA MIC (message or command) takes one second to process, then it has little bearing on system throughout. However, if every CA MII enqueue transaction took 1 second to process, then this would severely degrade a workload-intense production system. Tuning control file access, while important for all CA MIM address spaces, is most critical for customers running the CA MII component of CA MIM. Therefore, this section is geared toward tuning CA MIM address spaces having the CA MII component activated. You can use the DISPLAY FACILITIES command to determine which CA MIM components and facilities are active in a given CA MIM address space.

Tuning control file access rates is important regardless of the control file architecture or medium being used in your particular environment. Balancing control file access rates is desirable regardless of whether you are using a DASD control file, an XES control file, or any of the virtual control file methods. The control file tuning methods outlined here are viable for all CA MIM control file architectures.

Tuning control file access rates is an exercise that should be undertaken several times per year or when changes are made to your MIMplex. Changes to, or shifting of, managed workloads can alter control file access rates such that a previously balanced MIMplex becomes disproportionate once again. It is also recommended that you examine control file performance after processor upgrades are performed, or when systems are added or removed from your MIMplex.

The process of tuning control file access rates involves two steps or phases. The *Analysis* phase involves issuing the CA MIM DISPLAY IO and DISPLAY SERVICE commands on all systems in the MIMplex, and then analyzing the data for indications of unbalanced control file access rates. The *Implementation* phase involves the manual manipulation of the CA MIM tuning parameters to increase or decrease control file access rates on one or more systems. The CA MIM parameters used during the implementation phase are SETOPTION MODE, INTERVAL, and CYCLE. The following sections on control file tuning explain the processes involved in tuning control file cycle rates.

Control File Tuning: Analysis

The objective of the analysis phase is to:

- Examine your control file access rates (frequencies)
- Determine your CA MII enqueue service rates for every CA MIM address space in your MIMplex.

There are two CA MIM commands that provide the information used in the analysis phase: DISPLAY IO and DISPLAY SERVICE.

How You Display Control File Processing Information

The DISPLAY IO command provides information about your control file-processing environment.

Important! Every CA MIM address space has unique control file processing data.

Therefore, issue the DISPLAY IO command on every system in the MIMplex, and examine it on a system-by-system basis. Among the information, provided is:

- Communication method employed
- Number of blocks allocated
- Maximum number of blocks used
- Date and time of last format.

A great deal of information is displayed, however, the most important statistics involve the control file cycle rates and transaction processing data. The following illustration shows the DISPLAY IO data:

```
MIM0039 CONTROL FILE I/O DISPLAY:
  COMMUNICATION=DASDONLY   CURRENT=DASD
  MODE=DEMAND      CYCLES=1           INTERVAL=1.000
  FILE=00 NAME=MIM.PRIMARY                                  UNIT=2DC8
  TOTAL BLOCKS=1200    MAX USED=547     FREE=1199    BLKSIZE=6144
  LAST FORMAT:  2003.119   TOTAL READS=408.463M     WRITES=220.841M
  LAST RESTART AT  07:59:53 ON 2003.119
    COUNT: CYC=15.166M     BLOCKS READ=57.638M8    WRITTEN=27.639M
     XACT READ=88.885M       PROCESSED=44.093M     WRITTEN=26.650M
    AVG:   CYC=0.085302    BLOCKS READ=3.788       WRITTEN=4.120
     XACT READ=95.529        PROCESSED=2.907.000   WRITTEN=1.757
    RATE:  CYC=21.571      BLOCKS READ=165.584     WRITTEN=79.404
     XACT READ=2400.309      PROCESSED=120.025     WRITTEN=76.561
```

The key fields used in the control file tuning analysis phase have been bolded in the preceding illustration. The following describes each of these key fields:

AVG: CYC=.085302

Represents the average control file cycle completion time in seconds. It is comprised of:

- Time required to obtain the control file reserve.
- Read and process any transactions on the file.
- Process any local requests, including:
  - Writing any transactions
  - Releasing the control file
  - Performing any post file processing
  - Wait time before attempting the next control file access.

For example, the control file cycles on this system are taking 85 milliseconds, which is not acceptable for a 3390 DASD control file.

Note: The faster the control file cycle time, the more optimized CA MIM performance.

BLOCKS READ=3.788 WRITTEN=4.120

Indicates, on average, how many control file blocks are read from and written to the control file *during each control file cycle.* Optimally tuned MIMplexes have a value of less than 1.1 for each of these fields when COMMUNICATION=DASDONLY is specified.

A higher value usually means one or more CA MIM address spaces are not accessing the control file frequently enough. The transactions destined for those systems are constantly read and rewritten by CA MIM on your more active, workload intense systems. Values of 3.788 and 4.120 as shown in the example are not considered acceptable.

If COMMUNICATION=CTCDASD/CTCONLY/XCF is specified, then it may be acceptable to have BLDS READ and BLKS WRITTEN larger than 1.1. When CA MIM uses a virtual control file, the data is read and written using VCF buffers. The MIMINIT VCFBUFFERSIZE statement, with a default of 32768, determines the size of this buffer. The maximum number of blocks of data are "packed" into this VCF buffer, when it reads and writes data to the VCF master system. Current BLKSIZE value determines the amount of data in each block. Use the DISPLAY IO command to displays this value.

Examples:

AVG BLOCKS READ value is 2.0 and MIMINIT BLKSIZE is 6144, then CA MIM usually reads 12288 bytes of data during each control file transaction.

AVG BLOCKS READ value is 7.0 and MIMINIT BLKSIZE is 6144, then CA MIM usually reads 43008 bytes of data during each control file transaction.

VCFBUFFERSIZE value is 32768. CA MIM is not able to fit all of the data into one VCF buffer, and requires multiple VCF buffers for each control file access. The number of I/Os required are increased to complete each control file cycle. A more appropriate value for VCFBUFFERSIZE in this example would be 44000.

RATE: CYC=21.571

This field represents the number of control file cycles per second. The frequency of control file cycles is driven by either of two events:

- Managed workload activity
- A timer expiration

CA MIM on a system with large amounts of managed enqueue requests, accesses the control file more frequently.

Each CA MIM address space on a MIMplex containing systems running heavy to light enqueue workloads ranges, access the control file at different frequencies..

CA MIM on a production system performs control file cycles at a rate of 25 times per second. On a test system, CA MIM performs a control file cycle only one time per second. Severely disproportionate control file access rates by CA MIM on different systems elongate individual control file cycle times, and degrade CA MIM transaction processing times. Acceptable values for this field vary from site to site depending upon a number of factors. If this value is too high, then CA MIM address space CPU consumption increases. If this value is too low, then MIMplex throughput suffers.

For most sites, this value is set to 30 or less on any system. Some sites, however, use XES control files with average cycle rates as high as 45 control file cycles per second with great success. The number of control file cycles per second can be increased or decreased on a system-by-system basis. For more information see, Control File Tuning: Implementation (see page 139).

XACT READ=2400.309 PROCESSED=120.025

These fields represent the average number of transactions read, and the number of transactions processed by this system, during *each control file cycle.* In CA MIM, transactions remain in the control file for a very short time. Once a transaction is read and processed by all CA MIM address spaces, that transaction is removed from the control file.

During every control file cycle, the CA MIM address space reads in every transaction from the control file. CA MIM examines the routing mask to determine the transaction needs for processing by this system. A transaction previously processed by an earlier control file cycle is written back to the control file with new transactions. The process of reading constantly, but not process the transactions of other systems, causes the Transactions Read to Processed ratio to increase.

Ideally, you want every CA MIM address space to process at least *one* out of every ten transactions it reads from the control file. The pre*f*erred Transactions Read to Processed ratio is within 10 to 1. This ratio can be calculated by dividing the Transactions Read value by the Transactions Processed value. In the above example, during each control file cycle this system reads, on average, 2400 transactions from the control file. Where only 120 transactions are destined for this system. This means that during every control file this system is reading 2280 transactions that are not for this system. The Transactions Read to Processed ratio in the example is roughly 20 to 1, which is not acceptabl*e*. Increasing or decreasing the number of control file cycles per second on certain systems will correct this problem.

How You Display Enqueue Request Information

The DISPLAY SERVICE command provides information about enqueue requests being handled by CA MII. The data is unique to each CA MIM address space in your complex. Therefore, you should issue the DISPLAY SERVICE command on every system in the MIMplex, and examine the displays on each system. The most important information is the *number of enqueue requests* and the *time per enqueue request*. The following illustration shows the DISPLAY SERVICE data:

```
MIM1021 GDIF SERVICE DISPLAY
    REQUESTS    TIME/REQUEST    RATE/SECOND   RATE/CYCLE      SINCE
   16,896,706     0.075351       59.900       1.180  08:00:37 2003.119
```

The key fields used in the control file tuning analysis phase have been bolded in the preceding illustration. The following describes each of these key fields:

REQUESTS: 16,896,706

The REQUESTS field represents the number of managed enqueue requests that have been handled by CA MII on the system. Examining this field on each system helps you determine which systems are your workload-intense systems in terms of enqueue processing. You will find that the number of enqueue requests on a system has a direct correlation to the AVG: CYC value (DISPLAY IO command) in that the more enqueue requests on a system, the more control file cycles per second that system will perform. Production systems will have more enqueue requests and control file cycles per second than test or development systems.

TIME/REQUEST: 0.075351 SECS

The TIME/REQUEST field represents the average number of seconds it takes CA MII to service an individual managed enqueue request. A wide variety of internal and external factors have an impact on the CA MIM enqueue service times. Acceptable values for this field are between .050 (50ms) and .070 (70ms). Tuned MIMplexes will have values lower than the acceptable standards, and many sites enjoy values in the single-digit range (6ms). Our example of 75.351ms is definitely not in the range of acceptable enqueue service times.

Analysis Phase Summary

By examining the DISPLAY IO and DISPLAY SERVICE statistics for every system in the MIMplex, you should be able to identify certain systems that are not performing control file cycles often enough to keep pace with your workload-intense systems. Because enqueue workload is the primary factor in driving control file cycles, CA MIM on systems with low enqueue workloads do not perform control file cycles as often as CA MIM tasks on workload-intense systems. CA MIM on low workload systems may access the control file only one time per second, while CA MIM on workload-intense systems may perform control file cycles 25 times per second.

Because of the disproportionate cycle rates (access frequencies), CA MIM on workload-intense systems will have to constantly reread and rewrite transactions destined for systems that are not accessing the control file often enough. The process of continuously rereading an rewriting latent transactions causes more control file blocks to be used, more control file I/O operations, elongated control file cycle times, inefficient transaction processing rates, and less than optimum throughput.

Based on the data in the DISPLAY IO and DISPLAY SERVICE illustrations used earlier in this section, the following analysis can be made: the MIMplex is communicating across systems using a DASD control file. It is performing control file cycles at a rate of roughly 22 times per second. Control file cycles are most likely driven by the fact that it has a fairly high enqueue workload (the number of requests is 16,896,706). This MIMplex appears to be poorly tuned because of the following facts:

- The number of blocks read and written are more than the acceptable value of 1.1 for COMMUNICATION=DASDONLY.
- The average control file cycle time is longer than the acceptable value of 80ms (for 3390 DASD control file).
- The transactions read to transactions processed ratio is more than the acceptable 10:1.
- The enqueue time per request is above the highest acceptable value of 70ms. This is the most important factor.

In short, there seems to be one or more external systems in the MIMplex that are not accessing the control file often enough, and this system is constantly having to reread and rewrite the transactions of that system. The DISPLAY IO and DISPLAY SERVICE data from the external systems should be examined to determine which system is not accessing the control file often enough.

Note: To see how this can be corrected, see Control File Tuning: Example in this chapter.

The purpose of the analysis phase is to help you identify which systems may not be accessing the control file often enough in your MIMplex. It can also be used to identify which systems may be dominating the control file due to enormous enqueue workloads, although this is rare. In any case, CA MIM parameters can be defined to increase or decrease control file cycles if the analysis phase shows CA MIM on certain systems to be accessing the control file too often, or not enough. The control file tuning sections on parameters and implementation in this chapter explain how to use these CA MIM parameters to increase or decrease control file cycles on specific systems.

Control File Tuning: Parameters

Balancing control file cycles across the MIMplex will improve transaction processing efficiency, reduce control file block count usage, reduce the average control file cycle time, reduce individual enqueue service time, and improve overall MIMplex throughput. The objective of this section is to provide an understanding of the CA MIM parameters that are used to regulate control file cycle rates on one or more systems in the MIMplex. The CA MIM SETOPTION commands can be used to dynamically alter the CA MIM internal control file cycle processes. The result of these commands is to either increase or decrease control file cycles on a given system. Let us take a look at the parameters used to tune CA MIM control file cycle rates.

SETOPTION MODE=DEMAND/GROUPS

When running in DEMAND mode, mostly local CA MIA or CA MII resource activity drives local control file cycles. If no CA MIA or CA MII activity is present, then CA MIM Driver performs a control file cycle at the expiration of a timer value, which is calculated based on your SETOPTION CYCLE and INTERVAL parameters. DEMAND is the default processing mode as it allows CA MII enqueue activity to be handled in the most expedient manner. When running in GROUPS mode, local control file cycles are not driven by local managed resource activity, but by a timer expiration. GROUPS mode processing waits for a timer to expire, then checks to see if there are local CA MIA or CA MII transactions to propagate (any that have been queued/grouped while waiting for the timer to expire), and then performs a control file cycle. Because DEMAND mode processing performs a control file cycle as soon as access to a locally managed resource is requested (on demand), it is the more preferred mode of operation.

DEMAND MODE can be used to increase control file cycles on systems that do not have enough CA MIA or CA MII activity to keep pace with other more heavily loaded systems. When running in DEMAND mode, CYCLE=1, and reducing the INTERVAL value from its default of 1, you can force control file cycles to occur more frequently. The following table illustrates the number of control file cycles that occur when you set the INTERVAL parameter to a particular value when in DEMAND mode.

Using this technique you can increase the number of control file cycles to an appropriate level. For example, if you found that CA MIM on a system was performing control file cycles only one time per second, you could issue a SETOPT MODE=DEMAND,CYCLE=1, INTERVAL=.1 command on that system and CA MIM on that system would immediately begin performing control file cycles at a rate of 10 times per second.

Interval Value	Number of Control File Cycles per Second
1.00	1
0.50	2
0.20	5
0.15	7
0.10	10
0.08	13
0.06	17
0.05	20
0.04	25
0.03	33

GROUPS MODE is typically used to decrease the number of control file cycles. On rare occasions, some systems dominate the control file due to the fact that they have so much managed enqueue activity that they perform control file cycles at an excessive rate. This causes other systems to be locked out of the control file resulting in intermittent MIM0100 or MIM0200 messages. In these cases, it may be recommended that you place CA MIM on that system in GROUPS mode, set CYCLE to 1, and use the INTERVAL value to reduce the control file access rate. The preceding table illustrates the number of control file cycles that will occur when you set the INTERVAL parameter to a particular value when in GROUPS mode and have CYCLE set to 1.

Using this technique, you can decrease the number of control file cycles to an appropriate level. For example, if you found that CA MIM on a system was performing control file cycles at an excessive rate of 60 times per second when in DEMAND MODE, you could issue a SETOPT MODE=GROUPS,CYCLE=1, INTERVAL=.04 command on that system and CA MIM would immediately begin performing control file cycles at a decreased rate of only 25 times per second.

While running CA MIM in GROUPS mode on a system will reduce the number of control file cycles and reduce CA MII address space CPU consumption, it will also degrade CA MII enqueue service times. Therefore, placing a CA MIM address space in GROUPS mode should be performed only after consulting with Technical Support.

Control File Tuning: Implementation

In the previous sections you learned that you can use the DISPLAY IO and DISPLAY SERVICE commands to analyze control file cycle and transaction processing rates. You also learned that you could use the SETOPTION MODE, CYCLE, and INTERVAL parameters to increase or decrease CA MIM control file cycle rates on individual systems. In this section, you learn the procedures that you can use to tune control file access rates in your MIMplex.

1. Although not mandatory, it is best to begin with CA MIM having the defaults of SETOPTION MODE=DEMAND, CYCLE=1, INTERVAL=1. Your current values for these CA MIM parameters can be determined by issuing the DISPLAY IO command on every system. If you find that you are not running with these defaults, then you can execute the SETOPTION command to change these parameters to their defaults.
2. Issue DISPLAY IO=RESET and DISPLAY SERVICE=RESET to CA MIM on every system. This will re-initialize the performance statistics and will allow all CA MIM address spaces to begin accumulating statistics from the same starting point.
3. Allow the statistics to accumulate for at least a 24-hour period. This lets CA MIM to acquire statistics for all shifts, and ideally includes your heaviest workload period.
4. Issue the DISPLAY IO and DISPLAY SERVICE commands on all systems. Print the statistics.

   You will notice that the display data has two parts, one containing statistics since CA MIM startup, and the other containing statistics since last reset. You will want to use the statistics accumulated since the last reset command was issued.

   Analyze the statistics. Determine which systems, if any, need to be performing more control file cycles. Usually, control file access tuning involves increasing control file cycles only on one or two systems.

   Note: For help with interpreting this data, see Control File Tuning: Analysis in this chapter. If you would like assistance in analyzing your MIMplex statistics, contact CA MIM Technical Support.
5. Issue a SETOPTION MODE, CYCLE, or INTERVAL command on the selected system to adjust control file access rates appropriate to your environment. The command SETOPTION INTERVAL=.1 provides a control file cycle rate of 10 accesses per second and is a safe starting point when it is desirable to increase the number of cycles on an inactive system. The SETOPTION CYCLE=1 should always used unless otherwise directed by CA MIM Technical Support.

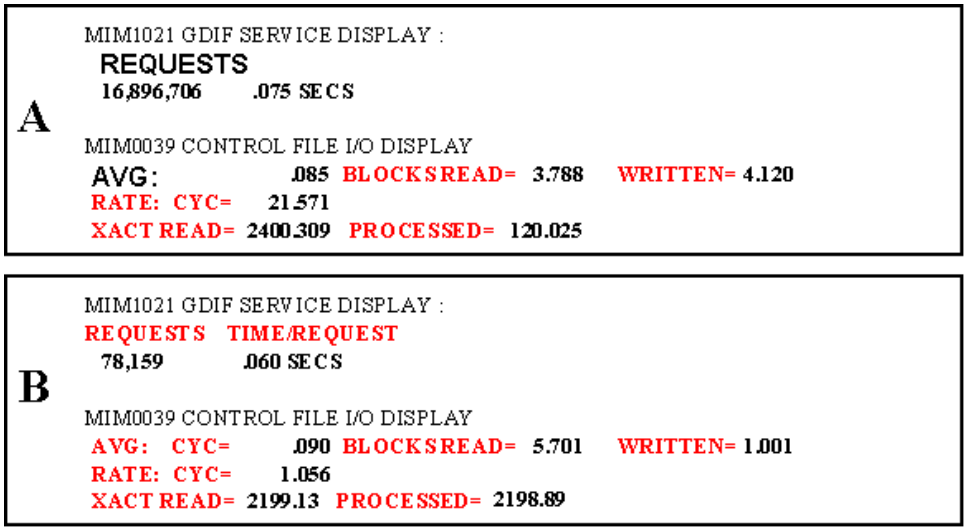
   More information:

   Control File Tuning: Parameters (see page 137)
6. Issue the commands DISPLAY IO=RESET and DISPLAY SERVICE=RESET to CA MIM on every system. This re-initializes the CA MIM performance statistics and lets CA MIM begin accumulating statistics reflecting the effects of your altered CA MIM tuning parameters.
7. Allow the new statistics to accumulate for 24 hours.

8. Issue the DISPLAY IO and DISPLAY SERVICE commands on all systems. Print the statistics. Then loop back to Step 5.
9. Repeat Steps 5 - 9 until you have adjusted the CA MIM tuning parameters such that the DISPLAY IO and DISPLAY SERVICE data contains acceptable values for every system.
10. Once you have achieved the desired results, you should make a permanent change to the MIMCMNDS member of the CA MIM parmlib. Using IFSYS/ENDIF statements you can ensure the desired SETOPTION MODE, CYCLE, AND INTERVAL values are defined for each systems.

Control File Tuning: Example

This section provides a control file tuning example, using a two-system MIMplex. Both SYSA and SYSB are running MODE=DEMAND, CYCLE=1, INTERVAL=1. The MIMplex is using the DASDONLY communication method with the control file on a shared 3390 DASD device. The key fields from the DISPLAY SERVICE and DISPLAY IO commands are shown in the following illustration:



The preceding data can be analyzed as follows: SYSA appears to be a workload-intense production system, while SYSB seems to be a test system as it has very little enqueue activity. CA MIM is performing control file cycles at a rate of roughly 22 times per second on SYSA, and only one time per second on SYSB. Remember that when in DEMAND mode, control file cycles are driven primarily by 1) CA MII enqueue workload or 2) a timer expiration. Because the number of enqueue requests on SYSA is high, it is a good assumption that CA MIM control file cycles are being driven by the enqueue workload. In addition, because the number of enqueue requests on SYSB is low, it is fair to say CA MIM control file cycles are being driven only by the timer expiration.

This MIMplex appears to be poorly tuned because of the following facts:

- The number of blocks read and written are more than the acceptable value of 1.1 for COMMUNICATION=DASDONLY.
- The average control file cycle times are longer than the acceptable value of 80ms (for 3390 DASD control file).
- The transactions read to transactions processed ratio is more than the acceptable 10:1.
- The enqueue time per request is above the highest acceptable value of 70ms. This is the most important factor.

In short, the disproportionate control file cycle rates by the CA MIM address spaces are degrading the CA MII enqueue service times.

This analysis shows that CA MIM on SYSB needs to be performing control file cycles more often, to keep pace with the SYSA control file cycle rate. The correct approach is to issue SETOPTION INTERVAL=.1 on SYSB as this forces CA MIM on SYSB to perform control file cycles at a rate of 10 times per second, instead of 1 time per second. Then issue DISPLAY IO=RESET and DISPLAY SERVICE=RESET on both SYSA and SYSB. After 24 hours, issue DISPLAY IO and DISPLAY SERVICE commands again on both systems and analyze the data.

