CA MIM™ Resource Sharing for z/VM

Programming Guide
Release 12.0
This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2014 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.
CA Technologies Product References

This document references the following CA Technologies products:

- CA 1® Tape Management
- CA MIA Tape Sharing for z/VM (CA MIA)
- CA MIC Message Sharing for z/VM (CA MIC)
- CA MIM™ Resource Sharing for z/VM (CA MIM)
- CA VM:Manager

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

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

**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.

- This improved guide details the latest CA MIM programming standards.
- Added the Hyperstar (see page 31) feature.
# Contents

## Chapter 1: Introduction

- Intended Audience ................................................................. 9
- Introduction to CA MIM .............................................................. 9
- Components and Facilities ......................................................... 9

## Chapter 2: Planning Initial Settings

- Files Used During Operations .................................................. 11
- General Statements ................................................................. 13
- Assigning Command Authorization to Users ................................. 14
  - Specifying the OPTION Statement ........................................... 14
  - Assigning Classes to User IDs .................................................. 15
- Tailoring the Automatic Commands Files .................................... 16
  - Specifying Commands ........................................................... 16
  - Defining a Command Alias ..................................................... 18
    - Displaying Valid Aliases ....................................................... 19
- Changing the Block Size for the Control Files .............................. 19
- Displaying Signon Messages at Startup ...................................... 20
- Displaying Messages About Commands Executed at Startup .......... 20
- Setting Command Timeout ....................................................... 21
- Putting an Inoperative System to Sleep .................................... 21
- Customizing Messages ............................................................ 21
  - Displaying Messages in Uppercase .......................................... 21
  - Default Message Tables ....................................................... 22
  - Creating a Custom Message Table .......................................... 24
  - Naming Convention for Messages .......................................... 26
  - Message Facility Statements and Commands ............................ 27

## Chapter 3: Communication Methods

- Physical Control File Communication Methods ............................ 30
- DASD Control File Communication ........................................... 30
- Virtual Control File Communication Methods ............................. 31
- Hyperstar .................................................................................. 31
- Initial Communication Methods Overview ................................ 32
- Allocating DASD Control Files .................................................. 34
  - Where to Locate Your DASD Control Files ................................. 34
- Configuring DASD Control Files ............................................... 35
Chapter 4: Starting and Stopping CA MIM

Startup Activities .......................................................... 61
  CA MIA Startup Activities ............................................. 62
How You Override Initialization Values at Startup Time .................. 62
  The FORMAT Override Parameter ...................................... 62
Product Termination Considerations ..................................... 64
  Global Shutdown Processing ........................................... 64
  Local Shutdown Processing ............................................ 66
How You Shut Down the System in a VCF Environment .................. 68
  How You Free Inactive Systems ....................................... 69

Chapter 5: Advanced Topics ............................................. 71

How You Activate Features of New Releases Dynamically ................. 72
How You Use the Dynamic Compatibility Feature with Checkpoint Files ............................................. 74
How You Activate New Features with ACTIVATE FEATURE ................ 75
How You Use ACTIVATE FEATURE with Checkpoint Files ............... 76
How to DEACTIVATE a FEATURE ......................................... 76
Performance Considerations ............................................... 77
CA MIM Driver ................................................................. 77
Control File Internals ......................................................... 78
  Global Management Record (GMR) ...................................... 78
Chapter 6: Troubleshooting

How You Resolve Problems ........................................................................................................... 97
Diagnostic Procedures ..................................................................................................................... 98
How You Respond to Control File Access Delays ......................................................................... 99
  Message MIM0061W ..................................................................................................................... 100
  Message MIM0062W ..................................................................................................................... 103
  Message MIM0063W ..................................................................................................................... 103
  Message MIM0100A ..................................................................................................................... 103
  Message MIM0200W ..................................................................................................................... 107
How You Activate Tracing .............................................................................................................. 110
How You Obtain Dumps ................................................................................................................ 110
Product Releases and Maintenance ............................................................................................... 110