You see that the average control file cycle times have gone down, that blocks read and written have gone down, that the transaction read to transaction processed ratio has improved, and that the enqueue time per request have improved. If further adjustments are warranted, then you might further reduce the INTERVAL on SYSB to .08 (13 accesses per second), reset the DISPLAY commands, and analyze the statistics again after 24 hours.

Once you have found the desired setting, make a permanent update to the MIMCMNDS member of parmlib so that CA MIM on SYSB will have those values in effect at startup. You can add the following statements to the MIMCMNDS member:

```
IFSYS SYSB
   SETOPTION INTERVAL=.08
ENDIF
```

In summary, the control file analysis phase is to be performed on all systems in the MIMplex. Based on this analysis, you typically alter the control fine cycle rates on only a small subset of the systems in the MIMplex. Once you have dynamically altered the control file cycle characteristics on a few systems, and are satisfied with the results, you need to update the CA MIM parmlib to have those values in effect at CA MIM start up.

How You Page-fix Cell Pooled Control Blocks

By default, the CA MIM cell pooled control blocks are pageable. This implies that paging activity for these virtual storage areas is based on reference patterns as well as the overall availability of real storage frames in the system. The CA MII component makes extensive use of the cell pool for its control blocks that describe ENQ resource names and ENQ resource ownership. Any extreme paging activity can severely impact ENQ response time on the local system or throughout the entire MIIplex.

Typically, the CA MIM working set size remains fixed in real storage and there is no need to page fix its cell pool resident control blocks. However, in times of critical pageable storage shortages, frames can be stolen from the working set pages. If a given system is configured as such that it is real storage constrained, then you may want to prevent real frames from being taken from the CA MII address space. This ensures that ENQ processing remains optimized throughout the MIIplex, even during periods of real storage constraint on certain systems.

To indicate that CA MIM should page fix cell pooled areas unconditionally when actively in use, specify the following MIMINIT keyword:

```
MIMINIT PAGEFIX=YES
```

To indicate that CA MIM should conditionally page fix actively used cell pooled areas as long as the system is not experiencing a real frame shortage, specify the following MIMINIT keyword:

```
MININIT PAGEFIX=COND
```

How to Offload CA MIM Work to zIIP Engines

For more information, see the scenario: *How to Offload CA MIM Work to zIIP Engines* on the CA MIM bookshelf.

Performance Considerations for Each Component

You have the option of activating all three CA MIM components (CA MIA, CA MIC, and CA MII) in a single CA MIM address space (CA MIA/CA MIC/CA MII), in two CA MIM address spaces (CA MIA/CA MIC, and CA MII), or in three CA MIM address spaces (CA MIA, CA MIC, CA MII). There are a number of other possible combinations as well.

From a performance standpoint, it is best to have CA MII running in an address space separated from CA MIA and CA MIC. Because enqueue service times are critical to most data centers, eliminating CA MIA and CA MIC activity in the CA MII task will improve CA MII enqueue service times. It is therefore a general recommendation that data centers requiring optimum enqueue service rates run CA MII separately from CA MIA and CA MIC.

Another benefit of running CA MIM components in separate address spaces is that problems affecting one component will not affect the others. For example, if you run all components in a single CA MIM address space, a problem involving the tape resource management may adversely affect DASD resource management or console resource management. Running CA MIM components in separate address spaces will limit the degree of impact component-specific problems may have on your data-sharing environment.

Each CA MIM component has unique performance parameters and tuning suggestions to optimize component-specific processes:

Component	Where to Look for Details
CA MIA	In Performance Considerations in the chapter “Advanced Topics” in the *CA MIA Programming Guide*.
CA MIC	In Performance Considerations in the chapter “Advanced Topics” in the *CA MIC Programming Guide*.
CA MII	In Performance Considerations in the chapter “Advanced Topics” in the *CA MII Programming Guide*.

The CA MIM Logging Facility

CA MIM has two logs for collecting information:

- The Message logs, which collect messages about CA MIM activities.
- Trace logs, which collect diagnostic information about CA MIM.

Collect Data in the CA MIM Logs.

A log is a set of sequential files that collects information about CA MIM. Two types of logs are provided:

- The Message logs collect messages about commands that are issued to CA MIM and its responses to those commands. They also collect CA MIM messages that indicate potential problems or significant events.
- Trace logs collect diagnostic information about CA MIM. CA Support uses this information.

Default Logs Provided by CA MIM.

By default, CA MIM provides two types of logs:

- A message log named MIMLOG, defined as a class A SYSOUT data set. CA MIM automatically directs messages to this log unless you remove the log (through a REMOVELOG command) or you provide a different message log.
- A trace log named MIMTRC, defined as a class A SYSOUT data set. This trace log is allocated when tracing is turned on, or when an ADDLOG command is issued.

CA MIM provides you with these default logs.

- Only define these logs when using a cataloged data set or when directing to a different SYSOUT class.

Characteristics of CA MIM Logs

All CA MIM logs have the following generic characteristics:

- Each log can contain up to four cataloged data sets or a single SYSOUT data set. When more than one cataloged data set is used, they must share the same logical record length (LRECL).
- CA MIM automatically switches to a new file when the current file becomes full. However, CA MIM overwrites data when all files become full.
- Each system must have its own log files. Log files cannot be shared between different CA MIM copies running on the same or different systems.
- The log files must not be multi-volume data sets.

Additional Considerations for Logs

When setting up your logs, keep in mind the following information:

- The order in which you define log files to CA MIM determines which file CA MIM uses first.

- If a CA MIM log is allocated to a data set, allocate the log with RECFM=FB,LRECL=132,BLKSIZE=15972.

Manipulating Log Files

Multiple operator commands can be used to manipulate the various log files within CA MIM. These commands include ADDLOG, OPENLOG, SWITCHLOG, CLOSELOG, PRINTLOG, REMOVELOG, and WRITELOG. For more information, see the *CA MIM Statement and Command Reference Guid*e for commands details.

MIMLOG Considerations

- Earlier releases of CA MIM were not able to provide a MIM job log when starting with SUB=MSTR. Now the messages are queued internally until JES is available, when CA MIM is started with SUB=MSTR before JES. The default is to write the queued message to SYSOUT A. To send the MIMLOG messages to a different class or to a data set on a disk, use the ADDLOG command.
- If CA MIM is started as SUB=MSTR before JES, it obtains an STC number as soon as JES becomes available.
- If JES terminates while CA MIM is active, CA MIM automatically disconnects from JES and queue messages internally, until JES is restarted. Operations do not need to take any special action when restarting JES while CA MIM is active.

  The MIMLOG can be eliminated by placing the following command at the top of the init member:

  ```
  ADDLOG MIMLOG DSNAME=NULLFILE
  ```

  MIM log messages go to the syslog.

  - CA MIM messages with Route Code 11 are in the JESYSMSG log and the MIMLOG.
  - CA MIM messages which are issued before initialization completes, are in the JESMSGLG log.

z/VM Considerations

This section discusses z/VM considerations.

How You Share Devices Under z/VM

When z/OS runs as a guest under z/VM, the way z/VM handles reserve/release processing can affect the integrity of resources needed by z/OS. Therefore, take these factors into account when you are running z/OS as a z/VM guest:

- Make sure that host z/VM systems forward serialization requests to DASD hardware so that the DASD hardware can perform serialization.

  This process, which serializes access among several real processors that are sharing a device, is known as *real reserve/release processing*.

- Use the z/VM *virtual reserve/release processing* feature to serialize access among systems that are running as guests under the same z/VM operating system.

How You Serialize Access in z/VM Environments

The type of reserve/release processing that you need depends on whether you have several real processors or several guest systems:

- *Real reserve/release processing* serializes access among different real processors (that is, systems that are not running under the same z/VM operating system).
- *Virtual reserve/release processing* serializes access among guest systems running under the same z/VM operating system.

If you have several guests running under one z/VM operating system and you have at least two real processors in your complex, use both real and virtual reserve/release processing.

A crucial difference between real and virtual reserve/release is that real reserve/release restricts access to an entire device (or *pack*), while virtual reserve/release restricts access to a *minidisk*. A minidisk is an addressable unit of storage on a device. If you define the entire device as a single minidisk, then you create a full-pack minidisk. You can also partition a device into several minidisks.

How You Serialize Access Through Real Reserve/Release Processing

Real reserve/release is a DASD hardware feature that serializes access among real processors by dedicating a device to one processor at a time. Use real reserve/release processing if two or more real processors are sharing a device and you are running CA MIM on both of these processors.

Real reserve/release dedicates a device by letting only one path group control the device at a time. Because each real processor uses a unique path to access a device, real reserve/release can prevent integrity exposures among real processors.

On z/OS systems, a RESERVE request causes z/OS to issue a reserve channel command word (CCW) for a device. When z/OS runs under z/VM, the z/VM CP component sends the reserve CCW to the device on behalf of the z/OS guest.

If the device is not dedicated to a processor already, then the DASD hardware executes the reserve CCW, which dedicates the device to the channel from which the reserve CCW was issued. The device rejects other reserve CCWs until the controlling z/OS guest issues a release CCW to release the device.

z/VM may remove reserve or release CCWs from the channel program of a guest under certain circumstances. If reserve or release CCWs are removed, then an integrity exposure can occur when several real processors are sharing a device.

Determine Whether z/VM Is Sending Reserve CCWs to Devices

To see if z/VM is removing or sending reserve CCWs to a DASD, issue a QUERY command. To see if z/VM is sending reserve CCWs to a certain device, specify the address of that device as a parameter on the QUERY command. For example, to see if z/VM is sending reserve CCWs to device 123, issue the following Class B command:

```
Q 123
```

To see if z/VM is sending reserve CCWs to any devices, issue this command instead:

```
Q DASD
```

If SHARED or DED is specified to the right of the volume label in the resulting display, then z/VM is sending reserve CCWs to the device. Otherwise, z/VM is removing reserve CCWs, which means that integrity cannot be ensured among real processors.

Tell z/VM to Send Reserve CCWs to Devices

If z/VM is not sending reserve and release CCWs to your devices, then take additional steps to make z/VM send these CCWs to your devices. This ensures that access is serialized among real processors. To make z/VM forward reserve CCWs temporarily (that is, until the next time you perform an IPL), issue a SET SHARED command for that device. Specify the address of the device as a parameter on the command. For example, to make z/VM send reserve CCWs to device 123, issue this Class B command:

```
SET SHARED ON FOR 123
```

To make z/VM forward reserve CCWs permanently, specify SHARED=YES on the RDEVICE macro in the HCPRIO assembly file, or SHARED YES on the RDEVICE macro in the system configuration file. Repeat this step for each shared device. Your changes to the RDEVICE macro take effect the next time you perform a system generation and IPL for z/VM.

How You Serialize Access Through Virtual Reserve/Release Processing

Virtual reserve/release is a z/VM feature that serializes access among guests running under the same z/VM operating system. Use virtual reserve/release processing if you are running more than one guest under the same z/VM system and you share devices among these guests.

You must invoke virtual reserve/release in this situation because guests under the same z/VM operating system share the same real channel to a device. Real reserve/release cannot ensure integrity among these guests, because it has no effect on guests that share a real channel to a device.

Virtual reserve/release dedicates a specified minidisk to a single guest at a time. You can use virtual reserve/release to dedicate part of a device to a guest (if you defined several minidisks on that device) or to dedicate an entire device to a guest (if you defined that device as a full-pack minidisk). The implications of dedicating only part of a device are discussed in detail in the next section.

During virtual reserve/release processing, CP intercepts a reserve CCW, marks the appropriate minidisk as reserved, and then sends the reserve CCW to the device. You may need to tell z/VM systems to send reserve CCWs properly, as described in Telling z/VM to Send Reserve CCWs to Devices in this chapter. The device then performs real reserve/release processing in response to the reserve CCW, which prevents other real processors from accessing the device while it is busy. CP prevents other guests running under the same z/VM operating system from accessing that minidisk until the controlling guest issues a release CCW. CP also sends the release CCW to the device so that other processors can access the device again.

How You Invoke Virtual Reserve/Release Processing

Invoke virtual reserve/release processing for any minidisk that is being shared by two or more guest systems running under the same z/VM operating system.

To invoke virtual reserve/release processing, specify an access mode of MWV in the MDISK statement for the appropriate minidisk in your z/VM directory. For example, to invoke virtual reserve/release processing for the full-pack minidisk 123, specify this MDISK statement:

```
MDISK 123 3380 000 885 MVSRES MWV MVSREAD MVSWRIT MVSMULT
```

This statement also indicates what type of device this is (3380), the first cylinder and number of cylinders that define the boundaries of this minidisk (000 and 885), the minidisk label (MVSRES), and various passwords for accessing the device (MVSREAD, MVSWRIT, and MVSMULT).

This MDISK statement defines minidisk 123 for a single guest. Other guests need to issue LINK commands with an access mode of MW in order to use this minidisk. For example, another system would issue this command to write to minidisk 123:

```
LINK MVS 123 123 MW MVSMULT
```

How You Serialize Access to Minidisks

An integrity exposure can occur if different real processors use reserve/release processing to serialize access to the same minidisk and that minidisk is not a full-pack minidisk.

Important! We strongly recommend that you specify your control file on a full-pack minidisk.

How You Alter System Definitions While CA MIM Executes

The ALTERSYS command lets you:

- Modify an existing system definition with a status of FREED or DISABLED
- Avoid using the SYSID execution parameter override and maintain correct system definitions (DISPLAY SYSTEMS)

Issue the ALTERSYS command dynamically after CA MIM is synchronized.

This example shows you how to modify a system definition. Here we are changing SYS32 with alias 32 to SYS33 with alias 33.

Follow these steps:

1. Shutdown and FREE SYS32.
2. Issue the ALTERSYS command in the following format:

   ```
   ALTERSYS SYS32,NAME=(SYS33,33),STATUS=ENABLED
   ```

   The ALTERSYS command changes the CA MIM system name from SYS32 to SYS33 and the CA MIM system alias to 33.
3. Verify the modification was successful by issuing the DISPLAY SYSTEMS command.
4. Replace the DEFSYS entry for SYS32 with an entry for SYS33 in your shared CA MIM initialization parameter member.
5. Start SYS33.

After a system definition is altered:

- The system is marked DISABLED (unless the STATUS keyword specifies ENABLED) and cannot join a currently executing MIMplex.
- Systems are marked DISABLED after a name alteration to prevent the use of that definition until it is explicitly enabled.
- You can explicitly enable the disabled system using the ALTERSYS STATUS=ENABLE command.
- If a DISABLED system attempts to join a currently executing MIMplex, the system is forced to terminate with a U0040 ABEND.

Chapter 4: Troubleshooting

This section contains the following topics:

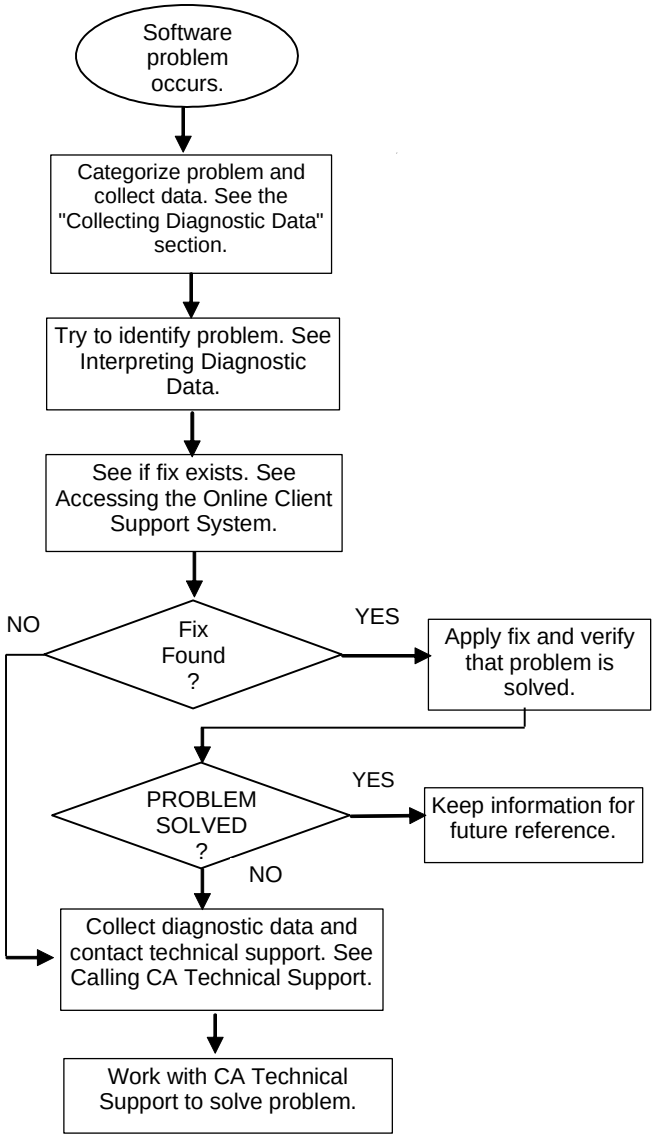
How You Resolve Problems

This section contains information about:

- Identifying and resolving problems
- Contacting CA Technical Support
- Receiving a new release of a product and ongoing maintenance
- Requesting product enhancements

Diagnostic Procedures

For a summary of the procedures you should follow if you have a problem with a CA software product, see the following flowchart. Each of these procedures is detailed on the following pages.



Diagnostic Data

The following information is helpful in diagnosing problems:

- Control statements used to activate your product
- JCL used to install or activate your product
- 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:
  - Fix numbers
  - The dates that fixes were applied
  - Names of components to which fixes were applied
- Short description of problems

How You Interpret Diagnostic Data

When you have collected the specified diagnostic data, write down your answers to the following questions:

- What was the sequence of events prior to the error condition?
- What were the circumstances when the problem occurred and what action did you take?
- Has this situation occurred before? What was different then?
- Did the problem occur after a particular PTF was applied or after a new release of the software was installed?
- Have you recently installed a new release of the operating system?
- 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.

How You Respond to Control File Access Delays

This section explains how to respond to various CA MIM messages that pertain to control file access delay.

Message MIM0061W

Message MIM0061W tells you that an external system that was active at startup time now appears to be inactive.

This warning message is issued by CA MIM when a system stops updating its time stamp. It is also issued at startup when a system that CA MIM is expecting does not join the complex. The system must be freed (using the FREE command) at startup to allow CA MIM to properly synchronize and clear the MIM0061W message.

To resolve problems indicated by this message

1. Wait to see if the system updates its timestamp. CA MIM removes highlighting for the MIM0061W message if the system updates its time stamp.

   Is highlighting removed for the MIM0061W message?

   Yes

   The system is active.

   No

   A system may appear inactive if processing on that system is slow, another system is dominating the control file, or for some other reason. When a system displays MIM0061W messages often and that system is not inactive, consider increasing the value for the MARGIN parameter on the SETOPTION command. Through this parameter, you can tell CA MIM to wait longer before issuing MIM0061W messages. Also, you may need to tune CA MIM.

   Go to Step 2.

2. See if an active system has a RESERVE request for the control file. Look for MIM0100A messages on all other systems.

   Are other systems displaying MIM0100A messages?

   Yes

   A local task may have issued a long RESERVE request that prevents other systems from accessing the control file. Go to a system that displays the MIM0100A message and follow the procedures for responding to MIM0100A messages.

   No

   Go to Step 3.

3. See if CA MIM is being dispatched on the inactive system. To do this, issue the z/OS D A,MIMGR command several times from the apparently inactive system. Note that MIMGR is the name of the CA MIM started task.

   What response do you receive?

   None

   z/OS may be inactive. Determine whether z/OS is inactive, and take corrective action.

   Message IEE115I

   Look at CPU use on each IEE115I message. Does CPU use change?

   - **No**-CA MIM is not being dispatched. See which task on that system is preventing other tasks from being dispatched and correct the error.
   - **Yes**-Go to Step 4.

4. See if the inactive system failed. To do this, issue a DISPLAY SYSTEMS command from the apparently inactive system.

   What response do you receive?

   None

   A hardware or software failure occurred on that system. You may need to complete a system reset (to clear the channel to the device) and then complete an IPL on that system.

   If you are sure the system will not recover immediately, and the system is not currently using any resources, then free resources and devices held on that system. To do this, issue a FREE command on another system. For example, to free resources held on system A1, issue a FREE A1 command from another system.

   Important! Integrity exposures can occur if you issue the FREE command for a system that is still actively using resources managed by CA MIM.

   Message IEE305I

   CA MIM failed on that system. Restart it on that system as soon as possible.

   Other response

   Go to Step 5.

5. See if the inactive system is migrating to a new control file. To do this, issue a DISPLAY SYSTEMS command from any system. Then, find the line for the inactive system on the MIM0108I message you receive.

   What value appears in the STATUS field?

   AWAKENING

   The apparently inactive system migrated to a new control file. Other systems will also migrate to that control file. Do not take action.

   MIGRATING

   Migration is underway. Wait 60 seconds to see if it completes. If migration does not complete in 60 seconds, then respond to the outstanding migration message on that system.

   Other value

   Contact CA Technical Support.

Guidelines for Researching Message MIM0061W Problems

Use the following guidelines when researching inactive system problems:

- Certain CA MIM messages that give you status information (such as the MIM0108I message) also indicate when a system may be inactive. If you suspect that a system is inactive, then look for MIM0061W, MIM0062W, or MIM0063W messages, and follow the procedures described in the next section for the appropriate responses to those messages.
- If you quiesced a system, then other systems display MIM0061W messages. Eventually, these systems display MIM0063W messages too. This is a normal part of quiesce processing.
- If CA MIM displays MIM0100A messages on other systems, then follow the procedures described in Responding to Message MIM0100A in this chapter.

Note: The CA MIM FREE command is useful for releasing devices and resources held by an inactive system. However, do not automatically issue the FREE command if a system becomes inactive. Follow the procedures for responding to the message you receive.

WARNING! We do not recommend the use of an automated operations package to respond to CA MIM messages.