Chapter 7: User Exits

Using Communication Words in Exit Routines ............................................................................. 111
  Passing the Parameter List to All Routines ................................................................................... 112
Managing User Exits ...................................................................................................................... 113
  Controlling Exit Routines Through Commands ........................................................................ 114
Common Exit Interface: MIMCMDXT ............................................................................................ 114
  Coding Rules .............................................................................................................................. 115
  Issuing Messages from MIMCMDXT ......................................................................................... 118
MIMINIXT Exit Routine .................................................................................................................. 119
  Coding Rules for the MIMINIXT Exit Routine .......................................................................... 119

Chapter 8: Maintaining LMP Keys

CALMP KEYS File ............................................................................................................................ 121
Chapter 1: Introduction

This section contains the following topics:

- **Intended Audience** (see page 9)
- **Introduction to CA MIM** (see page 9)

**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 two 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 that are 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 Preferences and Control Facility (TPCF)**
  
  Improves device processing by permitting tape device allocation requests by model and influencing device selection.
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 component improves operations productivity and performance, and ensures that all systems are monitored consistently.

CA MIC consists of the following facility:

Global Command and Message Facility (GCMF)

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

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
- 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.

CA MIM lets you define backup communication methods. This capability furnishes data centers with the redundancy which guarantees 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 described 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:

- **Files Used During Operations** (see page 11)
- **General Statements** (see page 13)
- **Assigning Command Authorization to Users** (see page 14)
- **Tailoring the Automatic Commands Files** (see page 16)
- **Defining a Command Alias** (see page 18)
- **Changing the Block Size for the Control Files** (see page 19)
- **Displaying Signon Messages at Startup** (see page 20)
- **Displaying Messages About Commands Executed at Startup** (see page 20)
- **Setting Command Timeout** (see page 21)
- **Putting an Inoperative System to Sleep** (see page 21)
- **Customizing Messages** (see page 21)

### Files Used During Operations

CA MIM uses the following files to establish options, parameters, and other initialization and operation information:

**AUTHUSER MIM A (DDNAME=AUTHUSER)**

This optional file contains information pertaining to users and their assigned authorization classes. For more information, see [Assigning Command Authorization to Users](#) (see page 14).

**CLASS MIM A (DDNAME=MIMCLASS)**

This optional file contains definitions for user-defined tape drive classes. For more information, see the [CA MIA Tape Sharing for z/VM Programming Guide](#).

**CMNDS MIM A (DDNAME=MIMCMNDS)**

This optional file contains commands that are issued during the initialization process. For more information, see [Format for CMNDS MIM File](#) (see page 17).

**DDNAMES MIM A (DDNAME=MIMNAMES)**

This file defines the location of DASD control files and checkpoint files. For more information, see the installation step for supplying control file and checkpoint file information in "Overview of Concepts and Installation" in the [CA MIM Installation Guide](#).

**CALMP KEYS Z (DDNAME=CALMPKEY)**

This file contains your product LMP license key, which CA supplies. The file may contain one or more LMP key statements for one or more CA z/VM products. For more information, see the installation step for entering LMP keys in the [CA MIM Installation Guide](#).
**Files Used During Operations**

**EDIMSGS MIM A (EDIMSGS)**

This parameter is a placeholder file. This parameter is required to be present, but does not affect product operation.

**ICMMSGS MIM A (ICMMSGS)**

This parameter is a placeholder file. This parameter is required to be present, but does not affect product operation.

**INIT MIM A (DDNAME=MIMINIT)**

This file provides product initialization values. Statements in the INIT MIM file let you define names and aliases for systems in your complex, set values for initialization parameters, and disable commands. For more information, see the specific functions that are described throughout this chapter.

**LINKAUTH MIM A (DDNAME=MIMLINKA)**

This file is used to establish authority for cross-system CP commands that are executed on your system. You can prevent specific commands from being sent to your system from any external user, and you can limit commands to external users with certain authority levels.

**MIAMSGS MIM A (MIAMSGS)**

This file contains templates that are used by the Allocation component to construct messages. You can alter the text in this file to change the text that is sent to users when the message is invoked.

**MICMSGS MIM A (MICMSGS)**

This file contains templates that are used by the Console component to construct messages. You can alter the text in this file to change the text that is sent to users when the message is invoked.

**MIIMSGS MIM A (MIIMSGS)**

This parameter is a placeholder file. This parameter is required to be present, but does not affect product operation.

**MIMMSGS MIM A (MIMMSGS)**

This file contains templates that are used by the CA MIM driver to construct messages. You can alter the text in this file to change the text that is sent to users when the message is invoked.

**SYNCH MIM A (DDNAME=MIMSYNCH)**

This optional file contains CP commands and CA MIM commands issued when CA MIM completes its synchronization process. For more information, see **Format for SYNCH MIM File** (see page 17).

**TRACE MIM A (DDNAME=MIMTRACE)**

(Optional) This file stores information about product activities. This file is created when the trace feature is activated. This file is not supplied on the distribution tape.
CHAPTER 2: PLANNING INITIAL SETTINGS

UNITS MIM A (DDNAME=MIMUNITS)

This file provides device control information. This file is used to accomplish the following tasks:

■ Place certain tape devices under CA MIA control.
■ Relate different real addresses on several systems to the same tape device.
■ Establish global names for tape devices that are controlled by CA MIA.
■ Define tape devices as managed by Autopath for z/VM.

For any of the previously listed files, you can change the default file name to another name by issuing a FILEDEF command before starting CA MIM. Associate the given ddname with a fixed-length 80-character-record CMS file in the FILEDEF command.

Example:

To reference a UNITS MIM file on disk C instead of the default file on disk A, include the following statement in your PROFILE GCS file:

FILEDEF MIMUNITS DISK UNITS MIM C

General Statements

The following general statements can be used in any of the CA MIM initialization or command file:

■ IFSYS
■ ENDIF
■ INCLUDE
■ LOG
■ NOLOG

Note: For more information, see the CA MIM Statement and Command Reference Guide.
Assigning Command Authorization to Users

(Optional) The AUTHUSER MIM file allows you to assign privilege classes to user IDs. Each privilege class grants a user ID the authorization to issue a certain set of commands. This file has an option that indicates how this product assigns a privilege class to a user ID that is not listed in the file.

When no AUTHUSER MIM file exists, each user ID is assigned the general USER class by default.

Specifying the OPTION Statement

To assign a privilege class when a user ID is not defined in the file, specify the Option statement at the beginning of the AUTHUSER MIM file. Place this statement as the first uncommented line in the file. Also the Option statement is listed before any user IDs that appear in the file. The option takes effect only for user IDs that have not been assigned a privilege class in the file.

The OPTION statement has the following format:

```
OPTION option_value
```

The valid `option_value` are:

**EXCL**

Specifies that a user whose user ID does not appear in the file is unauthorized to issue any CA MIM commands.

**INCL**

Specifies that a user whose user ID does not appear in the file be assigned the general USER class.

**INCL(class)**

Specifies that a user whose user ID does not appear in the file is assigned the privilege class that is indicated in parentheses after the INCL keyword. The `class` value can be either OPERATOR or USER.
Assigning Classes to User IDs

To assign a privilege class, specify the following two pieces of information on a line by themselves in the AUTHUSER MIM file:

```
userid class
```

You can specify any of the following privilege classes in the class field:

- **USER**
  General user class (restricted use of commands)

- **OPERATOR**
  Operator class (unrestricted use of commands)

- **OPER**
  Operator class (synonym for OPERATOR)

When a user ID is not assigned a class, this product uses the OPTION statement in the AUTHUSER MIM file to determine the class.

If no options are specified, and a user ID does not appear in the file, the user is assigned the general USER class. For information about commands requiring OPERATOR privileges, see the CA MIM Statement and Command Reference Guide.

**Example: A valid AUTHUSER MIM file**

```
OPTION EXCL

*USERID CLASS
MAINT OPERATOR
OPERATOR OPER
DSIAZ11 USER
DSIVB11 USER
```

**Note:** The AUTHUSER MIM file can also determine which users can issue cross-system commands on the local system. Cross-system commands can be issued from any user ID with USER or OPERATOR privilege, providing there is a cross-system command linkage for that user ID.
Tailoring the Automatic Commands Files

The CMNDS MIM and SYNCH MIM files are optional files that contain commands that are executed automatically at startup. You can use the CMNDS MIM file to execute CA MIM commands at startup before systems synchronize with each other. You can use the SYNCH MIM file to execute CA MIM commands and CP commands after all systems synchronize.

In general, there are no effective differences between issuing CA MIM commands from the CMNDS MIM or SYNCH MIM files. Where differences exist, they are noted in the usage notes for the commands.