Message MIM0062W

This message is issued only during migration processing. It indicates that CA MIM cannot continue processing because a system appears to be inactive. CA MIM cannot resume processing and migration does not complete until you respond to this message.

CA MIM issues MIM0062W messages during migration if CA MIM on an external system has not joined the migration to the new DASD or virtual control file. Because processing is suspended while the MIM0062W message is outstanding, you must respond to MIM0062W messages promptly.

Note: The MARGIN parameter on the SETOPTION command determines how many seconds CA MIM waits before issuing this message.

If you receive MIM0062W messages, then start with Step 2 of the procedure for responding to MIM0061W messages. This procedure is shown in the table in the previous section. Because CA MIM suspends all processing while MIM0062W messages are outstanding, do not wait to see if the apparently inactive system updates its time stamp (as described in Step 1 of that procedure).

Message MIM0063W

CA MIM issues this message after issuing the MIM0061W message several times.

CA MIM issues MIM0063W messages if a control file becomes overcrowded with transactions destined for an apparently inactive system. It then issues MIM0063Ws only after issuing MIM0061W messages several times to notify you that an external system may be inactive.

CA MIM deletes all control file records destined for this system and stops propagating information to this system. When the system becomes active again, all systems synchronize so that they exchange the most current information about the activities of each other.

If you receive the MIM0063W message on a system, then follow the procedures outlined in the table that provides responses to MIM0061W messages.

Message MIM0100A

CA MIM issues message MIM0100A if it cannot access a DASD control file that it needs to communicate transactions or to store checkpoint information.

Note: The LOCKOUT parameter on the SETOPTION command determines how many seconds CA MIM waits before issuing the MIM0100A message.

The ID for the inaccessible control file, as well as the unit and volume serial number for the device where this control file resides, are shown in this message. The reply ID for this WTOR message is shown at the very beginning of the message.

Take the following steps to resolve problems indicated by this message:

1. Wait to see if the local system can access the control file. CA MIM removes highlighting from the MIM0100A message if the system accesses the control file.

   Is the highlighting removed for the MIM0100A messages?

   Yes

   A temporary lockout occurred. That is, excessive activity or RESERVE requests temporarily prevented CA MIM from accessing its control file. You do not need to take action.

   If temporary lockouts occur often, then consider moving the control file to another volume. Also, you can increase the value for the LOCKOUT parameter on the SETOPTION command, so CA MIM waits longer before issuing MIM0100A messages.

   No

   Go to Step 2.

2. See if an external system has a long RESERVE request for the control file. To do this, see if CA MIM is issuing MIM0061W messages on any other system.

   Are other systems displaying MIM0061W messages?

   Yes

   A task on that external system may have issued a long RESERVE request for the volume where the control file resides. This prevents other systems from accessing the control file.

   To see if this happened, issue a DISPLAY RESERVES command from the system that displays the MIM0061W message. Note that this command is available only when you are running GDIF. In response, CA MIM issues message MIM1017, which lists the name of each job on the system that has an outstanding RESERVE request. Use this information to see whether any job controls the device on which the control file resides. The UCB address of the device is shown in MIM0100A messages.

   When a job has a RESERVE request for this device, you can wait for the job to issue a DEQ request. However, if the job does not issue a DEQ request soon and system performance continues to deteriorate, then cancel the job.

   If no job has a RESERVE request for this device, then go to STEP 3.

   No

   Skip STEP 3 and go directly to STEP 4.

3. See if other systems are dominating the control file. To do this, issue the DISPLAY IO=RESET command on each system that displays the MIM0061W message. Ignore the MIM0039 messages you receive. Wait ten seconds, and then issue a DISPLAY IO command on each system.

   Look at the statistics beneath the LAST RESET field on the last MIM0031 message. For each external system, multiply the number of cycles per second (represented by the CYC field on the RATE line) by the duration of an average cycle (represented by the CYC field on the AVG line). Then, total the values.

   What sum do you get?

   Close to One

   Other systems are dominating the control file. Perform one of these tasks:

   When you are using virtual control files, increase the value for the VCFMINDORM parameter on the SETOPTION command by .5 on each of those other systems.

   When you are not using virtual control files, issue a SETOPTION MODE=GROUPS command on each of those other systems. Also, change the value for the INTERVAL parameter on the SETOPTION command to .2 and change the value for the CYCLES parameter to 5.

   Not Close to One

   Go to Step 4.

4. See if hardware or software failed on an external system. To do this, issue a z/OS D A,L command on each system that is not displaying a MIM0100A message.

   What response do you get?

   None

   A hardware or software failure occurred on that system while the system had an outstanding RESERVE request for the device where the control file resides. You may need to perform a system reset (to clear the channel to the device) and then complete an IPL on that system.

   If you are sure that the system will not recover immediately, then free resources held on that system by issuing a FREE command from any other system. For example, to free resources held on system A1, issue a FREE A1 command from any other system.

   Message IEE104I

   Go to Step 5.

5. See if CA MIM is being dispatched on the inactive system. To do this, issue the z/OS D A,MIMGR command several times from the apparently inactive system (note that MIMGR is the name of the CA MIM started task).

   Does CPU use change from message to message?

   No

   CA MIM is not being dispatched. Do the following:

   a. Determine which task on that system is preventing other tasks from being dispatched.

   b. Correct the error.

   Yes

   Go to Step 6.

6. See if the device on which the control file resides failed. Do this in one of the following ways:

   - Look for MIM0118E or MIM0008E messages on other systems. These messages tell you an I/O error occurred when CA MIM tried to access its control file.
   - Use the z/OS D U command to display status information for this device. Append the address of the device address to this command. For example, issue the command D U,,,3C3,1 to display status information for device 3C3. MIM0100A messages show the UCB address for this device. You can issue this command from any system.
   - Look for z/OS IOS000I messages about this device. This message tells you a hardware failure occurred.

   Did the device fail?

   Yes

   Initiate migration to a DASD control file that resides on a different device. To do this, reply ABANDON to the outstanding MIM0100A message on all systems.

   All systems join the migration process automatically, *only* if all MIM0100A messages on all systems are cleared.

   No

   Go to Step 7.

7. See if a channel or control unit failed. To do this, go to a system that displays the MIM0100A message. From there, use a z/OS command to obtain status information about the channels and control units connecting the local system to the device where the control file resides. Use a z/OS display command appropriate for your version of z/OS.

   Do any channels or control units have an unusual status (such as BOXED or SUSPENDED)?

   Yes

   The channel or control unit failed. Initiate migration to a DASD control file that resides on another device by issuing a MIGRATE command from any system. For example, to migrate from file 01 to file 02, issue a MIGRATE CF=02 command.

   All systems join the migration process automatically.

   No

   Contact CA Technical Support.

Message MIM0200W

CA MIM issues message MIM0200W if it cannot access its virtual control file. The VCFMAXDELAY parameter on the SETOPTION command determines how many seconds CA MIM waits before issuing the MIM0200W message.

To resolve problems indicated by this message

1. See if the master system failed. To do this, issue the DISPLAY SYSTEMS command from your master system. The ID for the master system is shown in the MIM0200W message.

   What response do you receive?

   Message IEE0305I

   CA MIM failed on the master system.

   Important! Restart it on that system as soon as possible since integrity exposures can occur while CA MIM is inactive.

   Message MIM0108I

   The master system did not fail; it is still active. Go to Step 2.

None

A hardware or software failure occurred on your master system. Determine whether z/OS failed or a task on your master system is preventing other tasks from being dispatched. Then, correct the error.

If you are sure that the system will not be recovered immediately, then free resources held on that system. To do this, issue a FREE command on any other system. For example, to free resources held on system A1, issue a FREE A1 command from another system. CA MIM then initiates migration to a backup DASD control file or a new master on all other systems. You do not need to take any further action unless migration cannot complete.

2. See if a CTC device failed. To do this, look at the STATUS field on the MIM0108I message displayed on the master system.

   Does the value RESERVE appear in the STATUS field for any system?

   No

   A CTC device failed. As a result, migration did not begin as it should have. Initiate migration to a backup DASD control file, if available, by issuing a MIGRATE command that identifies the current master system and a DASD control file. For example, to migrate from the master system SYS1 to DASD control file 03, issue a MIGRATE CF=03 command. Other systems join the migration process automatically.

   If no DASD control file is available, then initiate migration to a new master system.

   Also, you may need to reset the CTC device. To see if this is necessary, issue a DISPLAY CTCPATH from any system. If the value ERROR appears on the MIM0176 message you receive, then reset that device by issuing a CTC RESET command.

   Yes

   Note the system ID of the system for which the value RESERVE appears, and go to Step 3.

3. See if the system that has the virtual reserve failed. To do this, issue a DISPLAY SYSTEMS command on the system for which the value RESERVE appears in the MIM0108I message.

   What response do you receive?

   Message IEE0305I

   CA MIM failed on that system while the system held a virtual reserve for the control file.

   Important! Restart it on that system as soon as possible, since integrity exposures can occur while CA MIM is inactive.

Message MIM0118E

An unexpected error occurred. If migration does not start automatically, then initiate migration to a backup DASD control file, if available, or migrate to a new master system. For example, to migrate from the Virtual Control File managed by system SYS1 to DASD control file 02, issue a MIGRATE CF=02 from any system. Then contact CA Technical Support.

None

A hardware or software failure occurred on that system while it held a virtual reserve for the control file. Determine whether z/OS failed or a task on this system is preventing other tasks from being dispatched. Then, correct the error.

Follow this sequence to maintain system integrity:

a. See if CA MIM is being dispatched on the inactive system. To do this, issue the following z/OS command several times from the apparently inactive system:

```
D A,MIMGR
```

Note that MIMGR is the name of the CA MIM started task. See if CPU use changes.

b. If the CPU use does not change, then CA MIM is not being dispatched. Determine which task on that system is preventing other tasks from being dispatched and correct the error.

c. If you are sure the system will not recover immediately, then free resources held on that system by issuing a FREE command on the master system.

How You Activate Tracing

Use the trace facilities in CA MIM only for problem diagnosis under the direction of CA Support. Tracing can add considerable overhead to CA MIM operations.

Control trace activity using the SETOPTION TRACE=ON | OFF command. Issue this command from a console at any time or included it in your MIMCMNDS or MIMSYNCH members. Use the following commands to control trace activity:

- SETOPTION MIM TRACE
- SETOPTION MIM TRACE
- SETOPTION MIM SETTRACE
- SETOPTION MIM SETPRINT
- SETOPTION MIM RESETTRACE
- SETOPTION MIM RESETPRINT

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

How You Send Trace Data to a SYSOUT Class

To tell CA MIM to write trace records to a SYSOUT class, issue the following command:

```
ADDLOG MIMTRC SYSOUT=A
```

(Optional) You can specify any one or more of the following parameters when allocating a SYSOUT data set:

COPIES

Prints copies.

DEST

Sends the data set to the specified print device.

FCB

Determines what forms control the buffer image is used during printing.

FORMS

Indicates what form is used when this data set is printed.

HOLD

Determines whether this SYSOUT data set is held until you release it. Specify HOLD=YES to hold the data set or HOLD=NO to have it print automatically when it is de-allocated.

UCS

Determines what universal character set, print train, or character arrangement table is used for this data set.

This example shows how to allocate aCA MIM trace data set and print five copies of this data set:

```
ALLOCATE DDNAME=MIMTRCTT SYSOUT=A COPIES=5
ADDLOG MIMTRC DDNAME=MIMTRCTT
```

How You Send Trace Data to an Output File

Trace data that is collected in the internal trace table can be sent to a CA MIM trace data set rather than to a SYSOUT class. To tell CA MIM to write trace records to a data set, preallocate a data set with the following attributes:

- RECFM=VB
- LRECL=132
- BLKSIZE=15972

Issue this command to start writing to the file:

```
ADDLOG MIMTRC DSNAME=(MIM.TRACE)
```

Alternatively, use the ADDLOG to allocate multiple data sets for your tracing output. A maximum of four files can be defined for tracing. The tracing continues to the next file once the first is full.

Note: Once wrapping has gone through all available trace files, it starts again from the top and writes over any previous data.

```
ADDLOG MIMTRC DSNAME=(MIM.TRACE1,MIM.TRACE2,MIM.TRACE3)
```

How You Obtain Dumps

You use the SYSDUMP command to obtain dumps.

Important! Use this command only under direction of CA Technical Support.

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

Product Releases and Maintenance

Customers are requested to operate only under currently supported releases of the product.

Customers with current maintenance agreements also receive ongoing product maintenance. When a new release of the product is available, a notice is sent to all current customers.

Note: For more information, see the *Installation Guide*.

How You 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:

- Contact your account manager or a Technical Support representative, who will initiate a Demand Analysis Request (DAR) for you.
- Enter your request through CA Support Online, the CA web-based, interactive support system. You can access Support Online at http://ca.com/support.

Chapter 5: User Exits

This section contains the following topics:

How You Manage User Exits

CA MIM provides exit routines that you can use to customize its processing. The following list describes each routine:

MIMATHXT

This routine customizes CA MIM command authorization processing.

MIMCMDXT

This routine prevents certain CA MIM commands from being issued on the local system or across systems. This routine can affect all components and facilities.

MIMINIXT

This routine acquires and initializes a global communication area that other CA MIM exit routines can share. This routine can affect all components and facilities.

Load, Enable, or Disable Exit Routines

All exit routines except for the MIMINIXT exit can be loaded, enabled, or disabled by using the SETOPTION EXIT command while CA MIM is running. You can even load new versions of the exit routines while CA MIM runs uninterrupted.

You can issue appropriate SETOPTION commands to manage CA MIM exit routines. The EXIT parameter lets you identify the exit routine you want to control. Additional parameters let you designate whether routines should remain active under normal and abend conditions, and let you provide a means to trace data and recover from abends.

For example, to control the MIMCMDXT exit routine, you could issue this SETOPTION command:

```
SETOPTION EXIT=(MIMCMDXT)
```

View Exit Routine Information

To see information about exit routines you are using, issue a DISPLAY EXIT command. CA MIM displays each logical exit routine name with its load module, address, status, and values set for protection and dump generation. The next screen shows the display for exit routine information:

```
F MIMGR,DISPLAY EXIT

MIM0264 MIM EXIT DISPLAY
EXIT          MODULE    ADDRESS       STATUS    PROT  DUMP  DISA
GCMCMDXT      USR       00006910      ACTIVE    YES   YES   NO
GCMCMDXT      USREX     00006920      INACT     YES   YES   NO
GCMSRCXT      USREXIT   00006930      INACT     NO    YES   NO
TPCRECXT      USREXIT   00006940      ACTIVE    NO    NO    YES
```

You can use the DISPLAY EXIT command to check on the status of an exit routine, except for MIMINIXT. Use the DISPLAY INIT command to find out whether the MIMINIT INITEXIT statement has been set.

Common Exit Interface (MIMINIXT)

CA MIM provides a common exit interface that you can use to dynamically manage any CA MIM exit routine. The common exit interface provides these advantages:

- You can dynamically load, activate, and de-activate exit routines.
- You can request an optional automatic recovery environment in the definition of the exit routine.
- CA MIM supports residency mode 31 (RMODE=31) fully. If you link an exit routine with RMODE=31, then the routine is loaded above the 16 MB line and is *always* invoked in addressing mode 31 (AMODE=31).

When you load or reload a routine and a prior copy of that routine exists, the old version of the routine is released from storage when it is no longer in use.

Note: You must modify routines written for previous releases of CA MIM to function with the new format and parameters.

The MIMINIXT exit routine is handled separately because it must run its initialization code before any other exit routine runs. You can use the MIMINIT INITEXIT statement in the initialization member to load and activate the MIMINIXT exit routine.

The MIMINIXT exit routine runs only when CA MIM begins operations, so it cannot be dynamically reloaded or otherwise changed. To rerun MIMINIXT or to run a new version, it is necessary to restart CA MIM.

Communication Words in Exit Routines

CA MIM exit routines use words that serve as communication pointers during processing. A *communication word* is a CA-defined word of memory that CA MIM sets at initialization and then passes unchanged for all subsequent calls of an exit routine. The routines use two types of communication words:

- A *routine-specific communication word* is unique to a particular exit routine. The routine-specific communication word may be used for any purpose, but most often it is used as a pointer to an area of storage. The exit routine is permitted to set the value of this word during an initialization call, and the value will be passed to the exit routine on each subsequent call.
- When an exit routine is reloaded, the prior value of its communication word is made available for the re-initialization call. Doing this enables the exit routine to either reuse the previous data area or release it from storage. The exit routine is responsible for maintaining any previously allocated storage.
- The value of the routine-specific communication word is retained under some conditions even if CA MIM is stopped and restarted. Therefore, it is possible for this word to be set to a nonzero value when the initialization call occurs during startup. It is the responsibility of the exit routine initialization code to determine whether any nonzero value was supplied, and if it was, to determine whether the value is still usable. The exit routine common parameter list provides a flag bit to indicate whether the exit initialization call is being made after a restart of CA MIM.
- For example, if the routine-specific communication word is set to point to storage obtained in CSA, then the pointer remains valid even if CA MIM is restarted. In this case, the initialization code could leave the word set as it is, and the exit routine would reuse the same area.
- However, if the word points to private storage that was obtained from the CA MIM address space, then the value during a second initialization pass no longer points to a valid area of storage, and a new area has to be obtained. This occurs because all storage obtained from the CA MIM address is freed when the address space is stopped.
- A *global communication word* is a common value that is set by the MIMINIXT exit routine and passed to all routines.
- The MIMINIXT exit routine is called once when CA MIM initializes, before any other exit routine is initialized. You can use the MIMINIXT routine to acquire and initialize a global exit communication area. The MIMINIXT exit routine is described in Coding the MIMINIXT Exit in this chapter.

Multiple Exit Routines

CA MII provides a sample exit driver for invoking multiple exit routines from any of the CA MII exit points. The XTDVR member of the CAI.CBTDSAMP data is a sample exit that can be used to assist in coding an exit driver routine that can call multiple exit routines at any one of these CA MII exit points:

- EDIABNXT
- EDIATRXT
- EDIOPTXT
- GDIXMPXT
- XCMCMDXT
- XCMCNFXT
- XCMNAVXT
- XCMMSGXT
- XCMPGMXT

Note: See the chapter “User Exits” in the *CA MII Programming Guide*.

Common Parameter List for All Exit Routines

All exit routines are invoked through a standard parameter list pointed to by register 1. The values are:

+0

Fullword containing the address of the parameter list of the exit routine. This is not used for initialization calls.

+4

Fullword containing the communication word of the exit routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call.

If CA MIM is restarted, then the value set during the first initialization call is passed during the second initialization, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the routine. The high order bit of the first byte is set on X'80' for the initialization call. If the initialization call is being made after the restart of CA MIM without an intervening IPL, then the X'40' bit is set in the first byte as well.

+C

Fullword containing the global communication word set by the exit routine.

The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this standard parameter list.

Note: For more information about a specific exit routine, see the appropriate *Programming Guide*.

You can use the optional MIMINIXT exit routine to acquire and initialize a global exit communication area that can be shared among exit routines for any CA MIM facility. You can use this communication area to allow routines to share site-specific data, such as a predefined user table.

CA MIM calls the MIMINIXT exit routine once per startup and before the initialization call for any other routine. On entry, +C in the common exit parameter list is either zero or the previous value from when you last shut down CA MIM. The flag byte at +8 is always set to X'80'.

The address of the global data area that is acquired and initialized must be stored at +C in the common exit parameter list before the routine returns to CA MIM.

Sample Exit Routines

Samples of the CA MII exit routines are contained in the CAI.CBTDSAMP data set. You can use these sample routines, or you can create your own routines. The CAI.CBTDMAC data set also contains mapping macros for the exit-specific parameter lists.

Exit Routine Programming Considerations

Use the guidelines in this section when coding your exit routines using the samples.

MIMATHXT Exit

You can use the optional MIMATHXT exit routine to augment the command authorization processing provided by CA MIM and the operating system security subsystem. The MIMATHXT is called only when the MIMINIT SAFCMDAUTH statement has activated CA MIM command authorization.

CA MIM calls the MIMATHXT exit routine after all parameters have been set for command validation by the security system, but before the actual system authorization facility (SAF) call is made.

You can use this exit routine to determine whether CA MIM should:

- Unconditionally permit command execution
- Unconditionally reject commands
- Conditionally permit command execution based on the security system decision

You can alter the command entity name or the UTOKEN that CA MIM passes to the security system for command authorization. You cannot alter the command text from this exit routine.

Follow these rules when coding the MIMATHXT routine:

- Preserve all registers except registers 0, 1, and 15.
- Review all code you define in this routine to prevent abends from occurring. CA MIM may try to recover from abends but can terminate if it experiences repeated abends.
- Do not use the z/OS OPEN, CLOSE, LOAD, LINK, BLDL, or TSO facilities, or any other z/OS facility that implicitly or explicitly calls the WAIT supervisor.
- Do not use the STIMER or TTIMER macros.
- Make sure that you interpret the command name properly when intercepting a command with this routine.
- Cross-system commands are stripped of their command characters by the time the routine intercepts them. Also, command aliases are converted to the new command name (and parameters). Therefore, base your comparisons on the true command rather than on any aliases you have defined for that command.
- Use the EXIT parameter on the SETOPTION command to identify the appropriate load module for the MIMATHXT routine. For example, specify SETOPTION EXIT=MIMATHXT when the module name is MIMATHXT. If you use a different module name, then specify EXIT=MIMATHXT and specify the module name on the LOAD parameter. For example, to use the load module UEXIT1 for the MIMATHXT routine, specify the command SETOPTION EXIT=(MIMATHXT LOAD=UEXIT1).

Assemble and link-edit the completed routine as an authorized program with the module name you specify on the SETOPTION command. Link-edit the routine into the authorized load library for CA MIM. The sample member named ASMEXIT in the CAI.CBTDJCL data set contains JCL that you can use to assemble and link-edit this routine.

Note: For more information, see the *Statement and Command Reference Guide*.