**Note:** CP commands can only be issued from the SYNCH MIM file.

Not all CA MIM commands are useful when they are placed in the CMNDS MIM or SYNCH MIM files. The following section briefly summarizes the commands that are best suited for inclusion in these files.

### Specifying Commands

Consider including these commands and statements in the CMNDS MIM, SYNCH MIM, or both files:

- **DEFALIAS**
  - Defines a shorthand notation for CA MIM commands.

- **DISABLE**
  - Disables a CA MIM command.

- **DISPLAY**
  - Displays information about the status and activity of CA MIM. This information is displayed in the console log.

- **LINK**
  - Create, modify, and delete the cross-system command linkages.

- **SETOPTION**
  - Controls the operation of CA MIM.

- **VARY**
  - Changes the default status of tape devices that are managed by CA MIA.

*For detailed information about each of these commands, see CA MIM Resource Sharing for z/VM Statement and Command Reference Guide.*
Format for CMNDS MIM File

When you code commands in the CMNDS MIM file, follow the format and syntax rules summarized here:

- Begin each command with the command name (such as VARY). This product ignores all leading spaces.
- End each command before column 73. If a command does not fit in columns 1-72 of a line, continue the command on the next line by typing a comma at the end of the current line and then typing the rest of the command on the next line.
- Place an asterisk (*) in column 1 to indicate that a line is a comment and not a command.
- Place a slash and then an asterisk (/*) before a comment written on a command line to distinguish the comment from the command. The command must precede the /*. Otherwise, the command is treated as a comment.
- Separate parameters with a comma or space.

A sample of the CMNDS MIM file is provided on the CA MIM distribution tape. You can either edit this sample file for your environment or create a new file. If you do not use the default file name, CMNDS MIM A, be sure to issue an appropriate FILEDEF command before starting CA MIM.

Format for SYNCH MIM File

When you code commands in the SYNCH MIM file, follow the format and syntax rules summarized here:

- Begin each CA MIM command with the "MI" command prefix. This product ignores all leading spaces.
- Follow the prefix with the command name (such as, DISPLAY) and any command parameters.
- You do not need to use a special prefix for CP commands.
- End each command before column 73. If a command does not fit in columns 1-72 of a line, continue the command on the next line by typing a comma at the end of the current line and then typing the rest of the command on the next line.
- Place an asterisk (*) in column 1 to indicate that a line is a comment and not a command.
- Place a slash and then an asterisk (/*) before a comment written on a command line to distinguish the comment from the command. The command must precede the /*. Otherwise, the command is treated as a comment.
- Separate parameters with a comma or space.
A sample SYNCH MIM file is provided on the CA MIM distribution tape. You can either edit this sample file for your environment or create a new file. If you do not use the default file name SYNCH MIM A, be sure to issue an appropriate FILEDEF command before starting CA MIM.

**Defining a Command Alias**

A command alias is a string of text that you can specify in place of a CA MIM command. You can use command aliases to:

- Issue a shorter version of a command. For example, in place of a display command that has several parameters.
- Issue a command in a way that is easier to remember, or more consistent with commands that you already know.

Command aliases are defined through the DEFALIAS command. To define a command alias, provide the text (or alias) you want to use in place of this command and the name of the command that is being assigned this alias. You also can specify any additional parameters that should be executed when this command is executed. Because the DEFALIAS command has positional parameters, specify this information in exactly this order.

For example, suppose that you wish to establish an alias for DISPLAY MIM OPTIONS. You could issue this shortened form of the DEFALIAS command to define a command alias of DMMO:

```
DEFA DMMO DISPLAY MIM OPTIONS
```

Specify DMMO as if you issued the DISPLAY MIM OPTIONS command. You can append more parameters to a command alias. Using the last example, you can append the COMMANDS parameter to the DMMO command alias as follows:

```
DMMO COMMANDS
```

This tells CA MIM to execute the command DISPLAY MIM OPTIONS COMMANDS.

**Note:** Use the command name, rather than the alias, when looking up information about that command or when contacting CA Technical Support. Also, you may need to redefine aliases when you are upgrading to a new release of this product because of changes in command format or function.
Displaying Valid Aliases

To display a list of valid aliases, issue this command:

```
DISPLAY COMMANDS
```

The following display shows the CA MIM commands and command aliases. Specifically, message MIM0053I displays the alias information.

<table>
<thead>
<tr>
<th>DISPLAY COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM0056 COMMANDS AVAILABLE FOR MIM</td>
</tr>
<tr>
<td>ACTIVATE,ATTACH,AUTHCHK,CANCEL,CTC,DEFALIAS,DETACH,DISABLE,DISPLAY,DOM,DUMP,FREE,GOMDISP,GOMDUMP,GOMOPT,GCS,GLOBALVALUE,GTDISP,GTAOPT,LINK,MIGRATE,MIMDISP,MIMDUMP,MIMOPT,MSGTABLE,QUERY,QUIESCE,REMOVE,RESTART,RESYNCH,ROUTE,SETOPTION,SHUTDOWN,SYSDUMP,TIMEOUT,TPCDISP,TPCOPT,USERDATA,VARY,VARYINT,VCF</td>
</tr>
<tr>
<td>MIM0053 MIM COMMAND ALIAS DISPLAY:</td>
</tr>
<tr>
<td>ALIAS SUBSTITUTION TEXT</td>
</tr>
<tr>
<td>DFA DISPLAY FACILITIES</td>
</tr>
<tr>
<td>DCF DISPLAY FILES</td>
</tr>
</tbody>
</table>

Changing the Block Size for the Control Files

CA MIM dynamically adjusts the block size that is used to access its control files. For a DASD file that has already been formatted, MIM gets the block size from the file itself. For a DASD file that must be formatted, CA MIM determines an optimal block size based on the device type and the size of the file. The actual block size is never less than the value specified through MIMINIT BLKSIZE=, nor greater than 32760.

To change the minimum block size, specify a new value for the BLKSIZE parameter on a MIMINIT statement. For example, you can change the block size to 12288 bytes by specifying this MIMINIT statement in the INIT MIM file:

```
MIMINIT BLKSIZE=12288
```

To see the actual block size for the active control file, you can issue the ‘DISPLAY MIM IO’ command.

**Note:** In some environments, larger BLKSIZE values can improve system performance by reducing the number of discrete read/write operations to the CA MIM control file. This especially is true, when message traffic is extremely heavy for the CA MIC component of CA MIM.
Displaying Signon Messages at Startup

The default value of SIGNON=YES specifies that CA MIM facilities issue signon messages when they initialize. To specify that facilities do not issue signon messages, change the signon value to NO.

For example, you can tell CA MIM not to display these messages by specifying this MIMINIT statement in the INIT MIM file:

```
MIMINIT SIGNON=NO
```

To display these messages the next time you restart, specify SIGNON=YES.

A typical signon message follows:

```
MIM2001 Global Tape Allocation Facility/GTAF
Copyright (c) 2014 CA. All rights reserved.
```

Displaying Messages About Commands Executed at Startup

This product displays a message whenever it executes a command that is specified in the CMNDS MIM or SYNCH MIM files. If you do not want to see routine messages, specify SUPPRESSRESP=YES on the MIMINIT statement. YES is the default for SUPPRESSRESP.

For example, to suppress what could be a substantial number of routine messages, you could specify this MIMINIT statement:

```
MIMINIT SUPPRESSRESP=YES
```

To display the messages again, specify SUPPRESSRESP=NO.

The SUPPRESSRESP parameter does not stop this product from issuing messages when a command in one of the files cannot be executed. For example, because of a syntax error.
Setting Command Timeout

To set a command timeout value for your CA MIM commands using the SETOPTION CMDTIMEOUT command. The command timeout value represents the maximum amount of time that can elapse between the start and end of the command processing. This product forces the command to end and issues the error message MIM0244E. The default setting for the SETOPTION CMDTIMEOUT is 60 seconds.

You can disable the command timeout by issuing the command SETOPTION CMDTIMEOUT=NONE.

Putting an Inoperative System to Sleep

This product allows you to set a time limit for how long a system can receive MIM0061W messages. These messages indicate that the system is inoperative. After this time expires, the system is put to sleep. You can use the SETOPTION HIBERNATE command to set this time limit. The default time period is three minutes.

Once a system is in the sleep state, it either joins the complex when it is operative again, or you can FREE the system and restart it later. For more information about the HIBERNATE parameter, see the CA MIM Statement and Command Reference Guide.