Entry Environment:

Supervisor state, problem program key, addressing mode (AMODE) 31.
You must preserve this entry environment upon entry and restore it upon return.

Common Parameter List

R1

This register contains the address of the parameter list. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this standard parameter list. It has the following format:

+0

Fullword containing the address of the parameter list of the MIMATHXT exit routine. This is not used for initialization calls.

+4

Fullword containing the communication word of the MIMATHXT routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during the second initialization, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the routine. The high order bit of the first byte is set on X'80' for the initialization call.

+C

Fullword containing the global communication word set by the MIMINIXT exit routine.

R13

This register contains the address of one standard save area.

R14

This register contains the return address for CA MIM.

R15

This register contains the entry point address for the MIMATHXT routine.

All other registers are undefined.

MIMATHXT Exit-specific Parameter List

+0

Fullword containing the address of the command text buffer. You cannot alter the command text.

+4

Fullword containing the true length of the command text buffer. You cannot alter the command text length.

+8

Fullword containing the address of the command verb. You cannot alter the command verb.

+C

Fullword containing the length of the command verb. You cannot alter the command verb length.

+10

Fullword containing the address of a two-byte field containing the MCS command issuer authority in z/OS, or the address of a four-byte field containing the CP user class in z/VM.

+14

Fullword containing the address of the converted UTOKEN that is passed to the operating system security subsystem when performing command validation. This area is mapped by the ICHUTOKN mapping macro. You can change the content of the UTOKEN area; however, you cannot change the UTOKEN length.

+18

Fullword containing the address of the 39-byte command entity name that is passed to the operating system security subsystem when performing command validation. You can change the content of the entity name area, but the name cannot exceed 39 bytes.

+1C

Fullword containing the command issuer source as an eight-byte field, and is interpreted depending upon the setting of the flag byte at offset +24.

+24

Flag byte that designates the content of offset +1C:

X'80'

The source is the two-byte TJID of a TSO user.

X'40'

The source is the four-byte console ID.

X'20'

The source is the four-byte MCS extended console ID.

X'10'

The source is the internally issued command.

X'08'

The source is the eight-byte CMS user ID.

+25

Flag byte that designates the CA MIM command designation:

X'80'

Indicates that the command is an authorized CA MIM command.

Return Code, Register 15:

0

CA MIM should continue normally by calling the security subsystem and conditionally executing the command based on the decision made by the security subsystem.

4

CA MIM bypasses calling the security subsystem and unconditionally executes the command.

8

CA MIM should bypass calling the security subsystem and unconditionally reject the command. Message MIM0420 is returned as a command response indicating that the MIMATHXT rejected the command.

Note: You must restore all registers (except 0, 1, and 15) upon return.

MIMCMDXT Exit

You can use the optional MIMCMDXT exit routine to prevent certain CA MIM commands from being issued on the local system or across systems. You can also increase the command authority level associated with a console or TSO user ID so that operators or TSO users can issue CA MIM commands that they could not issue otherwise.

You cannot use this routine to prevent z/OS or JES commands from being issued on a local system or to increase the command authority level needed to issue z/OS or JES commands. You can change the length and contents of the text of a command with the MIMCMDXT routine.

CA MIM invokes this routine whenever it issues a command on the local system and whenever a cross-system command is issued from the local system.

Follow these rules when coding the MIMCMDXT routine:

- Preserve all registers except registers 0, 1, and 15.
- Review all code you define in this routine to prevent abends from occurring. CA MIM may try to recover from abends, but can terminate if it experiences repeated abends.

- Do not use the z/OS OPEN, CLOSE, LOAD, LINK, or BLDL macros, the TSO facilities, or any other z/OS facility that implicitly or explicitly calls the WAIT supervisor.
- Do not use the STIMER or TTIMER macros.
- Make sure that the routine uses the appropriate command authority level for each situation.
- If you use the CA MIC LINK command to change the authority level for a TSO user or console, then CA MIM uses the authority level assigned by the LINK command for cross-system commands. Otherwise, TSO users are assigned informational command authority (X'0000'), and the command authority level for an MCS console is established in the control block for that console.
- TSO users and consoles need informational command authority to issue CA MIM display-type commands. They need system control command authority (that is, X'80') to issue other CA MIM commands.
- Make sure that you interpret the command name properly when intercepting a command with this routine.
- Cross-system commands are stripped of their command characters by the time the routine intercepts them. Also, command aliases are converted to the new command name (and parameters). Therefore, base your comparisons on the true command rather than on any aliases you have defined for that command.
- If you want to alter the text of a command, then do so at offset +4 and *not* at offset +0.
- Provide a save area of at least 72 fullwords if you want to issue messages from this routine. Mandatory code for this subroutine is shown in the example included in Issuing Messages from MIMCMDXT in this chapter.

  Also, do not change registers 10 or 11 if you use this subroutine.
- Use the EXIT parameter on the SETOPTION command to identify the appropriate load module for the MIMCMDXT routine. For example, specify SETOPTION EXIT=MIMCMDXT when the module name is MIMCMDXT. If you use a different module name, then specify EXIT=MIMCMDXT and specify the module name on the LOAD parameter. For example, to use the load module UEXIT1 for the MIMCMDXT routine, specify the command SETOPTION EXIT=(MIMCMDXT LOAD=UEXIT1).

Assemble and link-edit the completed routine as an authorized program with the module name you specify on the SETOPTION command.

Link-edit the routine into the authorized load library for CA MIM.

The sample member named ASMEXIT in the CAI.CBTDJCL data set contains JCL that you can use to assemble and link-edit this routine.

Note: For more information, see the *Statement and Command Reference Guide*.

Entry Environment:

Supervisor state, problem program key, addressing mode (AMODE) 31.
You must preserve this entry environment upon entry and restore it upon return.

Common Parameter List

R1

Fullword containing the address of the parameter list, in the following format:

+0

Fullword containing the address of the parameter list of the MIMCMDXT exit routine. This is not used for initialization calls.

+4

Fullword containing the communication word of the MIMCMDXT routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during the second initialization, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the routine. The high order bit of the first byte is set on X'80' for the initialization call.

+C

Fullword containing the global communication word set by the MIMINIXT exit routine.

R10

This register is reserved. Do not alter it if you are using a message subroutine to issue messages.

R11

This register is reserved. Do not alter it if you are using a message subroutine to issue messages.

R13

This register contains the address of one standard save area.

R14

This register contains the return address for CA MIM.

R15

This register contains the entry point address for the MIMCMDXT routine.

All other registers are undefined.

MIMCMDXT Exit-specific Parameter List

+0

Fullword containing the address of a two-word vector. The first word of the vector contains the address of a command name, and the second word of the vector contains the true length of the command name. Do not alter this information.

+4

Fullword containing the address of the command text buffer. You can change the command text contained in this buffer.

+8

Halfword containing the true length of the command text buffer.

+A

(Two-bytes reserved for user exit compatibility)

+C

Fullword containing the address of the CA MIM message routine that you can use to send a message to the command issuer. If you want to use this routine, then make sure your exit provides a save area of at least 72 fullwords and that registers 10 and 11 are the same as on entry to your exit when the message routine is called.

+10

Fullword containing the address of the MCS authority flags.

+14

(12 reserved bytes)

+20

Fullword containing the four-byte console ID.

+24

One-byte unsigned integer containing the command source type defined as follows:

0 Real MCS console
4 Extended MCS (EMCS) console with migration ID
8 Extended MCS (EMCS) console
12 Subsystem console
16 Product (N/A)
20 TSO user
24 INSTREAM (JCL converter)
28 INTERNAL (no console)

+25

(Three bytes reserved)

+28

Eight-byte command source name. Can be console name, TSO user ID, INSTREAM, or INTERNAL depending on command source type.

+30

Eight-byte Command and Response Token (CART). Binary zeros if not specified or available.

+38

(16 bytes reserved)

+48

Eight-byte MIMCMDXT user exit parameter list eye catcher “MIMCMDXT.”

Return Code, Register 15:

0

CA MIM should process this command normally.

4

CA MIM should reject this command and issue message MIM0140 to notify the issuing console or TSO user.

8

CA MIM should reject this command but should not issue a message to notify the issuing console or TSO user.

If you specify a value other than 0, 4, or 8, then CA MIM rejects this command and issues MIM0141 to notify the issuing console or TSO user.

Note: You must restore all registers (except 0, 1, and 15) upon return.

Issuing Messages From MIMCMDXT

If you need to send a message from the MIMCMDXT routine to the console or TSO user who issued a command, then use the message subroutine provided at offset +C of the exit-specific parameter list. This message is recorded in the MIM trace data set if the TRACE feature is active. The next example shows mandatory code for the message subroutine:

```
CMDEXIT  SAVE   (14,12)
         LR     R12,R15
         USING  CMDEXIT,R12
         LR     R5,R1          *Saves input parameters
         ST     R13,BIGSAVE+4  *Preserves the save area pointer
         LA     R13,BIGSAVE    *Points to the large save area
         LA     R1,LFORMWTO    *Points at the list-form of the WTO message
         L      R15,12(R5)     *Points to the message subroutine
         BALR   R14,R15        *Calls the message subroutine
BIGSAVE  DS     72F            *Save area and workspace
LFORMWTO WTO    'message text',MF=L,MCSFLAG=(RESP,REG0)
```

MIMINIXT Exit

You can use the optional MIMINIXT exit routine to acquire and initialize a global exit communication area that can be shared among exit routines for any CA MIM facility. You can use this communication area to allow routines to share site-specific data, such as a predefined user table.

CA MIM calls the MIMINIXT exit routine once per startup and before the initialization call for any other routine. On entry, +C in the common exit parameter list is either zero or the previous value from when you last shut down CA MIM. The flag byte at +8 is always set to X'80'.

The address of the global data area that is acquired and initialized must be stored at +C in the common exit parameter list before the routine returns to CA MIM.

Follow these rules when coding the MIMINIXT routine:

- Preserve all registers except registers 0, 1, and 15.
- Consider the execution environment for all user-specified exit routines when issuing a GETMAIN request for a global data area.
- Note that CA MIM does not provide serialization for the updating of a global data area. The serialization inherent in the execution environment of the routine is the only serialization that occurs.
- Use the INITEXIT parameter on the MIMINIT statement to identify the appropriate load module for the MIMINIXT routine. If you do not provide the module name on the INITEXIT parameter, then CA MIM tries to load and activate module MIMINIXT.

  Note: Specifying MIMINIT INITEXIT=NONE causes CA MIM to bypass all MIMINIXT processing.

  The sample member named MIMINIXT in the CAI.CBTDSRC data set contains a sample MIMINIXT exit routine.

Entry Environment:

Supervisor state, problem program key, addressing mode (AMODE) 31.
You must preserve this entry environment and restore it upon return.

Common Parameter List

R1

This register contains the address of the parameter list in the following format:

+0

This fullword is not used, therefore the value is unpredictable.

+4

Fullword containing the communication word of the MIMINIXT exit routine. This is initially set to zero. The content of this word may be set by the routine during its initialization call and is passed unchanged during the next restart whenever possible.

+8

Fullword of flags used to pass status information to the routine. The MIMINIXT exit routine is always called with the first byte of the status flag, which is set to X'80'.

+C

Upon entry to the MIMINIXT exit routine, this fullword contains zeros or the contents of the global communication word from a previous execution of CA MIM. The MIMINIXT exit routine may set the value of this word. The value set is passed to all other exit routines every time they are called, and is also passed to the MIMINIXT exit routine if CA MIM is restarted.

The common parameter list is mapped by the UXPARM member of the CAI.CBTDMAC data set.

R13

This register contains the address of one standard save area.

R14

This register contains the return address for CA MIM.

R15

This register contains the entry point address for the MIMINIXT routine.

All other registers are undefined.

Note: No return codes are supported.

Chapter 6: Utilities and Other Interfaces

This section contains the following topics:

CTC Path Validation Utility

Cross-system channel-to-channel communication paths consist of an end-point device on one system and an end-point device on another system. The two end-point devices are connected using a hardware channel path. The channel path and end-point device addresses are created using HCD or IOCP on each system involved. The CTC Path Validation Utility allows sites to test the CTC communication path to ensure the two end-point devices are connected to one another.

Utility Components

The CTC Path Validation Utility consists of two components:

- **The MIMCTPRF Program**

  This program should have been downloaded automatically as part of the standard CA MIM product installation process. Check your CA MIM load library to ensure this program was downloaded during the install process.

- **The PROCCTC JCL Procedure**

  The sample utility JCL PROC should also have been downloaded as part of the standard CA MIM product installation. Check the CAI.CBTDPROC data set for the sample CTC Path Validation Utility JCL PROC named PROCCTC.

Install the Utility

The utility needs to be activated on both systems owning the end-point CTC devices in question.

The device addresses are passed on through the z/OS START command used to activate the utility. If the two addresses provided represent the same CTC connection, then the utility exchanges system information and displays the result of the exchange. If the two device addresses fail to communicate, then other error messages may be issued, and the CTC validation I/O from the utility remains pending on each device until the utility is canceled through operator command.

The general syntax for the checking 3-digit CTC devices is:

```
START PROCCTC,ddd
```

The general syntax for the checking 4-digit CTC devices is:

```
START PROCCTC,/dddd
```

Example:

Assume you were told device 810 on SYSA was connected to SYSB device 910. To verify this connection, perform the following steps:

1. Customize the PROCTC JCL PROC, rename it to CTCTEST, and copy it to an appropriate PROCLIB.
2. On SYSA, issue command START CTCTEST,810.
3. On SYSB, issue command START CTCTEST,910.
4. Monitor message traffic on SYSA and SYSB. If the utility ends normally on each system, then the two devices were able to communicate to one another across the complex. This typically occurs a few seconds after the START commands are issued on each system.
5. If the utility runs successfully, then you could define this particular channel-to-channel connection to CA MIM using CTCPATH statements placed in the MIMINIT parmlib member:

   ```
   CTCPATH  FROMSYS=SYSA  ADDR=810  TOSYS=SYSB
   CTCPATH  FROMSYS=SYSB  ADDR=910  TOSYS=SYSA
   ```

6. If the utility determines the devices are unable to communicate cross-system, then contact your local system programmer or OEM Service Engineer for hardware connectivity assistance. In this case, you need to terminate the utility on each system by using the z/OS CANCEL command.

DASD Reserve Validation Utility

One way to allow multiple operating systems to access a common data structure would be to place the data structure on a DASD device that has been genned with the “shared” attribute. Programs accessing that data structure could serialize access to the data structure by using the IBM RESERVE/DEQ macros. Serialized access is achieved with hardware reserves if the device is genned properly on all systems, and the macros are coded properly in all programs accessing the data structure. DASD controller hardware reserve microcode ensures that only a single program on a single system is able to read and update the data structure at any given moment.

The CA MIM DASD Reserve Validation Utility allows sites to determine whether the hardware reserve process is working properly for a particular shared DASD device. If a reserve failure is detected, then an error message is issued and the utility abends with a U0011 .

Note: The MIMPROOF program issues hardware reserves with a QNAME of MIMPROOF. This QNAME should not be specified in the MIMQNAME member if the SELECT is specified on GDIINIT PROCESS. If ALLSYSTEMS is specified on GDIINIT PROCESS, then the MIMQNAME member should be updated with the following entry: MIMPROOF GDIF=NO

Utility Components

The DASD Reserve Validation Utility consists of two components:

- The MIMPROOF Program This program should have been downloaded automatically as part of the standard CA MIM product installation process. Check your CA MIM load library to assure this program was downloaded during the install process.
- The PROCRSV JCL Procedure The sample utility JCL PROC should also have been downloaded as part of the standard CA MIM product installation. Check the CAI.CBTDPROC data set for the sample DASD Reserve Validation Utility JCL PROC named PROCRSV.

Install the Utility

The utility requires that a test data set be allocated on the shared DASD device in question. Then, the JCL PROC of the utility needs to be customized and started on all systems that have access to the DASD device being tested. The utility remains active on all systems until 1) it is terminated using a z/OS CANCEL command, 2) it is terminated by replying to its outstanding WTOR, or 3) it detects a reserve failure, issues an error message, then abends with a U0011.

To install the DASD reserve validation facility

1. Allocate a test data set on the shared DASD volume in question. The following is a sample JCL to perform the allocation:

```
//ALLOCDS JOB 1,'allocate data set',CLASS=A
// *
//ALLOC EXEC PGM=IEFBR14
//SHARED DD DISP=(NEW,CATLG),DSN=MIM.PROOF.TEST.DATASET
// UNIT=xxxx,VOL=SER=XXXXXX,SPACE=(TRK,1)
```

2. Customize the PROCRSV JCL PROC, rename it to RSVTEST, and copy it to an appropriate proclib.

3. Start the RSVTEST JCL PROC on all systems that have access to the DASD device being tested. In a complex having eight systems accessing the same DASD device, the z/OS START commands would be coded as follows:

```
On SYSA: START RSVTEST,INDEX=0,CYCLE=02,FORMAT=F (on 1st system)
On SYSB: START RSVTEST,INDEX=1,CYCLE=03 (on 2nd system)
On SYSC: START RSVTEST,INDEX=2,CYCLE=04 (on 3rd system)
On SYSD: START RSVTEST,INDEX=3,CYCLE=05 (on 4th system)
On SYSE: START RSVTEST,INDEX=4,CYCLE=06 (on 5th system)
On SYSF: START RSVTEST,INDEX=5,CYCLE=07 (o   n 6th system)
On SYSG: START RSVTEST,INDEX=6,CYCLE=08 (on 7th system)
On SYSH: START RSVTEST,INDEX=7,CYCLE=09 (on 8th system)
```

Notes:

- INDEX is the relative system index number having a range of 0-7.
- CYCLE is the reserve delay interval having a range of 00-99(decimal).You must include the leading 0 if the value is 9 or less. i.e. “09” not “9” This value determines the number of 0.1 second intervals that MIM will wait while reserving the file, and also how long MIM will wait after it has released the file. For example, if CYCLE=05, then MIM will reserve the file for .5 seconds, release the file, wait another .5 seconds, then reserve it again. So, MIM should have a total of approximately 1 reserve per second. The best test results are achieved by staggering this value on each system as shown above.
- FORMAT has a value of F or N and is used to initialize the test data set. The first utility proc started must specify FORMAT=F; all subsequent systems must specify FORMAT=N.

4. Monitor message traffic on all eight systems for MIMPROOF messages. When the utility is active on each system, a WTOR is issued that can be used to terminate the utility manually. The utility runs indefinitely until you respond to the WTOR of the utility, or a hardware reserve failure is detected. If a reserve failure is detected, then another MIMPROOF error message is issued containing some diagnostic data, and it will abend with a U0011. The utility cannot determine why the failure occurred. It can only tell you that at least two systems were able to gain simultaneous access to the file. This should never occur if normal reserve processing is being properly maintained by the DASD controller microcode. Contact your DASD hardware vendor for assistance in determining root cause.

Parmlib SyntaxSCAN Utility

The CA MIM SyntaxSCAN Utility allows you to check CA MIM parmlib statements and commands for syntax errors. The utility runs as a started task and interrogates these CA MIM parmlib members: MIMINIT, MIMCMNDS, MIMQNAME, MIMUNITS, GDIEXMPT, and EDIPARMS. It also interrogates these CA MIM message library members: EDIMSGS, ICMMSGS, MIAMSGS, MICMSGS, MIMMSGS, and MIMMSGX. When started, the utility simulates CA MIM address space activation. Statements and commands found in the CA MIM parmlib members are parsed and checked for errors, all errors are noted, and the utility ends.

Utility Components

The CA MIM SyntaxSCAN Utility consists of two components:

- **The MIMSYNTX Program**

  This program should have been downloaded automatically as part of the standard CA MIM product installation process. Check your CA MIM load library to ensure this program was downloaded during the install process.

- **The PROCSYN JCL Procedure**

  The utility JCL PROC should also have been downloaded as part of the standard CA MIM product installation. Check the CAI.CBTDPROC data set for the sample CA MIM SyntaxSCAN Utility JCL PROC named PROCSYN.

Install the Utility

This section describes how to install the Parmlib SyntaxSCAN utility.

Remember that this utility only simulates CA MIM activation, so keep the following in mind:

- No real device or multi-system resource management occurs with the utility task.
- No CA MIM front-ends or SVC intercepts are loaded into CSA storage.
- No multi-system communications protocol is checked. For example, no DASD reserves, no CTC I/O / handshaking, no registration with XCF.

To install the Parmlib SyntaxSCAN facility

1. Customize the PROCSYN JCL PROC, rename it to MIMSCAN, and copy it to an appropriate proclib.

   Use the PROC symbolic override parameters to specify the MIMINIT and MIMCMNDS members to be scanned. You may reference the same members currently being used by currently active CA MIM tasks, or new members you have created.

   The MIMSYNCH member typically contains commands for other products (z/OS, JES2). Because CA MIM cannot accurately verify the syntax of commands for other products, the utility JCL PROC is coded to bypass scanning of the MIMSYNCH member. You will notice a SYNCH=NO parameter purposely coded in the JCL PROC of the utility; leave this as is.

   We recommend that:

   - You code LOG at the top of each CA MIM parmlib member
   - MIMINIT SUPPRESSRESP=NO is coded in the MIMINIT member to ensure that all error messages are issued when MIMSCAN runs.

2. You need to APF-authorize the CA MIM load library prior to starting MIMSCAN.
3. Start the MIMSCAN JCL PROC using the z/OS START command. You may run the SyntaxSCAN Utility alone or concurrently with other active CA MIM tasks. This is possible, because the MIMSYNTX program executes as simulated CA MIM task that really examines CA MIM parmlib members only for syntax errors.
4. Monitor the execution of the utility. Usually, the JCL PROC of the utility terminates on its own after it has completed the scan of all CA MIM parmlib members. It typically takes less than one minute to complete the scan. After the utility ends, examine the syslog or joblog to determine if any syntax errors were found. If the utility ended with a U0040 abend, then syntax errors were found in the CA MIM parmlib. Examine the error message associated with the U0040 abend to determine which statement or command was in-error, correct the error, and rerun the CA MIM SyntaxSCAN Utility. Continue this process until the utility terminates normally. That is, without any U0040 abends.

Note: Even if no U0040 abend occurs, there may be unrecognized commands in the commands member. Review the joblog or syslog for any error messages.

How You Use the SyntaxSCAN Utility

You may run the SyntaxSCAN utility alone or concurrently with active CA MIM address spaces, and you may reference the same parmlib and members.

For example, suppose you have an active CA MII task for which you have dynamically added a QNAME using the ADDQNAME command. You have decided to make this a permanently managed QNAME, so you have appended the new QNAME definition to the end of the MIMQNAME member. You now want to be sure that the newly specified QNAME definition is syntactically correct.

You could customize the SyntaxSCAN utility startup procedure to point to the same parmlib and members that the CA MII task uses during production. Then, you could execute the SyntaxSCAN utility concurrently with the active CA MII task to verify all parmlib changes. Any syntax errors would be noted by a U0040 abend, as well as by error messages that would indicate the problem.

TSO Command Interface Utility

The TSO Command Interface Utility is an optional CA MIM service that allows TSO users to issue CA MIM commands. Minimally, it enables TSO users to issue CA MIM DISPLAY commands from their individual TSO sessions. The command responses are then directed back to the TSO user.

If you are running CA MIC, then the basic CA MIM TSO Command Interface Utility can be extended to execute any CA MIM, z/OS, or z/OS subsystem command. Commands can be routed to any or all systems in the MICplex.

You can optionally code the MIMCMDXT and GCMCMDXT user exits to prohibit the execution of specific commands based on the TSO user ID.

You are required to have a valid CA MIM command prefix (CMDPREFIX) defined for this interface to function. If you have specified CMDPREFIX=NONE, then the MIMTSO interface will not recognize the active CA MIM address space.

Utility Components

The TSO Command Interface Utility consists of two components:

- **The MIMTSO Program**

  The MIMTSO program should have been downloaded as part of the standard CA MIM product installation process. Check your CA MIM load library to ensure the MIMTSO program was downloaded as part of the installation process.

- **The MIMTSO CLIST**

  The MIMTSO CLIST should have been downloaded as part of the standard CA MIM product installation process. Check the CAI.CBTDCLS0 data set to ensure the MIMTSO CLIST was downloaded as part of the installation process.

Install the Utility

The TSO Command Interface Utility can be implemented in two ways - as a *command processor*, or as a *called program*.

- Invoking the MIMTSO program from a TSO session as a command processor allows you to directly append the desired command to the MIMTSO command. The MIMTSO program will immediately be executed, the appended command will then be executed, and the result will be return to the TSO session.
- Invoking the MIMTSO CLIST from a TSO session as a called program requires you to invoke the MIMTSO CLIST first. Once the MIMTSO CLIST executes, you will be presented with a READY prompt indicating that you can then present commands to CA MIM for execution.

Follow the installation instructions for the specific method in which you would like to invoke the TSO Command Interface Utility.

To install the interface as a command processor

Add the MIM APF-Authorized loadlib that contains the MIMTSO program so that it will be automatically found through LINKLST search processes.

1. Authorize the TSO service facility to call the MIMTSO program as a command processor by adding the MIMTSO program name to the AUTHCMD and AUTHPGM statements in SYS1.PARMLIB(IKJTSOxx).
2. If you are running CA MIC and would like to issue commands to any system in the MICplex, then you will need to add a CA MIC LINK command for each TSO user ID desiring this ability. Here is a sample:

   ```
   LINK TSOUSER=OPER1  SYSID=ALL  AUTH=MASTER
   ```

   This sample LINK command allows the TSO userid “OPER1” to issue any command to any system in the MICplex.

Example: Local Command

```
MIMTSO @DISPLAY SYSTEMS
```

Where @ is the CA MIM command character.

Example: Global Command

```
MIMTSO @SYSB @DISPLAY SYSTEMS
```

Where the SYSB is the CA MIM name of an external system.

To install the interface as a called program

1. Customize the LIBRARY parameter in the sample MIMTSO CLIST found in the data set CAI.CBTDCLS0 so that it points to your CA MIM APF-authorized loadlib.
2. Copy the updated MIMTSO CLIST to a CLIST library that is automatically searched for TSO users. The search order for CLIST libraries is defined using the //SYSPROC DD statement found in the logon procedure of each TSO user.
3. Authorize the TSO service facility to call the MIMTSO program as a command processor by adding the MIMTSO program name to the AUTHCMD and AUTHPGM statements in SYS1.PARMLIB(IKJTSOxx).
4. If you are running CA MIC and would like to issue commands to any system in the MICplex, then you will need to add a CA MIC LINK command for each TSO user ID desiring this ability. Here is a sample:

   ```
   LINK TSOUSER=OPER1  SYSID=ALL  AUTH=MASTER
   ```

5. This sample LINK command will allow the TSO userid OPER1 to issue any command to any system in the MICplex.
6. To invoke the CLIST, you would enter:

   ```
   %MIMTSO
   ```

   %

   Specifies the TSO CLIST command character

   MIMTSO

   Specifies the CLIST name

   Following is an example of MIMTSO when executed on a system where CA MII and CA MIA are running in separate address spaces:

   ```
     MIM/TSO command interface (11.6      CTD1150)
   THE FOLLOWING MIM TASKS ARE ACTIVE:
       MIAPROC   USES COMMAND PREFIX(:       )
       MIIPROC   USES COMMAND PREFIX(&       )
   MIMTSO
   ```

How You Toubleshoot the TSO Command Interface Utility

Follow the suggested guidelines to diagnose the following common errors associated with invoking the TSO Command Interface Utility:

S047 Abends

If MIMTSO is installed as a command processor, then verify that the MIMTSO module is in a LINKLST library.

If MIMTSO is called from a CLIST, then verify that:

- The TSO EXEC call is used, if you called the CLIST from ISPF.
- The library that contains the MIMTSO module is explicitly APF-authorized.

If MIMTSO is installed as a command processor or as a called program, then verify that:

- The library that contains the MIMTSO module is APF-authorized.
- The MIMTSO module is linked AC=1.

Incomplete Command Responses

The HIBFREXT and BUFRSIZE parameters found in SYS1.PARMLIB(TSOKEYxx) regulate the number of command data you may receive in a TSO session. The total lines you can receive at any one time are equal to the HIBFREXT value divided by the BUFRSIZE value. The total line output can vary due to timing considerations when a user presses the Enter key while messages are being sent to the TSO session. BUFRSIZE=132 and HIBFREXT=6600 gives you a 50-line maximum.

No Output When Using MIMTSO as a Command Processor

Verify that the your TSO PROFILE specifies INTERCOM rather than NOINTERCOM in your TSO user profile. You can check your profile values by issuing the TSO PROFILE command.

'Invalid Command' Message from TSO or ISPF

If MIMTSO is a command processor, then verify that the MIMTSO program is in an APF-authorized loadlib that is being search through LINKLIST processing.

'Insufficient Authority' Messages

For z/OS commands issued through CA MIC, use the CA MIC LINK command to assign authority levels for TSO users. For all CA MIM components, you can change command authority for each TSO user from the default INFO authority by using the MIMCMDXT exit routine.

'CA MIM is not active'

The CA MIM address space needs to be ACTIVE with a valid CMDPREFIX that can be set with the MIMINIT CMDPREFIX statement or the SETOPTION MIM CMDPREFIX command. CMDPREFIX=NONE cannot be specified.

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

Message Facility

The CA MIM message facility provides the ability to customize messages issued by CA MIM. The message facility lets you to customize message text and routing attributes such as routing and descriptor codes. The default message tables are located in the CA MIM message data set, known as the MIMMSGS data set.

You specify the MIMMSGS data set on the //MIMMSGS DD statement in the JCL procedure that you use to start CA MIM. A sample MIMMSGS data set is installed with the CA MIM product into data set CAI.CBTDMSEN.

This facility can be used only to customize CA MIM messages. Only the messages that are found in the default message tables can be customized.

The message facility also lets you create non-English versions of CA MIM messages. While CA MIM does not provide default, non-English message tables, you can create customized, non-English message tables.

Default Message Tables

The default message table members are named:

```
EDIMSGS, ICMMSGS, MIAMSGS, MICMSGS, MIIMSGS, MIMMSGS, MIMMSGX
```

Default message table members must reside in the MIMSMGS data set. They define the default message attributes. You use the MIMMSGX member of the MIMMSGS data set to create site-specific message definitions that override the default attributes.

MIMMSGS Message Table

The MIMMSGS member of the MIMMSGS data set contains message facility statements that are processed as part of CA MIM initialization. The MIMMSGS member is the primary message table and points to all other message tables used by CA MIM. For the general structure of this member, see the MIMMSGS member of the CAI.CBTDMSEN data set.

The TABLE statement defines the MIMMSGS member as the primary message table from which all other message tables are called. All subsequent MSG and MSGR statements belong to the defined table. The MSG and MSGR statements in MIMMSGS define CA MIM driver messages, which can be customized. The CA MIM facility-specific message tables (MICMSGS, for example, is used for CA MIC) contain TABLE, MSG, and MSGR statements unique to that facility.

The INCLUDE statement directs the message facility to locate the indicated message table and insert the statements found in the member at the point specified.

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

How You Create a Custom Message Table

Each message table member provides access to message text that may be reworded or translated into languages other than English. You may also change various routing attributes of messages. We recommend that you do not directly edit the default message table members. Instead, define MSG and MSGR statements in the MIMMSGX member to override the MSG and MSGR statements found in the default members.

By default, the MIMMSGX member is not called during initialization because the INCLUDE MIMMSGX statement is commented in the MIMMSGS member. Uncomment the INCLUDE MIMMSGX statement to call the MIMMSGX member during initialization. The MSG and MSGR statements found therein override the corresponding MSG and MSGR statements found in the default members.

Follow these guidelines when using the MIMMSGX member:

- You can modify only messages found in the default message table members.
- Follow the message ID naming convention for CA MIM messages. Message IDs are expected to be eight characters in length. Do not change this.
- All message facility syntactical rules are enforced.
- You can use any language for message text.

Message Table Syntax Rules

The following syntax rules govern the statements that are placed in the message table members. The variable substitution capabilities that are used in the MSG Statement *text* parameter are:

- Only columns 1-71 are examined.
- Leading and trailing blanks are discarded.
- Comments can be inserted freely, and are designated by an asterisk (*) in column 1.
- Any line that ends with a comma (,) or a plus sign (+) is assumed to be continued on the next line. When splitting message text across lines, the (+) is converted to a single blank.
- Up to nine variable substitutions can be requested using text in each message. The “at” sign designates each substitution (@). Each substitution is numbered by @*n*, where *n* is a decimal number 1-9. Optionally, special formatting can be requested in the following ways:
  - A vertical bar (|) preceding the parameter number (*n*) indicates that leading blanks can be removed. The vertical bar can be coded as either X’6A’ or X’4F’, depending on the emulator.
  - A vertical bar (|) following the parameter number (*n*) indicates that trailing blanks can be removed. The vertical bar can be coded as either X’6A’ or X’4F’, depending on the emulator.
  - Periods (.) preceding the parameter number (*n*) indicate that the substituted variable can be right justified, after removing leading and trailing blanks. The width of the inserted field is 2, plus the number of periods. The substituted variable is truncated if necessary.
  - Inserting periods (.) following the parameter number (*n*) indicates that the substituted variable can be left justified, after removing leading and trailing blanks. The width of the inserted field is 2, plus the number of periods. The substituted variable is truncated if necessary.

Rule Examples

The following are examples of how to specify message table functions:

```
MSG 'MIM0001I parm @1 is inserted without change'

MSG 'MIM0002I parm @|2 is stripped of leading blanks,+ while parm @1| is stripped of
trailing blanks'

MSG 'MIM0003I parm @...1 is exactly 5 characters after+ substitution, right
justified.'

MSG 'MIM0005I parm @1... is exactly 5 characters after+ substitution, left justified.'
```

Naming Convention for Messages

The standard format suggested for CA MIM messages is:

`MIM####t`

where #### is a decimal number between 0001 and 9999, and *t* is one of the following letters used to designate the message type:

A

Indicates an action message

E

Indicates an error message

I

Indicates an informational message

W

Indicates an warning message

In this naming standard, message number ranges are reserved for use by functional areas and facilities. The current ranges for message numbers are listed here:

Range	Functional Area
MIM0001 - MIM0999	CA MIM General Messages
MIM0500 - MIM0599	GCS (z/VM) Messages
MIM0600 - MIM0799	CA MIM General Messages
MIM0800 - MIM0899	Message Facility Messages
MIM0900 - MIM0999	XCF Support Messages
MIM1000 - MIM1999	GDIF and ECMF Messages
MIM2000 - MIM2999	GTAF and TPCF Messages
MIM3000 - MIM3999	GCMF Messages
MIM4000 - MIM4999	EDIF Messages
MIM5000 - MIM5999	Open
MIM6000 - MIM6999	ICMF Messages
MIM7000 - MIM7999	Open
MIM8000 - MIM8399	Used to define the variable text segments substituted in the @n parameters.
MIM8600-MIM8799	LXCF Messages

Range	Functional Area
MIMH1000-MIMH2000	Health Check Messages

Message Facility Statements and Commands

The following statements and commands are used by the CA MIM message facility. Statements are used in message table members and are referenced during initialization, while the commands can be issued at any time to display or modify current message facility options.

- TABLE statement
- MSG statement
- MSGR statement
- INCLUDE statement
- DISPLAY MSGTABLE command
- MSGTABLE command

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

How You Control MSG Statement Logging

You can use NOLOG and LOG statements in the message facility input members to control logging of MSG and MSGR statements through MIM0079 message to the system log. Place a NOLOG statement immediately preceding those MSG statements for which you want logging suppressed. Place a LOG statement immediately preceding those MSG statements for which you want logging enabled. As distributed, all message facility input members have a NOLOG statement as their first statement.

Help Facility

The HELP facility can be used to provide additional information for CA MIM commands, statements, and messages. This section explains how to allocate a new MIMHELP library and how to use this facility.

Note: CA no longer supplies a default MIMHELP library for CA MIM. If you used this facility in a previous release of CA MIM, you can still use it with this release. However, it may contain inaccurate or incomplete information about commands and messages. You can customize your own MIMHELP library by adding, deleting, or modifying members for the HELP facility. The CAI.CBTDHENU data set contains a sample MIMHELP library.

How You Add or Delete Help Topics

You can use the MIMHELP data set if you want to make any type of information available through the CA MIM HELP facility.

You identify the MIMHELP data set using the //MIMHELP DD statement in the CA MIM startup procedure. You can add or delete informational members in this data set. You also can modify any member if you want to update the information contained in that member. The changes you make take effect the next time you start CA MIM.

How You Request Information from the Help Facility

To obtain information from the HELP facility while CA MIM is running, use the HELP command:

- To obtain a list of the topics in your MIMHELP data set, specify HELP INDEX or HELP. CA MIM provides a list of HELP topics, as well as a brief description of each topic.
- To obtain information about a specific topic (informational member), name that topic on your HELP command. For example, to see detailed information about the EXEMPT command, you would specify the command HELP EXEMPT.

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

ALTSEREXTENDED Interface

IBM made a change to the ISGNQXITPREBATCH exit parameter list in z/OS 1.7 to aid in determining if a resource is to be managed globally by CA MIM. For a resource to be globally managed by an alternate serialization product (CA MII), this exit routine should set the NQPB_ER_AltSerExtended bit for the resource. This indication would then be returned on an ISGQUERY request, and/or on an ISGENQ TEST=YES request.

Report Generation for CA MIM

CA MIM reports give you an effective tool to evaluate the performance of the systems in your complex. The reports use a sampling of statistical data to yield important information about the operating activities in your complex.

Note: This functionality requires the CA Easytrieve Interface. For more CA Common Services requirements information, see the appendix “CCS for z/OS Component Requirements” in the *Installation Guide*.

How You Plan Your Reports

CA MIM uses a single SMF record for the record collection process. Using record *subtypes*, you can identify the statistical records to be collected for the different reports available under the CA MIM facilities (such as GDIF, ECMF, TPCF, and so on.).

The major steps involved in the report generation process are:

1. Specify the SMF record number on a MIMINIT RECORDTYPE statement in the initialization member. Once this is set, you should not need to change it.
2. Specify the record collection criteria using the SETOPTION command and begin the record collection process.

   Note: You must allow the record collection process to run for a while before dumping the SMF records to data sets.
3. Dump the SMF records to a usable data set through the IBM IFASMFDP utility.
4. Run a report or create a report output file.

Important! CA MIM provides a set of reports for general use. If, however, any one of these reports does not meet your specific requirements, then you can write your own customized reports in the language of your choice, using the SMF records supplied by CA MIM.

Note: For more information, see the MIMSTREC member in the CAI.CBTDMAC data set provided on your installation tape.

The default DETAIL reports do not produce an end-of-day summary. You can obtain a summary report using the EACH parameter setting.

Specify an SMF Record Number

Before any statistical records can be collected for use in the CA MIM report process, you need to specify an SMF record number. To do this, you place a MIMINIT RECORDTYPE initialization statement in the initialization member.

For example, if you decide to use the default SMF record number of 189, then you would specify the following statement in the initialization member:

```
MIMINIT RECORDTYPE=189
```

Now you are ready to specify the record collection criteria for the report or reports you want to run.

Specify a Record Type

Use the SETOPTION MIM STATCOLLECT command to specify the type of statistical records to be collected.

Specify the SUBTYPE operand to select specific record subtypes for a facility, or specify ALL to collect records for all record subtypes available under that facility. The record subtypes correspond directly to the reports available for each facility.

The following are record subtypes for the CA MIM facility:

Record Subtype	Facility	Description of Report
CF	CA MIM	CA MIM Control File Performance report is used to assist you in tuning the CA MIM control files.
FC	CA MIM	CA MIM Control File I/O report details activity in the CA MIM control files. It can be used in conjunction with the CF report to assist with control file placement.
VF	CA MIM	CA MIM Virtual Control File I/O Performance report details the activity on the CTC devices.

For example, the following command tells CA MIM to collect statistics for the report subtype CF:

```
SETOPTION MIM STATCOLLECT(SUBTYPE=CF)
```

Important! To activate the record collection process for all record subtypes, you must repeat this procedure for each subtype, specifying the SETOPTION STATCOLLECT command and the facility associated with that record subtype.

Sample Command for Record Collection

The following example shows how to begin the SMF record collection process for the *CA MIM Control File Performance* report. This example indicates data sampling every 30 seconds, and a statistical record recording interval of every hour:

```
SETOPTION MIM STATCOLLECT(SUBTYPE=CF) STATCYCLE=30, STATINTERVAL=60
```

STATCYCLE

Specifies the sampling cycle time, in seconds, which CA MIM waits between each statistical data sampling. The default value is 60.

STATINTERVAL

Specifies how often, in minutes, statistical data samples are recorded to SMF data sets. The default value is 15.

Note: The STATCYCLE and STATINTERVAL values apply to all record collection subtypes for a given CA MIM facility (GDIF, and so on).

Once the statistical record collection process has begun, records are collected and written to the SMF data sets. You need to use your site-specific procedures for dumping the records to physical sequential data sets that can be used by CA MIM to produce reports.

Note: For more information, see the *CA MIM Statement and Command Reference Guide*.

Dumping the SMF Records

After you have created the statistics records using the SETOPTION commands, you must put these records into a usable form for the CA MIM report generation utility or the record selection utility. To do this, you need to execute the IBM utility IFASMFDP.

The IFASMFDP utility copies all records from the SMF data sets to SMF dump data sets for permanent storage. The dumped data sets are of a physical sequential organization and have a variable blocked spanned (VBS) record format. This format allows CA MIM to either run a hardcopy report from these records or put these records into record subtype output files for later report processing.

Note: For more information about the SMF dump program procedure, see the appropriate IBM System Management Facility (SMF) guide.

Generating a Report

Once you have dumped the SMF records, you can generate a hardcopy report using the CA MIM report generation utility, MIMEZRPT. This utility program is designed to execute as a batch job to post-process the SMF records previously created by the CA MIM address space.

To use this utility, you need to code JCL statements to specify various input and output files required by the report generation utility, as well as code control cards to request a given report and specify any desired customizing options for the specified report. Reports are generated on demand, and the report output contains information based on the statistics collected. Sample JCL for running hardcopy reports is shown in Sample JCL for Report Generation in this chapter.

DDnames Used by MIMEZRPT

The following ddnames are used by the MIMEZRPT utility:

ERRORS

Specifies the data set to which the MIMEZRPT utility program writes runtime error messages. Typically, this is a spooled data set.

EZTVFM

Specifies the work data set available for use by the MIMEZRPT utility program.

PANDD

Specifies the CAI.CBTDEZTM data set containing the CA Easytrieve ZZTMACxx macros used for report generation.

REPORTS

Specifies the data set to which the MIMEZRPT utility generates the requested report. Usually, this is a spooled data set.

SMFIN

Specifies the data set containing the CA MIM SMF records used to produce the named reports in variable, blocked, or spanned format. Data set concatenation is supported, including data sets residing on unlike device types (for example, mixed disk and tape).

STEPLIB

Specifies the CAI.CBTDLOAD load module library containing the MIMEZRPT utility program.

SYSIN

Specifies the CAI.CBTDEZTR data set containing the CA Easytrieve report generation programs.

Report Generation Requests Using CA Easytrieve

Report generation requests are supplied to the MIMEZRPT utility program in card image format. The card images are contained in a PDS member specified on the REPORT symbolic in the MIMEZRPT JCL supplied in the CAI.CBTDJCL library. This PDS data set name is specified on the SYSIN DD in the same JCL. The SYSIN input contains the name of the report to be generated and any keywords required to run the report.

There is a limit of one report per single execution of the MIMEZRPT utility program. If you need several reports, you must execute multiple jobs or multiple job steps.

Note: For a description of how to code the control card images, see Control Card Syntax for MIMEZRPT SMF Report Generation (see page 204) in this chapter.

Several keywords are supplied for optional customizing of statistical reports.

The following common keywords are available for use in the SYSIN input for customizing the CF, FC, and VF SMF reports.

RECTYPE

Specifies the SMF record type. Default is 255

SHIFT

Specifies a time range for which records are to be included. Default is '00:00 24:00', which covers the entire day.

FROM

Specifies a starting timestamp, in the form 'YYYY/MM/DD HH:MM'; SMF records with timestamps earlier than this value are discarded. Default is 'ALL'.

TO

Specifies a starting timestamp, in the form 'YYYY/MM/DD HH:MM'; SMF records with timestamps after this value are discarded.

MIMNAME

Specifies the name of a CA MIM address space; only records written by the specified CA MIM are included. Default is '*', which selects data for all CA MIM tasks; data for multiple MIM tasks are separated, because the name is part of the sort specification.

SYSTEM

Specifies the name of a system; only records written by the specified system are included. Default is '*', which selects data for all systems; data for multiple systems are separated because the name is part of the sort specification

LINECOUNT

Determines the page breaks for the report by specifying how many lines are printed on each page. Specify a number from 1 to 32767.

EACH

Specifies the detail line grouping interval. The default is RECORD. The options for EACH are MINUTE, HOUR, DAY, MONTH, and RECORD.

The following unique keyword is available for use in the SYSIN input for customizing the VF SMF report.

SKIPZEROS

Controls whether paths that have had no activity should be omitted from the report. Default is 'N'

Sample JCL for Report Generation

For sample JCL to generate a report, see the MIMEZRPT member in the CAI.CBTDJCL data set.

Control Card Syntax for MIMEZRPT SMF Report Generation

Control cards are processed as 80-byte card images. However, only card columns 1 through 72 are examined for parsing purposes. Card columns 73 through 80 are ignored. Comments are supported and begin with an '*' (asterisk). When the first non-blank character of a statement is an '*' (asterisk), the remainder of that record is a comment statement. You can use comment statements at any place within a program, except within a continued statement. A multiple line comment requires an '*' (asterisk) as the first non-blank character of each line statement.

Control card operands are not column-dependent and can begin in any card column. Either a space, or a comma followed by a space, should be placed between the report macro specification and the first control card keyword. Either a space or a space followed by a comma should separate sequences of control card keywords and their operand values. The following example illustrates the general rules for control card data entry:

```
*-------------------------------------------------------------
* This is a comment that extends over multiple cards *
*-------------------------------------------------------------
%REPORT_MACRO KEYWORD OPERAND1, KEYWORD2 OPERAND2
```

Control card continuation is supported over any number of control card images. Control card continuations should be placed following a keyword/operand sequence. To continue a control card, a plus sign (+) or a hyphen (-) should be placed in any card column following the comma, up to column 72.

The last non-blank character of a statement terminates the statement unless that character is a - (hyphen) or a + (plus). The - indicates that the statement continues at the start of the next statement area. The + indicates that the statement continues with the first non-blank character in the next statement area. The difference between - and + is important only when continuing words.

Example: Control Card Syntax

This example illustrates the general rules for continuing control card specifications:

```
*-------------------------------------------------------------
* This is a comment that extends over multiple cards
*-------------------------------------------------------------
%REPORT_MACRO KEYWORD1 OPERAND1, +
KEYWORD2 OPERAND2, +
KEYWORD3 OPERAND3
```

Sample Reports

This section contains sample reports.

Control File Performance Report Summary (CF)

```
    (1)                 (2)                                                                                    (3)
    2004/07/13 14:06   CA MIM Control File Performance Report                                          PAGE     1
 (4)System:     SYSA              (5)Jobname:  MIMGR      (6)Release:   11.6      (7)Service Level:  0000
 (8)From:       2003/10/25 00:00  (9)Each:     RECORD
(10)To:         2004/03/12 10:52 (11)Shift:    00:00 24:00
                     (13)    (14)     (15)      (16)      (17)       (18)      (19)    (20)   (21)     (22)  (23)   (24)
                      CF    Engage   ScanFile    Heart     Diseng     After     Wait    Blocks Blocks   XACT   XACT   XACT
(12)Interval Start  Cycles   Time      Time       Time      Time       Time      Time      In     Out     In     Out    Proc
  ---------------- ------ --------- --------- --------- --------- --------- --------- ------ ------- ------ ------ ------
   2004/03/07 16:58   3970  0.005413  0.000098  0.000022  0.001640  0.000058  0.219573   1.43   1.33  206.4    1.4   31.2
                                       ECMF  0.000001            0.000000                            0.0    0.0
                                       GDIF  0.000019            0.000056                           31.2    1.4
```

Field	Description
(1)	Date and time the report was created.
(2)	Title.
(3)	Page number of the report.
(4)	System name of CA MIM image as defined by the DEFSYS statement.
(5)	Name of the CA MIM started task.
(6)	The CA MIM release number.
(7)	The CA MIM service level.
(8)	Date and timestamp of the earliest record in the reporting interval.
(9)	Summary level for the data line. In this example, each RECORD is displayed in the detail line. If each HOUR was specified, then the detail line would be a summary of the hour's activity.

Field	Description
(10)	Date and timestamp of the latest record in the recording interval.
(11)	Range of records included in the report.
INTERVAL START(12)	Start date and timestamp that the report line interval data covers.
CF CYCLES (13)	The total number of control file accesses made by CA MIM during the reporting interval.
ENGAGE TIME (14)	The average elapsed time, in seconds, during the reporting interval to obtain the control file RESERVE and read the first block from the control file. The first block contains the GMR (Global Master Record), the block allocation mask, and as many transactions as will fit in the remainder of the block. Once the control file RESERVE is obtained, CA MIM on the local system has exclusive ownership of its data structures in relation to the external systems.
SCANFILE TIME (15)	The average elapsed time, in seconds, during the reporting interval to examine transactions in the blocks read in for transactions not seen yet by the local system and to disburse those transactions to the various facilities, including the driver, on the local system. During this time, control blocks on the local system are altered (built, changed, or deleted) based on transactions from the external systems to reflect changes that have occurred on the external systems. If the number of transactions has extended to multiple blocks, then any control file read or write I/O time related to examining and disbursing the transactions to the local system is included in this time. Usually, this time appears as 0.000 if, on the average, all CA MIM transactions are contained in the first block. Important! Increases in this column may indicate that CA MIM is rereading transactions that have not been processed yet by other systems.
HEART TIME (16)	The average elapsed time, in seconds, during the reporting interval that the CA MIM driver and all active facilities spent in the performance of global activities on the local system. During this time, control blocks on the local system can be altered (built, changed, deleted) to reflect requested changes on the local system. Transactions detailing the applied data structure changes are added to the control file. This column shows the total, average, elapsed time for all processing and then breaks out the amount of time each facility used of the total, average, elapsed time.
DISENGAGE TIME (17)	The average elapsed time, in seconds, during the reporting interval required to write the final updated block, if any, to write the GMR record, and to release the control file RESERVE.

Field	Description
AFTER TIME (18)	The average elapsed time, in seconds, during the reporting interval, that the CA MIM driver and all active facilities spent in the performance of activities on the local system that were a result of the control file cycle that completed, but that do not require ownership of the control file for serialization of data structures. It is during this time that units of work in other address spaces on the local system that have been waiting on the CA MIM address space for service are notified to continue execution. This column shows the total average elapsed time for all processing and then breaks out the amount of time each facility used of the total elapsed time.
WAIT TIME (19)	The average elapsed time, in seconds, during the reporting interval that CA MIM remained dormant while waiting for an event to cause the need to access the control file.
BLOCKS IN (20)	The average number of control file blocks read per control file cycle during the reporting interval.
BLOCKS OUT (21)	The average number of control file blocks written per control file cycle during the reporting interval.
XACT IN (22)	The average number of transactions read per control file cycle. This column shows the total average number of transactions read for all processing and then breaks out the average number of transactions that can be attributed to a given facility. Note that only global facilities receive transactions.
XACT OUT (23)	The average number of transactions written per control file cycle. This column shows the total average number of transactions written for all processing and then breaks out the average number of transactions that can be attributed to a given facility. Note that only global facilities generate transactions.
XACT PROC (24)	The average number of transactions processed per control file cycle.

Control File I/O Report Summary (FC)

```
   (1)                (2)                                                                                    (3)
   2004/07/13 14:07   CA MIM Control File I/O Report                                                  PAGE    1
 (4)System:     SYSA               (5)Jobname:  MIMGR      (6)Release:   11.6        (7) Service Level:  0000
 (8)From:       2003/10/25 00:00   (9)Each:     RECORD
(10)To:         2004/03/12 10:52  (11)Shift:    00:00 24:00
                 (13)             (15)             (17)                (19)                (21)                (23)
        (12)      CF     (14)     Average (16)     Average  (18)       Average  (20)       Average     (22)     Average
 Interval Start  Cycles  BLKS RD BLKS RD BLKS WR BLKS WR  XACT RD    XACT RD   XACT WR    XACT WR   XACT PR    XACT PR
---------------- ------- ------- ------- ------- ------- ---------- --------- ---------- --------- ---------- ---------
2004/03/07 16:58    3970    5664   1.427    5265   1.326     819297   206.372       5709     1.438     123842    31.194
2004/08/29 15:58      11      15   1.364      19   1.727          8     0.727         10     0.909          8     0.727

  (24)
  ***Migrate from: Master=SYSA     VOL=****** DSN********
  ***Migrate to:   Master=SYSB     VOL=****** DSN********

2004/08/29 15:58      22      26   1.182      26   1.182         10     0.455          2     0.091         10     0.455

  (24)
  ***Migrate from: Master=SYSB     VOL=****** DSN********
  ***Migrate to:   Master=SYSA     VOL=****** DSN********

2004/08/29 15:58      11      12   1.091      11   1.000          8     0.727         10     0.909          8     0.727

  (24)
  ***Migrate from: Master=SYSA     VOL=****** DSN********
  ***Migrate to:   Master=******** VOL=MIMX01 DSN=MIM.CF01

2004/08/29 15:58      10      11   1.100      11   1.100          4     0.400          8     0.800          4     0.400

  (24)
  ***Migrate from: Master=******** VOL=MIMX01 DSN=MIM.CF01
  ***Migrate to:   Master=SYSB     VOL=****** DSN********

2004/08/29 15:58      12      16   1.333      16   1.333          8     0.667          2     0.167          8     0.667

  (24)
  ***Migrate from: Master=SYSB     VOL=****** DSN********
  ***Migrate to:   Master=SYSA     VOL=****** DSN********

2004/08/29 15:59      60      60   1.000      60   1.000          8     0.133          2     0.033          8     0.133

  (25)
  ***Current CF: Master=SYSA     VOL=****** DSN********
```

Field	Description
(1)	Date and time the report was created.
(2)	Report title.
(3)	Report page number.
(4)	System name of the CA MIM image as defined by the DEFSYS statement.
(5)	Name of the CA MIM started task.
(6)	The CA MIM service number.
(7)	The CA MIM PTF level.

Field	Description
(8)	Date and timestamp of the earliest record in the reporting interval.
(9)	Summary level for the data line. In this example, each RECORD is displayed in the detail line. If each HOUR was specified, then the detail line would be a summary of the hour's activity.
(10)	Date and timestamp of the latest record in the recording interval.
(11)	Range of records included in the report.
INTERVAL START(12)	Start date and timestamp that the report line interval data covers.
CF CYCLES (13)	The total number of control file accesses made by CA MIM during the reporting interval. This number is similar to the COUNT: CYC value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
BLKS RD (14)	The total number of control file blocks read by CA MIM during the reporting interval. This number is similar to the COUNT: BLOCKS READ value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
AVERAGE BLKS RD (15)	The average number of control file blocks read per control file access by CA MIM during the reporting interval. This number is similar to the AVG: BLOCKS READ value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
BLKS WR (16)	The total number of control file blocks written by CA MIM during the reporting interval. This number is similar to the COUNT: BLOCKS WRITTEN value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
AVERAGE BLKS WR (17)	The average number of control file blocks written per control file access by CA MIM during the reporting interval. This number is similar to the AVG: BLOCKS WRITTEN value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
XACT RD (18)	The total number of control file transactions read by CA MIM during the reporting interval. This number is similar to the COUNT: XACT READ value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.

Field	Description
AVERAGE XACT RD (19)	The average number of control file transactions read per control file access by CA MIM during the reporting interval. This number is similar to the AVG: XACT READ value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
XACT WR (20)	The total number of control file transactions written by CA MIM during the reporting interval. This number is similar to the COUNT: XACT WRITTEN value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
AVERAGE XACT WR (21)	The average number of control file transactions written per control file access by CA MIM during the reporting interval. This number is similar to the AVG: XACT WRITTEN value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
XACT PR (22)	The total number of control file transactions processed by CA MIM during the reporting interval. This number is similar to the COUNT: XACT PROCESSED value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
AVERAGE XACT PR (23)	The average number of control file transactions processed per control file access by CA MIM during the reporting interval. This number is similar to the AVG: XACT PROCESSED value displayed in the MIM0039 message generated by the DISPLAY MIM IO command.
(24)	When a control file migration takes place during the reporting interval, this report line is generated to show that a control file migration has taken place. It shows the prior control file location (first line) as well as the active control file location (second line) at the end of the reporting interval.
(25)	The location of the control file at the end of the reporting interval. CF and VOLSER are the data set name and volser, respectively, of the active DASD control file, or asterisks (*), if the active control file is a virtual control file. MASTER is the name of the current master system that is maintaining the virtual control file, or asterisks, if the current control file is on DASD. This information is similar to that displayed in the MIM0039 message generated by the DISPLAY MIM IO command.

Virtual Control File (CTC) Performance Report (VF)

```
 (1)               (2)                                                                                  (3)
    2004/07/13 14:07   CA MIM Virtual Control File Performance Report                                    PAGE      1
 (4)System:     SYSB               (5)Jobname:  MIMGR       (6)Release:   11.6    (7)Service Level:  0000
 (8)From:       2003/10/25 00:00   (9)Each:     RECORD
(10)To:         2004/03/12 10:52 (11)Shift:     00:00 24:00
                   (13)       (15)    (16)    (17)      (18)     (19)     (20)     (21)    (22)
     (12)         Sample (14) System   Reserve   Reserve    Physical   Physical   Logical    Logical   Error  (23)
  Interval Start  Count  DEV    Name     Time      Count    Blocks Rd  Blocks Wr  Blocks Rd  Blocks Wr  Count Message
 ---------------- ------ ---- -------- --------- ---------- ---------- ---------- ---------- ---------- ----- -----------------
 2004/03/07 16:58     15 9010 SYSA       .005387      3,970      3,970      3,970      5,664      5,610     0  Master=SYSA
                         C210 SYSA       .000000          0          0          0          0          0     0
                         C310 SYSA       .000000          0          0          0          0          0     0
```

Field	Description
(1)	Date and time the report was created.
(2)	Report title.
(3)	Report page number.
(4)	System name of the CA MIM image as defined by the DEFSYS statement.
(5)	Name of the CA MIM started task.
(6)	The CA MIM release number.
(7)	The CA MIM service level.
(8)	Date and timestamp of the earliest record in the reporting interval.
(9)	Summary level for the data line. In this example, each RECORD is displayed in the detail line. If each HOUR was specified, then the detail line would be a summary of the hour's activity.
(10)	Date and timestamp of the latest record in the recording interval.
(11)	Range of records included in the report.
INTERVAL START(12)	Start date and timestamp that the report line interval data covers.
SAMPLE COUNT (13)	The total number of data samples taken during the reporting interval.
DEV (14)	Device address of the CTC device, or 'XCF' (which represents a logic path to the external system using XCF communications). This value is similar to the DEV column of the MIM0176 message that is generated by the DISPLAY PATH command.

Field	Description
SYSTEM NAME (15)	External system name, as defined by a DEFSYS statement, to which the CTC device or logical XCF path connects the local system. This value is similar to the SYSTEM column of the MIM0176 message that is generated by the DISPLAY PATH command.
RESERVE TIME (16)	The average elapsed time, in seconds, that the master system took to process virtual reserve request for the specified path represented by the CTC device during the reporting interval. This time covers the time from when the master system receives a request from the client and in turn sends the control file to this client. This value is similar to the RSVDELAY column of the MIM0176 message that is generated by the DISPLAY PATH command.
RESERVE COUNT (17)	The total number of virtual reserve requests received over the specified CTC path during the reporting interval.
PHYSICAL BLKS RD (18)	The total number of buffers (of a size up to the value specified by the MIMINIT VCFBUFFERSIZE parameter) that were received over the specified CTC path during the reporting interval. This value is similar to the READ column of the MIM0176 message that is generated by the DISPLAY PATH command.
PHYSICAL BLKS WR (19)	The total number of buffers (of a size up to the value specified by the MIMINIT VCFBUFFERSIZE parameter) that were transmitted over the specified CTC path during the reporting interval. This value is similar to the WRITTEN column of the MIM0176 message that is generated by the DISPLAY PATH command.
LOGICAL BLKS RD (20)	The total number of buffers (of a size up to the current BLKSIZE) that were received over the specified CTC path during the reporting interval. As many logical blocks of size BLKSIZE that will fit into a transfer buffer of size VCFBUFFERSIZE can be transmitted or received with a single I/O operation.
LOGICAL BLKS WR (21)	The total number of buffers (of a size up to the current BLKSIZE) that were transmitted over the specified CTC path during the reporting interval. As many logical blocks of size BLKSIZE that will fit into a transfer buffer of size VCFBUFFERSIZE can be transmitted or received with a single I/O operation.
ERROR COUNT (22)	The number of errors that occurred on the specified CTC path during the reporting interval. This value is similar to the ERR column of the MIM0176 message that is generated by the DISPLAY PATH command.
MASTER (23)	The name of the master system at the end of the reporting interval.

Appendix A: CA MIM Health Checks

This Appendix describes health checks for C*A MIM*. The product owner for all CA MIM health checks is CA_MIM.

This section contains the following topics:

MIM_GDIF_PROCESS_ALLSYS

Description

Checks to see if GDIF is running in PROCESS=ALLSYSTEMS mode. This is a one-time check that runs when GDIF is started. This is a medium-severity health check.

Best Practice

When using ENQs/RESERVES for data integrity, you must manually enter the QNAME when running in PROCESS=SELECT mode, which may result in possible data integrity exposures. In ALLSYSTEMS mode, GDIF manages the ENQs/RESERVES. Setting GDIINIT PROCESS=ALLSYSTEMS results in much lower maintenance requirements and the elimination of multi-systems data integrity exposures. Contact CA Technical Support to request help with converting to PROCESS=ALLSYSTEMS mode.

Parameters accepted

None

Debug Support

No.

Verbose Support

No.

Reference

See the section GDIF Processing Modes in the *CA MII Programming Guide*.

Message

See the *Message and Code Reference Guide*.

MIM_GDIF_ACF2_NOSMC

Description

Checks to see if the CA ACF2 logon ID record of the CA MIM address space has the NO-SMC attribute set. This is a one-time check that runs when GDIF is started. This is a high-severity health check.

Best Practice

Add the NO-SMC attribute on the CA ACF2 logon ID record for the CA MIM address space. This will ensure that concurrent security requests in the CA MIM address space, which require access to the CA ACF2 databases, do not get single threaded and result in possible system suspends.

Parameters accepted

No.

Debug Support

No.

Verbose Support

No.

Reference

See Step 13 in the *CA ACF2 Installation Guide*.

Message

See the *Message and Code Reference Guide*.

MIM_GDIF_CF_SZ

Description

A given instance of GDIF (CA MII) monitors ENQ activity that it has been configured to manage. This ENQ activity is communicated to other instances of CA MIM by transferring information through a commonly accessed control file. If a given instance of GDIF tries to notify another instance of CA MIM about recently managed ENQ workload, and the currently allocated primary control file is too small to contain the information required to do so, the transfer fails.

This health check determines how much control file space is required to communicate the maximum managed ENQ activity to another instance of CA MIM. It does so by internally monitoring the amount of managed ENQ activity (threshold value) and, on an hourly basis, comparing that level with the size of each allocated control file. If the threshold value exceeds the size of any of the allocated control file, a health check exception event is raised.

Best Practice

Make sure all allocated control files for a given CA MIM system are large enough to permit a global copy of the largest working ENQ workload occurring on that CA MIM system.

Parameters accepted

None

Debug Support

No

Verbose Support

No

Reference

See the section Control File Size Considerations (see page 61) in this guide.

Message

See the *Message and Code Reference Guide*.

MIM_DRVR_CF_STATUS

Description

A given instance of CA MIM usually has one or more secondary or alternate control files in addition to a primary control file. If a file I/O error or other unusual event occurs on the primary control file, CA MIM switches to a secondary control file, assuming one is defined. If there are no other usable secondary control files defined and another error occurs, a product or system outage may occur.

This control file status health check periodically checks defined control files to determine if all are usable (can be written and read by CA MIM), and raises an exception if any allocated control file is not usable.

Best Practice

Monitor all allocated control files and ensure that all control files can be read and written to by CA MIM. This health check will alert you if one or more files is not usable.

Parameters accepted

None

Debug Support

No

Verbose Support

No

Reference

See the sections Allocate DASD Control Files (see page 56) and Performance Considerations (see page 121) in this guide.

Message

See the *Message and Code Reference Guide*.

MIM_GDIF_RESTART_MNGR

Description

The MIM_GDIF_RESTART_MNGR health check verifies that GDIF is started with the Restart Manager feature active. This heath check is a one-time check that occurs during an initialization. The health check severity is medium.

Best Practice

If the MII address space running GDIF terminates, integrity exposures can occur. The Restart Manager feature automatically restarts MIM in the event of unexpected termination. During a restart, global integrity is preserved. We recommend as a best practice that the Restart Manager feature is active when using GDIF.

Parameters accepted

None

Debug Support

No

Verbose Support

No

Reference

For more information, see ‘Restart Manager Usage’ in the Installation Guide.

Message

See the *Message and Code Reference Guide*.

Appendix B: Integration with CA OPS/MVS

This section contains the following topics:

Overview

CA MIM provides seamless integration with CA OPS/MVS in several areas, including product state management, health check status management, and individual automation event management.

You do not need to do anything for CA MIM to enable the product interface to CA OPS/MVS and you do not need to issue any CA MIM initialization statement or commands to activate the interface. If CA MIM and CA OPS/MVS are active in the same z/OS image, CA MIM automatically communicates CA MIMautomation events to CA OPS/MVS.

Before the CA MIM-to-CA OPS/MVS interface existed, CA OPS/MVS could automate CA MIM events through console messages, and CA OPS/MVS can still use CA MIM console message traffic for automation events. Console messages are processed by CA OPS/MVS)MSG rules. Depending upon the particular automation event, CA OPS/MVS rules may need to correlate console messages and, in some instances, issue commands and interrogate command responses to collect information about a given event.

On the other hand, the CA MIM-to-CA OPS/MVS interface is an integrated two-way communication mechanism. CA MIM passes automation event information directly to CA OPS/MVS, including all pertinent data related to the automation event in the form of OPS REXX variables. These events can be automated using a CA OPS/MVS)API rule. This interface is more efficient and reduces the complexity of CA OPS/MVS automation rules as the event data in readily available in REXX variables. This internal interface also eliminates the dependence on the format and content of console message text.

Also, CA MIM can take action on an automation event as directed by a CA OPS/MVS)API automation rule. CA MIM and CA OPS/MVS seamlessly work together to automate the management of shared-system resources.

Enable CA OPS/MVS Event Notification

To enable the interface from CA OPS/MVS, set the CA OPS/MVS parameter APIACTIVE to ON. This allows CA OPS/MVS to acknowledge and process the events generated by CA MIM.

API Rules

For each automation event presented by CA MIM, 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. Some variable names are common, regardless of the automation event while other variable names are specific to the automation event.

Note: For more API information, see the *CA OPS/MVS AOF Rules User Guide*.

Important! Unless otherwise noted, all of the variables are a data type of character and are read-only variables; therefore, a CA OPS/MVS rule program cannot store data into these variables. An attempt to alter the content of read-only variables will result in a CA OPS/MVS rule program failure.

CA MIM Product State Management

CA MIM provides a direct interface to the CA OPS/MVS System State Manager (SSM) application to notify CA OPS/MVS of the current operating state of the given CA MIM address space. The CA OPS/MVS SSM application can use this information to automatically control the operation of the CA MIM address space, as well as any other address space that is dependent upon the CA MIM address space being active.

The CA MIM product active state is presented to CA OPS/MVS and can be processed by the following rule:

```
)API CASTATE
```

The available OPS/REXX variables for CA MIM product state management are:

OPS/REXX Variable	Value
API.APPLICATION	MiMgr
API.VERSION	Current release
API.LEVEL	Service Level
API.EVENTID	CASTATE
API.MSGID	CASTATE
API.TEXT	State of MiMgr is state

The API.TEXT variable has the following format:

`State of` ***`appl_id`*** `is` ***`current_state`***'

appl_id

Specifies the same value as the API.APPLICATION variable

current_state

STARTING

CA MIM is in the process of initializing

UP

CA MIM is active and is managing global resource activity

STOPPING

CA MIM is in the processing of terminating

DOWN

CA MIM is in the process of exiting the system

Member SSMAPI of HLQ.OPS.STATEMAN.RULES demonstrates how to use the CASTATE API.

Note: For more information, see the CA OPS/MVS documentation.

CA MIM Health Check State Management

CA MIM provides a continuous health check state directly to CA OPS/MVS. CA OPS/MVS can use this information in several ways to determine the operational health of the CA MIM address space.

CA MIM issues a heartbeat update every sixty seconds that notifies CA OPS/MVS of the current operational health state of the CA MIM address space. If CA MIM detects a health state change, it immediately generates a heart beat update without waiting for the sixty second heart beat interval to expire. In this way, CA MIM provides CA OPS/MVS with a constant operational health state view of the CA MIM address space.

CA OPS/MVS can also react to the lack of a heart beat update from the CA MIM address space and an indication that there is either a potential problem with the CA MIM address space, or there is a larger system level problem that is taking place.

Note: For more information on using the CA OPS/MVS Health Check application, see the CA OPS/MVS documentation.

The CA MIM product health check heart beat state is presented to CA OPS/MVS and can be processed by the following rule:

```
)API CAHEARTBT
```

The available OPS/REXX variables for CA MIM state management are:

OPS/REXX Variable	Value
API.APPLICATION	MiMgr
API.VERSION	Current release
API.LEVEL	Service level
API.EVENTID	CAHEARTBT
API.MSGID	CAHEARTBT
API.TEXT	Health check message text

The API.TEXT variable has the following format:

appl_id Status: *health_state* Reason: *reason_text*

appl_id

Specifies the value of the API.APPLICATION variable.

health_state

health_state can be one of the following:

NORMAL

CA MIM is operating normally, without any detected problems.

WARNING

CA MIM is operating, but has detected a potential problem that might impact normal processing.

PROBLEM

CA MIM is operating, but has detected a problem that is impacting the processing of global resource activity.

reason_text

reason_text can be one of the following:

Active

CA MIM is operating normally, without any detected problems.

Synchronizing

CA MIM is in the process of joining a synchronized MIMPLEX with other instances of CA MIM operating on external systems. CA MIM should only appear in this state a minimal amount of time.

CF Lockout

CA MIM has detected a control file lock out condition. CA MIM cannot process any global resource activity on the local system.

Migrating

CA MIM has detected a migration is in progress to an alternate control file of VCF MASTER system. CA MIM should only appear in this state a minimal amount of time CA MIM cannot process any global resource activity on the local system until the control file migration successfully completes.

Initializing

CA MIM is in the process of performing initialization processing, in preparation to attempt to join a synchronized MIMPLEX.

Pending Synchronization

CA MIM has completed initialization processing and is now waiting to begin the synchronization process with external MIMPLEX systems. CA MIM should only appear in this state a minimal amount of time.

Quiescing

CA MIM is in the process of quiescing in response to a CA MIM QUIESCE command.

Quiesced

CA MIM has quiesced processing and is now dormant in response to a CA MIM QUIESCE command. CA MIM remains in this state until it receives a CA MIMM RESTART command.

Stopping

CA MIM is in the process of shutting down.

Unknown

The reason for the health state cannot be determined.

Members APIHRTB1, APIHRTB2, and APIHRTB3 of HLQ.OPS.SAMPLE.RULES demonstrate how you use the CAHEARTBT API.

Glossary

ACL

ACL is the abbreviation for automatic cartridge loader, a feature available on IBM 3480-type tape devices.

ACL processing

ACL processing is special processing that z/OS performs for IBM 3480-type devices in which the automatic cartridge loader (ACL) feature is installed and active. When the ACL feature of a device is installed and active, z/OS tries to use that device for only nonspecific, non-temporary volume requests.

action message

An *action message* is a message that is assigned a descriptor code of 1 (system failure messages), 2 (immediate action messages), 3 (eventual action messages), or 11 (critical eventual action messages). Action messages with descriptor codes of 1, 2, or 11 notify you of important events and are displayed on consoles as highlighted, non-deletable messages. Also see *descriptor code*.

active ACL status

An *active ACL status* is a type of device status that you can assign through the CA MIA VARY command. You can assign active ACL status only to IBM 3480-type devices on which the ACL feature is installed. When you change the ACL status to active, z/OS performs ACL processing on that device during allocation.

active status

See active ACL status.

address routine

An *address routine* is an optional routine in the CA MIA application program interface. This routine retrieves the Unit Control Block (UCB) names of all devices on the local system. A sample routine named API1SM02 is provided in the CAI.CBTDSAMP data set.

alias name

See system alias.

allocation recovery

Allocation recovery is the part of the z/OS device allocation process that a job enters when it cannot allocate a suitable online device. During allocation recovery, z/OS determines how to handle that job (cancel it, make it wait for a device to become available, or vary a suitable offline device online). Based on this information, z/OS issues messages IEF238D (and sometimes other additional messages) to tell you what your options are. You can use the SETOPTION command, the CA MIA VARY command, and the TPCRECXT exit routine to influence allocation recovery when TPCF is running.

alternate path support

Alternate path support is a z/VM feature that lets you define alternate channels from a real processor to a DASD or to a tape device to improve performance.

ASCB CHAP

ASCB CHAP is a z/OS function that GDIF uses to change the dispatching priority of a user address space during critical ENQ processing.

assignable tape devices

Assignable tape devices refers to the following types of devices: 3480, 3490, 3590, and any other type of compatible tape device.

attribute verification

An *attribute verification* is an EDIF feature that you can use to prevent programs from making inappropriate changes to the attributes of a data set. During attribute verification, EDIF determines whether a non-exempted program will change the attributes of a data set. If so, then EDIF records the violation. EDIF also abends the program to prevent the update if the ABEND option is in effect. You can enable this feature by specifying the ATTRIBUTES option on an EDIF processing statement.

attribute violation

An *attribute violation* is an update in which the attributes of the data sets will be changed by the program if the update takes place. The EDIF attribute verification feature lets you detect attribute violations and prevent attribute changes from taking place.

authority level

An *authority level* is an assigned code that determines which cross-system commands a target console can execute. You can assign an authority level to a target console through the AUTHORITY parameter on the LINK command. However, the LINK command does not change the z/OS authority level for that console. z/OS authority levels determine which local commands a console can execute.

authorization statement

See the glossary definition for *LMP codes*.

AUTOFREE

AUTORFREE is an ECMF feature that automatically frees certain data sets that were dynamically allocated by TSO users and are currently marked not in use.

broadcast

Broadcast is the process in which GCMF sends cross-system messages to a group of consoles based on criteria defined in a collection set.

CALServ

CA-L-Serv is a component of CA Common Services for z/OS that allows CA MIC to route commands and messages across systems, as an alternative to using CTC or DASD control files. CA-L-Serv uses VTAM to communicate information among systems. The Intersystem Communication Facility (ICMF) must be active before you can use CA-L-Serv.

checkpoint files

Checkpoint files are non-shared DASD files used to track system status information when using CTCONLY or XCF communication methods. They are also required for the REQUEUE feature of the ECMF facility within CA-MII to track job status information when CA-MIM is shutdown and then restarted.

collection set

A *collection set* is a message routing definition that allows a local console, product, TSO user, or system log to receive messages from one or more external systems. Collection sets are created through the COLLECT command.

command

A *command* is a line of text that establishes elements of the CA MIM operating environment that you can change while it is running. You can issue commands from the MIMCMNDS or MIMSYNCH member of the parameter data set, from a console, or from a TSO session.

command alias

A *command alias* is a site-defined alias that you can substitute for a CA MIM command, using the DEFALIAS command. For example, if you define the alias DS for the DISPLAY SYSTEMS command, you can specify DS when you want to see status information about the systems in your complex, instead of specifying DISPLAY SYSTEMS.

command prefix character

A *command prefix character* is a character that you can use to prefix CA MIM commands. Assign this character using the CMDPREFIX parameter on the SETOPTION command.

command routing path

A *command routing path* is a path between a local console, product, or TSO user and one or more external systems. This routing path enables that console, product, or TSO user to issue commands to the external system and to receive the cross-system responses to those commands. Command routing paths are created through the LINK command. Command routing paths are also known as linkages.

command source

Command source is the product, TSO users, or consoles authorized to issue cross-system commands through a linkage.

communication method

You must select a *communication method* to allow CA MIM to share important information between systems. The methods available for use are DASDONLY, CTCONLY, CTCDASD, XCF, ICMF, and NONE.

conflict

A *conflict* is a situation in which two or more tasks need access to the same resource at the same time and cannot share access to that resource.

console pool

A *console pool* is a group of consoles from which GCMF can allocate a console to execute a cross-system command. GCMF uses the pool only for dedicated and shared linkages. Consoles in this pool are known as console pool members.

console pool member

A *console pool member* is a console in the GCMF console pool. Inactive MCS consoles and subsystem consoles may be members of the console pool.

control file

A *control file* is a data set that CA MIM uses to communicate information among systems, to store checkpoint data, or both. When CTCDASD or CTCONLY are active, CA MIM uses a *virtual control file* to communicate information. DASD control files are used for DASDONLY communication and as a backup control file for CTCDASD. You also can define backup (or alternate) DASD control files, which are used if the current control file becomes unusable.

coupling facility

The *coupling facility* is a combination of hardware and software technologies that provide z/OS components and subsystems with the ability to share and transport data among systems in a sysplex.

CP

CP is an abbreviation for Control Program, the portion of the z/VM operating system that manages real resources (such as real memory) and I/O operations to tape devices, printers, and so on.

CPU

CPU is an abbreviation for central processing unit, the portion of a computer that controls overall activity and fetches, decodes, and executes instructions.

CPU image

See *image*.

CTC device

A *CTC device* is a Channel-to-Channel Adapter or IBM 3088-type device that is physically connecting two systems.

CTC master system

See master system.

CTC path

A *CTC path* is a logical path connecting two systems. You need to define CTC paths when you are using the CTCDASD or CTCONLY communication methods so you can transmit cross-system information across CTCAs or IBM 3088-type devices. CTC paths do not connect systems physically; you need to use CTCAs or IBM 3088-type devices to connect systems physically.

CTCA

CTCA is the abbreviation for Channel-to-Channel Adapter, a device that you can use to connect two systems physically.

DASD

DASD is an abbreviation for direct access storage device.

data set control block

A *data set control block* is a data set label that describes the DASD space and attributes for a data set on a DASD device.

DCB

DCB is an abbreviation for data control block, an interface between an executing program and a data set.

deadly embrace

A *deadly embrace* is an unresolvable set of conflicts in which tasks on different systems lock each other out of the devices or resources that the tasks need to complete. For example, if task 1 controls device 1 and needs device 2, and task 2 controls device 2 and needs device 1, neither task can complete. You must cancel these tasks to end the conflict.

dedicated device

A *dedicated device* is a device that has been given dedicated status through the CA MIA VARY command.

dedicated linkage

A *dedicated linkage* is a linkage in which a single console pool member is used as the target console. Any other linkages cannot use that console pool member. Specifying POOL=DEDICATE on the LINK command creates dedicated linkages.

dedicated resource

A *dedicated resource* is a data set or device that is reserved for a particular system, program, function, or user.

dedicated status

Dedicated status is a type of device status you can assign through the CA MIA VARY command. Dedicated status identifies a device that can be allocated only on one system, unless no other suitable device is available. A locally dedicated device is a device that is dedicated to the local system (that is, the system you are currently on). An externally dedicated device is one that is dedicated to a different system.

descriptor code

A *descriptor code* is code used by MCS to route a class of messages to consoles. Descriptor codes identify types of messages; for example, system failure messages (descriptor code 1), immediate action messages (descriptor code 2), eventual action messages (descriptor code 3), or critical eventual action messages (descriptor code 11). These messages provide you with information about system status, situations that require the attention of an operator, and so on. Although many messages are assigned descriptor codes, some messages do not have them. You can use the z/OS CONTROL V,LEVEL command to select messages for each MCS console, based on the descriptor code assigned to those messages.

destination

The *destination* is the local consoles, TSO users, product, or system log that is receiving cross-system commands gathered by a collection set.

device control list

A *device control list* is a series of entries that provide CA MIA with information about devices that it should be managing. See MIMUNITS member.

device control member

The *device control member* is the optional member that provides CA MIA with the local and global names of the devices that it should be managing. This member is identified through the DEVLIST parameter on the MIMINIT statement. By default, CA MIA uses the member named MIMUNITS.

device group

A *device group* is a set of devices created by z/OS during the system generation process. z/OS systems use device groups in creating the eligible device list, from which it selects a suitable device for allocation. Each device group contains one device.

device preference value

A *device preference value* is a type of device status you can assign through the CA MIA VARY command. TPCF uses these values to preference a device from a group of otherwise equally acceptable devices. The higher the value you assign, the more preferenced the device.

DISP=SHR verification

DISP=SHR verification is an EDIF feature that you can use to prevent programs from updating a data set when DISP=SHR is specified in the JCL of the job. The value DISP=SHR does not prevent other programs from updating the data set at the same time. When you enable this feature, EDIF examines the JCL for the programs you designate when one of those programs tries to update the data set. If DISP=SHR is specified, then EDIF records the violation. EDIF also abends the program to prevent the update if the ABEND option is in effect. You can enable this feature by specifying the CHECKEXCLUSIVE parameter on an EDIF processing statement.

display command

A *display command* is a command that you can use to obtain information about resources, devices, messages, systems, and so on. You can issue these commands from the CA MIM parameter data set, from a console, or from a TSO session.

display panel

A *display panel* is an optional panel in the CA MIA application program interface. This panel shows you status information for the managed devices you name on a retrieval panel. The name of the display panel is API1PNL2.

display routine

A *display routine* is an optional routine in the CA MIA application program interface. This routine enables you to display information about tape devices on an ISPF screen. The name of this routine is API1SM01.

DSCB

DSCB is the abbreviation for data set control block.

dual allocation

A *dual allocation* is a situation in which two or more jobs concurrently allocate the same device.

ECMF

ECMF is an acronym for the ENQ Conflict Management Facility, which is available with the CA MII component. ECMF issues messages when a resource conflict occurs. ECMF also resolves certain types of conflicts automatically.

EDIF

EDIF is an acronym for the Enhanced Data Set Integrity Facility, which is available with the CA MII component. EDIF detects and prevents common user errors that can compromise data set integrity.

EDIF processing list

An *EDIF processing list* is a series of EDIF processing statements that tell EDIF what types of data set damage to detect or prevent. A sample processing list is provided in member EDIPARMS in the CAI.CBTDPARM data set. The member that contains this list is called the EDIPARMS member throughout the CA MIM documentation.

EDIF processing statement

An *EDIF processing statement* is a statement that tells EDIF what types of data set damage to detect or prevent for a group of data sets or for a single data set. The DEFAULT, DSORG, PREFIX, SUFFIX, PATTERN, and DATASET statements are all EDIF processing statements. The UTILITY statement is a special type of EDIF processing statement identifying programs that should be processed the same way.

EDIPARMS member

The *EDIPARMS member* is a member of the parameter data set that contains EDIF processing statements, which tell EDIF what types of data set damage to detect and prevent. Collectively, the contents of this member are called the EDIF processing list. A sample member called EDIPARMS is provided in the CAI.CBTDPARM data set.

EDL

EDL is the abbreviation for eligible device list. Also see eligible device list.

EDT

EDT is the abbreviation for eligible device table, a table of device addresses used during device allocation. Also see eligible device table.

eligible device list (EDL)

An *eligible device list (EDL)* is a list of devices that z/OS builds for a device allocation request. The eligible device list contains the unit control block address of every device with the correct physical characteristics for that request. The eligible device list consists of one or more device groups that contain these device addresses.

eligible device table (EDT)

An *eligible device table (EDT*) is a list of devices from which z/OS chooses during device allocation. The eligible device table contains the unit control block address of each device known to the local system. When a job requests a device, z/OS extracts the addresses of the devices with the correct physical characteristics for that request. This list of device addresses is known as the eligible device list.

elimination logic

Elimination logic is the logic in the TPCEDLXT exit routine and job reserve processing that TPCF uses to eliminate unwanted devices before z/OS allocation processing begins. TPCF uses this logic to remove devices from the z/OS eligible device list during allocation; TPCF also uses this logic to eliminate devices from the z/OS offline device list during allocation recovery.

ENQ facility

The *ENQ facility* is a z/OS facility that serializes access to resources in a system. The ENQ facility establishes a queue of tasks waiting to use a resource, and the system control program manages this queue. A task (or z/OS) issues an ENQ request to use this facility. Also see ENQ request.

ENQ request

An *ENQ request* is a request that uses the z/OS ENQ facility to serialize access to resources that will be used by multiple tasks on the same system.
A task can request shared or exclusive access to a resource. Shared access usually means that the task does not update the resource; therefore, other tasks can share access to that resource at the same time. Exclusive access usually means that the task will update the resource so that other tasks cannot share access to that resource at the same time without risking an integrity exposure.

A three-part naming convention is used on an ENQ request: the major name (or QNAME) of the class of resources, the minor name (or RNAME) of the specific resource that is needed, and the scope, which indicates whether the task will serialize access in its address space, in the system, or among systems. Also see QNAME and RNAME.

esoteric group

An *esoteric group* is a site-defined group of devices. Esoteric groups are created through the UNITNAME macro. Esoteric groups have a critical effect on the device groups z/OS creates for an allocation request.

exclusion parameters

Exclusion parameters are parameters on the COLLECT command that tell GCMF to exclude certain messages that otherwise would be collected by that collection set. You can use exclusion parameters to collect only a subset of any message category. For example, you can collect all messages issued by a job, except a message you name on an exclusion parameter. A collection set never collects messages you specify on an exclusion parameter, even if another parameter in that collection set selects those messages. Also see inclusion parameters.

exclusive linkage

An *exclusive link* is a linkage in which you specifically assign a console as the target console. Specifying the TGTCONS parameter on a LINK command creates exclusive linkages. Also see target console.

exempt list

An *exempt list* is a series of statements that provide GDIF with supplemental and more specific information on how to process ENQ and RESERVE requests and hardware reserves for resources. A sample exempt list is provided in the member called GDIEXMPT in the CAI.CBTDPARM data set. The member that contains the exempt list is called the GDIEXMPT member.

exit routine

An *exit routine* is a site-defined program that is called at a predetermined time during processing. You can use exit routines to change the way CA MIM and its facilities handle commands, messages, ENQ and RESERVE requests, device allocations, and resource conflicts.

externally dedicated device

An *externally dedicated device* is a device that has been given dedicated status on another system through the CA MIA VARY command. Also see dedicated status.

externally reserved device

An *externally reserved device* is a device that has been given reserved status on another system through the CA MIA VARY command. Also see reserved status.

GCMF

GCMF is an acronym for the Global Command and Message Facility, which is available with the CA MIC component. GCMF enables you to issue cross-system commands and to collect messages from other systems.

GCMF console pool

See *console pool*.

GDIEXMPT member

The *GDIEXMPT member* is a member of the parameter data set containing statements that give GDIF supplemental information on handling ENQ and RESERVE requests and hardware reserves for specific resources. GDIF uses this member for requests only if EXEMPT=YES is specified on a QNAME statement for that class of resources. A sample member called GDIEXMPT is provided in the CAI.CBTDPARM data set. Collectively, the contents of the GDIEXMPT member are called the exempt list.

GDIF

GDIF is an acronym for the Global Data Integrity Facility, which is available with the CA MII component. GDIF ensures integrity for shared resources by converting RESERVE requests to global ENQ requests and by propagating ENQ requests to all systems in a complex.

generic group

A *generic group* is a group of physically identical devices. z/OS creates generic groups during the system generation process. Generic groups have a critical effect on the device groups z/OS create for an allocation request.

global

The term *global* applies to all resources or processors in a shared-device, multiprocessor, or multi-image environment.

global device name

See *global name*.

global ENQ request

A *global ENQ request* is an ENQ or RESERVE request that GDIF has propagated to all systems.

global name

A *global name* is a unique name that can be used by all systems when referring to the same tape device. A global name can be three or four characters in length and may be alphanumeric or numeric. Global names are critical when a device has different unit control block names on different systems. CA MIA uses the unit control block name as the global name for a device unless you specifically assign a global name through the MIMUNITS member of the CA MIM parameter data set.

GTAF

GTAF is an acronym for the Global Tape Allocation Facility, which is available with the CA MIA component. GTAF enables you to share tape devices among systems.

hardware reserve

A *hardware reserve* is an I/O instruction that dedicates a DASD to a single processor to serialize access to one of the resources on that device. On z/OS systems, hardware reserves are produced through RESERVE requests. Also see RESERVE request.

I/O

I/O is an abbreviation for input/output.

ICMF

ICMF is an acronym for the Intersystem Communication Facility, which is part of the CA MIC component. ICMF enables you to route cross-system commands and messages through the CA-L-Serv communication product, rather than through CTCs or shared DASD.

image

An *image* is a logical/physical partition of a CPU that functions as a separate processing unit. A single CPU can be divided into multiple images, each operating independently and each running under a different operating system. z/VM running on an image can create further images.

inaccessible device

An *inaccessible device* is a device to which there is no physical access path, logical access path, or both.

inclusion parameters

Inclusion parameters are parameters on the COLLECT command that tell GCMF which messages to collect. Also see exclusion parameters.

index number

See *system index number*.

ineligible device

An *ineligible device* is a device that is unavailable for allocation to the requesting job.

initialization

Initialization is a process in which CA MIM reads the startup information you have provided and stores that information for use during operations.

initialization parameters

Initialization parameters are parameters that define elements of the CA MIM operating environment that cannot be modified while it is running. You can specify initialization parameters only on initialization statements. Also see initialization statements.

initialization statements

Initialization statements are statements that define elements of the CA MIM operating environment that cannot be modified while it is running. Initialization statements are identified by the suffix INIT. Some facilities have their own initialization statements, and the initialization statement named MIMINIT influences all facilities.

initialization values

Initialization values are elements of the CA MIM operating environment that are defined through initialization statements. You cannot change these values while CA MIM is running.

initiator

An *initiator* is a z/OS started task that selects jobs for execution based upon job class, scheduling priority, and other performance criteria.

integrity exposure

Integrity exposure is a situation in which access to a resource has not been serialized, which makes it possible for that resource to be damaged.

JCL

JCL is an abbreviation for job control language, the language used to describe the resource and execution requirements of a job to the operating system.

linkage

A *linkage* is a cross-system routing path that allows a console, product, or TSO user on the local system to issue commands to one or more external systems. Linkages also enable the issuing console, product, or TSO user to receive the cross-system responses to these commands. Linkages, which also are known as command routing paths, are created through the LINK command.

LMP codes

LMP codes are needed to license CA MIM for use at your site. These codes are placed in the KEYS member in the OPTLIB data set in the CAS9 JCL procedure.

local device name

See local name.

local name

The *local name* is a name obtained from the unit control block address for a tape device. A device may have different local names on different systems. For example, a device can have the local name 1A0 on one system and 2A0 on another system. The CA MIA application program interface and CA MIA display commands use local names when referring to devices; however, most CA MIA processing and commands use global names when referring to devices.

locally dedicated device

A *locally dedicated device* is a device that has been given dedicated status on the local system through the CA MIA VARY command. Also see dedicated status.

locally reserved device

A device that has been given reserved status on the local system through the CA MIA VARY command. Also see reserved status.

locking mechanism

A *locking mechanism* is a bit-mask that z/OS uses to serialize access to a tape device group.

managed device

A *managed device* is a tape device that CA MIA is managing. You can tell CA MIA to manage a device by specifying the local name of that device in a device control list. You also can tell CA MIA to manage an entire class of devices, such as all tape devices, by specifying the appropriate value on the DEVCLASS parameter on the MIMINIT statement.

managed resource

A *managed resource* is a resource that GDIF, ECMF, or both are managing. You can tell GDIF, ECMF, or both to manage a resource by specifying the QNAME for that resource in your QNAME list.

master system

A *master system* is a designated system that manages the virtual control file when the CTCDASD or CTCONLY communication methods are active. This system must be connected physically to all other systems in your complex through Channel-to-Channel Adapters or IBM 3088-type devices. This system also must be connected logically to all other systems through CTCPATH statements.

MCS

MCS is an abbreviation for multiple console support, the portion of the z/OS operating system that controls consoles and message traffic to consoles.

message ID

A *message ID* is a character string that identifies a message. Also called a message prefix.

message routing definition

A *message routing definition* is a rule that tells GCMF which messages to collect, the systems from which to collect these messages, and the local product, TSO user, console, or system log that is to receive these messages. Message routing definitions also are known as collection sets. You can create message routing definitions through the COLLECT command.

message type

A *message type* is a classification that tells GCMF which messages to collect. You can enter a simple list of message types to collect a broad group of messages; however, you also can collect a subset of any message type.

migration

Migration is a process in which CA MIM suspends all requests for control file services and shifts cross-system communication to another control file.

MIMCMNDS member

The *MIMCMNDS member* is a member of the parameter data set containing CA MIM commands that should be executed during the initialization process. A sample member called MIMCMNDS is provided in the CAI.CBTDPARM data set.

MIMINIT member

The *MIMINIT member* is a member of the parameter data set that contains initialization statements for CA MIM and its facilities. A sample member called MIMINIT is provided in the CAI.CBTDPARM data set.

MIMMSGS member

The *MIMMSGS member* is the message member for CA MIM. MIMMSGS member contains CA MIM Message Facility statements that are processed as part of CA MIM initialization. This member is the primary message table and points to all other message tables used by CA MIM. A sample MIMMSGS member is provided in the CAI.CBTDMENU data set.

MIMPARMS data set

The *MIMPARMS data set* is the parameter data set for CA MIM. This data set contains members that provide the statements and commands that should be executed at startup time. The MIMPARMS data set is identified through the //MIMPARMS DD statement in the startup procedure. A sample MIMPARMS data set is provided in the CAI.CBTDPARM data set.

MIMplex

A *MIMplex* is the collection of all systems supervised by CA MIM.

MIMQNAME member

The *MIMQNAME member* is the member of the parameter data set containing statements that tells GDIF and ECMF how to handle ENQ and RESERVE requests for classes of resources. A sample member called MIMQNAME is provided in the CAI.CBTDPARM data set. Collectively, the contents of the MIMQNAME member are called the QNAME list.

MIMSYNCH member

The *MIMSYNCH member* is the member of the parameter data set containing commands that should be executed at the end of the system synchronization process. The commands in this member can be CA MIM, z/OS, or JES commands. A sample member called MIMSYNCH is provided in the CAI.CBTDPARM data set.

MIMTRC data set

The MIMTRC data set is the data sets that collect trace data about commands, command output, ENQ and RESERVE requests, resource conflicts, and so on.

MIMUNITS member

The *MIMUNITS member* is the member of the parameter data set containing the names of the devices that CA MIA should manage. This member also is called the device control member. Collectively, the contents of the MIMUNITS member are called the device control list.

minidisk

A *minidisk* is a logical, addressable unit of storage on a physical device. A minidisk can be an entire device (called a full-pack minidisk) or a subsection of a device.

monitor type

A *monitor type* is a code used by MCS to route a functionally related group of monitor messages to specified consoles. A monitor type is assigned to all monitor messages and some non-monitor messages; monitor types also can be assigned to consoles. MCS matches the monitor type on the message with the monitor type on the console when routing local messages. You can use monitor types to tell GCMF which messages to collect; you can also send messages to consoles based on monitor types.

multi-line message

A *multi-line message* is a group of messages that can be displayed out-of-line in a predefined display area or displayed inline as a group on a console.

multiple console support

Multiple console support is a portion of the z/OS operating system that controls consoles and message traffic to consoles. Abbreviated as MCS.

non-specific linkage

A *not-specific linkage* is a linkage that uses a member of the console pool to execute cross-system commands. This console pool is designated through the POOL parameter on the LINK command.

non-specific volume request

A *not-specific volume request* is a tape volume request in which no particular volume is requested. Nonspecific volume requests are made by specifying DISP=(NEW) in the JCL of a job.

non-temporary volume request

A *not-temporary volume request* is a tape volume request in which the data set on that volume is saved at the end of the step. Non-temporary volume requests are made by specifying the values DISP=(KEEP), DISP=(CATLG), or DISP=(PASS) in the JCL of a job.

not-available device

A *not-available device* is a device that has been given not-available status through the CA MIA VARY command. Also see not-available status.

not-available status

A *not-available status* is a type of device status you can assign through the CA MIA VARY command. Not-available status identifies a device that should not be selected for allocation unless no other device is available.

offline device list

An *offline device list* is a list of offline devices that z/OS allocation creates for a job that cannot allocate a suitable online device. The offline device list is created for jobs in allocation recovery. This list contains the unit control block name of every offline device with the correct physical characteristics for that request.

operating values

Operating values are elements of the CA MIM operating environment that are defined through commands. You can establish these operating values at startup time, and you can change operating values at any time.

overgenned device

An *overgenned device* is a device that has been given overgenned status through the CA MIA VARY command. Also see overgenned status.

overgenned status

O*vergenned status* is a type of device status you can assign through the CA MIA VARY command. Overgenned status identifies a tape unit control block address for which there is no physical device. TPCF makes z/OS ignore overgenned devices when z/OS is selecting a device for allocation.

parallel sysplex

A *parallel sysplex* is a sysplex running in a supported IBM z/OS complex, which utilizes the 976x processor and coupling facility hardware. Also see sysplex.

parameter data set

A *parameter data set* is the data set identified through the //MIMPARMS DD statement in the CA MIM startup procedure. This data set contains required and optional members that provide initialization and operating values.

PDF

PDF is the abbreviation for Program Development Facility or Portable Document Format (Adobe).

PDS

PDS is the abbreviation for partitioned data set.

preference logic

Preference logic is the logic in the TPCSRMXT exit routine and in job reserve and VARY PREF processing that TPCF uses to eliminate unwanted devices after z/OS allocation has eliminated unavailable devices. TPCF uses this logic to remove devices from the candidate list z/OS creates during allocation. Preference logic tells TPCF which devices you prefer to use whenever possible. TPCF never examines preference logic during allocation recovery.

preference value

See device preference value.

processing options

Processing options are options specified on EDIF processing statements. These options determine how EDIF processes a group of data sets or a single data set.

pseudo-volume serial number

A *pseudo-volume serial number* is a value that CA MIA generates and propagates to identify the system on which a device is allocated. This value appears in the format ss=GTA, where ss is the system alias for the allocating system. z/OS displays this number only when a device is allocated on an external system.

PTF

PTF is the abbreviation for program temporary fix.

QNAME

A *QNAME* is an eight-byte name that identifies a class of resources (such as data sets, control blocks, and so on) on an ENQ or RESERVE request. A QNAME also may indicate what function a task will perform with that class of resources.

QNAME list

A *QNAME list* is a series of statements that tell GDIF and ECMF how to process ENQ and RESERVE requests for classes of resources. A sample QNAME list is provided in the member called MIMQNAME in the CAI.CBTDPARM data set. The member that contains the QNAME list is called the MIMQNAME member.

read verification

Read verification is an EDIF feature that you can use to prevent unauthorized programs from reading a data set. During read verification, EDIF verifies that a program is authorized to read the data set. If it is not, then EDIF records the violation. EDIF also abends the program to prevent the read operation if the ABEND option is in effect. You can enable this feature by specifying the ACCESSCHECK option on an EDIF processing statement.

real reserve/release processing

Real reserve/release processing is a DASD hardware feature that serializes access among real processors by dedicating a device to one processor at a time. Real reserve/release processing is requested through a reserve channel command word, which is produced when a task or z/OS issues a RESERVE request for a resource. The hardware is released through a release channel command word.

release CCW

A *release CCW* is a channel command word that releases a device or minidisk that had been dedicated to a processor or guest system. Also see real reserve/release processing, virtual reserve/release processing.

REQUEUE feature

The *REQUEUE feature* is an ECMF feature that places a batch job on hold and returns it to the input job queue when that job requests an unavailable resource. When the requested resource becomes available, ECMF frees the job so that it can execute. The REQUEUE feature affects only batch jobs that issue requests on which the QNAME SYSDSN is specified and works only at job start.

reserve CCW

A *reserve CCW* is a channel command word that serializes access to a device or minidisk. Also see real reserve/release processing, virtual reserve/release processing.

RESERVE facility

The *RESERVE facility* is a z/OS facility that serializes access to a resource by dedicating the DASD volume on which that resource resides to the system on which the requesting task is executing. Tasks issue RESERVE requests to use this facility. Also see RESERVE request.

RESERVE request

A *RESERVE request* is a special type of ENQ request that z/OS uses to serialize access to resources that will be shared by multiple systems.
A RESERVE request contains the UCB address of the DASD on which the requested resource resides. When the task obtains access to this resource, z/OS issues an I/O instruction that dedicates the DASD to the system on which the task is executing. This I/O instruction is known as a hardware reserve.

reserved device

A *reserved device* is a device that can be allocated only by a certain job or group of jobs. You can reserve devices through the CA MIA VARY command. Also see reserved status.

reserved status

A *reserved status* is a type of device status you can assign through the CA MIA VARY command. Reserved status identifies a device that can be allocated only by a certain job or by a group of jobs.

resource

A *resource* is any part of a computer system (such as a CPU, a data set, software, and so on) that a job or task requires.

resource conflict

See conflict.

retrieval panel

A *retrieval panel* is an optional panel in the CA MIA application program interface. This panel shows you the unit control block name of all local devices and accepts the names of the devices for which you want information. A sample retrieval panel is provided in member API1PNL1 in the CAI.CBTDPENU data.

RNAME

An *RNAME* is a 1- to 255-byte name that indicates what specific resource the task that issued an ENQ or RESERVE request needs.

routing code

A *routing code* is a code MCS uses to route a functionally related group of messages to the appropriate consoles, TSO sessions, logs, and so on. Routing codes are assigned to many, but not all messages; routing codes are also assigned to consoles and TSO users. To determine where to route messages, MCS matches the routing code on the message with the routing code assigned to a console or TSO user.
You can use routing codes to tell GCMF which messages to collect. You also can send messages to consoles based on routing codes. EDIF uses routing codes to determine which consoles and TSO sessions should receive messages about update violations, read violations, attribute violations, and data set conflicts.

routing definition

See message routing definition.

routing path

See command routing path.

serialization

Serialization is a process that controls access to resources to ensure resource integrity. Tasks can perform serialization themselves, or they can invoke the z/OS ENQ or RESERVE facilities to perform serialization. Also see ENQ facility and RESERVE facility.

service cycle

A *service cycle* is a designated length of time that CA MIM waits before accessing its control file automatically. The length of a service cycle is the product of the values for the INTERVAL and CYCLES parameters on the SETOPTION command.

service interval

A *service interval* is a designated length of time that CA MIM waits before querying its global facilities (that is, GDIF, GTAF, and GCMF) for cross-system transactions. If one or more of these facilities have transactions, then CA MIM accesses its control file at this time. The length of a service interval is set through the INTERVAL parameter on the SETOPTION command.

shared linkage

A *shared linkage* is a linkage in which the console pool member that serves as the target console can be used by other linkages as needed. Shared linkages are created by specifying POOL=SHARE on a LINK command.

significant characters

Significant characters are non-wildcard characters in a QNAME/RNAME string or in a job name. Blanks also are considered significant characters. GDIF uses significant characters to determine the order in which to examine LOCAL and GLOBAL statements that you have specified in your exempt list.

SMF

SMF is an abbreviation for the IBM System Management Facilities.

source system

A *source system* is an external system from which you are collecting messages through a collection set.

specific linkage

A *specific linkage* is a linkage that uses a designated console to execute cross-system commands. This console is assigned through the UCMID parameter on the LINK command.

specific volume request

A *specific volume request* is a tape volume request in which a certain data set is requested.

statement

A *statement* is a line of text that establishes elements of the CA MIM operating environment that you cannot change while it is running. You can specify statements only in members of the CA MIM parameter data set. Each of these statements is read during the product initialization process.

subsystem console

A *subsystem console* is a console that has been defined to the system-one for which there is no physical device or device address. Subsystem consoles are defined during the system generation process or in a member of the SYS1.PARMLIB data set, depending on what version of z/OS you are running. GCMF allocates these consoles to execute cross-system commands issued through dedicated and shared linkages.

synchronization

Synchronization is the process in which the systems in a complex establish contact with each other and obtain the most current information about the activities of each other. CA MIM accomplishes synchronization through its control files.

sysplex

A *sysplex* is an IBM strategy for providing a single-image view of a multiple-image complex. IBM sysplex initiatives are included in MVS/ESA SP 5.2.0 and above.

system

A *system* is a logical/physical partition of a CPU that functions as a separate processing unit. A system can be a separate CPU or a unique operating system. Also see image.

system alias

A *system alias* is a unique one- to two-character name that identifies a system to CA MIM. You can define system aliases through the DEFSYS statement. If you do not define aliases for your systems, then CA MIM uses the index number of a system as its alias.

system ID

A *system ID* is a unique character string that identifies a system to CA MIM. These are the three types of system IDs accepted: system index numbers, system names, and system aliases. Also see system index number, system name, and system alias.

system index number

A *system index number* is a unique number CA MIM generates the first time it recognizes a system. It uses this number to identify the origin and destination of internal transactions.

system name

A *system name* is a unique one- to eight-character name that identifies a system to CA MIM. You can assign system names through the DEFSYS statement. If you do not specifically assign a system name to a system, then CA MIM uses the SMF ID of the system as its name.

target console

A *target console* is the console that is executing cross-system commands issued through a linkage. A target console is the recipient of a command; it is not the console from which the cross-system command was issued.

target system

A *target system* is the external system that is receiving and executing cross-system commands issued through a linkage.

temporary volume request

A *temporary volume request* is a tape volume request in which the data set will not be saved at the end of the step.

TPCF

TPCF is an acronym for the Tape Preferencing and Control Facility, which is available with the CA MIA component. TPCF lets you influence device selection during the device allocation process.

trace

A *trace* is a CA MIM function that maintains a log of time-stamped information.

trace data set

A trace data set is the data set that you can use to collect trace data commands. For example, command output, ENQ and RESERVE requests, and resource conflicts commands. This data set is named MIMTRC. Also see MIMTRC data set.

TSO

TSO is an abbreviation for time-sharing option, which is the component of z/OS that allows users to create and maintain programs and data sets, run jobs, view output displays, and perform other functions online from a terminal.

TSO user ID

A *TSO user ID* is a unique character string that identifies a TSO user.

UCB

UCB is an abbreviation for unit control block, from which the name, address, and status of a device are obtained.

UCMID

A *UCMID* is a unique number assigned to a console that identifies that console. MCS uses the UCMID to indicate where a command originated from, to route messages to specific consoles, and to route command responses to the appropriate console when you append the z/OS L parameter to a command.

update violation

An *update violation* is an update that can result in damage to a data set. This update may be performed by a program that is not authorized to update the data set, by a job that specifies DISP=SHR in its JCL, or by programs that use different naming conventions for the same data set.

utility verification

Utility verification is an EDIF feature that you can use to prevent inappropriate programs from updating a data set. During utility verification, EDIF determines whether a program is authorized to update the data set. If it is not, then EDIF records the violation, and also abends the program to prevent the update if the ABEND option is in effect. You can enable this feature by specifying the UTILITY option on an EDIF processing statement.

virtual control file

A *virtual control file* is an area in memory that CA MIM uses to communicate information among systems when CTCDASD or CTCONLY are active. CA MIM directs transactions to this file. The system that manages this file is known as the master system.

virtual reserve/release processing

Virtual reserve/release processing is a z/VM feature that serializes access among guests running under the same z/VM operating system. Virtual reserve/release processing dedicates a specified minidisk to a single guest at a time.

volser

Volser is an abbreviation for volume serial number.

VSAM

VSAM is an abbreviation for virtual storage access method. VSAM stores fixed- or variable-length records on a direct access storage device in sequences determined by record-specific key fields, by record number, or by the order in which records were entered.

VTOC

VTOC is an abbreviation for volume table of contents.

wait-eligible device

A *wait-eligible device* is a device that is currently allocated, but will be available to the requesting job after the current user has de-allocated the device.

wildcard characters

Wildcard characters are special characters that let you match a character string (such as a QNAME, RNAME, job name, pattern, or message ID) with another character string that has only some of the same characters. For example, you can use a wildcard character to match a character string that you provide with any job name that has the same prefix.

XCF communication

XCF communication is a cross-system communication method using the IBM coupling facility hardware in a sysplex environment. CA MIM uses this method to communicate control file transactions among systems in a MIMplex

z/OS

z/OS is an operating system for IBM mainframe computers. z/OS is a renamed, repackaged, and enhanced version of the OS/390 operating system.

z/OS ENQ facility

See ENQ facility.

z/OS RESERVE facility

See RESERVE facility.

z/VM

z/VM is a generic term for the z/VM and VM/ESA operating systems.

Index

T

U

V

W

Z