Customizing Messages

The product message facility allows you to customize message text that is issued by CA MIM. This facility can only be used to customize CA MIM messages. Only the messages that are found in the default message tables can be customized.

The message facility also has the ability to provide non-English versions of CA MIM messages. While this product does not provide default, non-English message tables, you can create customized non-English message tables.

Displaying Messages in Uppercase

By default, messages that are issued by the CA MIM message facility are displayed in mixed case (both uppercase and lowercase) characters. To see messages in uppercase characters only, specify this statement:

MIMINIT MSGTEXT=UPPERCASE

Your specification takes effect as soon as the MIMINIT statement is processed.
Default Message Tables

When CA MIM is installed, default message tables are downloaded as CMS files. The default message table files are named:

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

The default file type is MIM. Store the default message table files on the MIMGR 191 disk. They define the default message attributes. The MIMMSGX file is used to create site-specific message definitions that override the default attributes.
MIMMSGS Message Table

The MIMMSGS file contains message facility statements that are processed as part of the initialization. The MIMMSGS file is the primary message table and points to all other message tables used by CA MIM. The following example shows the statements that are contained in the MIMMSGS file, and the general structure of this file.

NOLOG
TABLE MIMMSGS,LANGUAGE=ENGLISH

*=======================================================================*
*   MIM0001 - MIM0999   DEFINE MIM DRIVER MESSAGES
*=======================================================================*
* The CA MIM Message Facility reads the statements
* in this file during initialization. This file is the
* primary message table and calls all the facility-specific
* MESSAGE TABLES.
*
*=======================================================================*
MSG 'MIM0001W Too many continuations - excess discarded',
  ROUTCDE=(1,2,10),DESC=(4)
MSG 'MIM0002E @1 terminating in initialization',
  ROUTCDE=(1,2,10),DESC=(1)
MSG 'MIM0003E No usable CONTROL files specified',
  ROUTCDE=(1,2,10),DESC=(2)
MSG 'MIM0004W File @1 - JFCB access or OPEN failed',
  ROUTCDE=(1,2,10),DESC=(2)
MSG 'MIM0005E @1 - already active in this system',
  ROUTCDE=(1,2,10),DESC=(2)
MSG 'MIM0006E File @1 - invalid device',
  ROUTCDE=(1,2,10),DESC=(2)
MSG 'MIM0007E File @1 - inadequate space',
  ROUTCDE=(1,2,10),DESC=(2)
MSG 'MIM0014E File @1 - system with ident @2 already ACTIVE',
  ROUTCDE=(1,2,10),DESC=(2)
MSG 'MIM0015E File @1 full - system @2 cannot be added',
  ROUTCDE=(1,2,10),DESC=(2)
MSG 'MIM0016W WARNING - File @1 on non-shared device',
  ROUTCDE=(1,2,10),DESC=(3,7)
MSG 'MIM0017I @1',
  MCSFLAG=HRDCPY
MSG 'MIM0018I Following statements accepted from member @1',
  MCSFLAG=HRDCPY
MSG 'MIM0020I MIM @1 ready - system @2',
  ROUTCDE=(2,11),DESC=(4)
Customizing Messages

The TABLE statement defines the MIMMSGS file as the primary message table from which all other message tables are called. All subsequent MSG statements belong to the defined table. The MSG statement in MIMMSGS defines CA MIM driver messages, which can be customized. The CA MIM facility-specific message tables contain TABLE and the MSG statements unique to that facility. For example, MICMSGS is used for CA MIC.

The INCLUDE statement directs the message facility to locate the indicated message table and insert the statements that are found in the file at the point specified.

Creating a Custom Message Table

The message table file provides access to message text that can be reworded or translated into languages other than English. We recommend not directly editing the default message table. Instead, define the MSG statements in the MIMMSGX file to override the MSG statements found in the default files.

By default, the MIMMSGX file is not called during initialization because the INCLUDE MIMMSGX statement is commented in the MIMMSGS file. Once the INCLUDE MIMMSGX statement is uncommented, the MIMMSGX file is called during the initialization. The MSG statements found therein override the corresponding MSG statements found in the default files.

Follow these guidelines when using the MIMMSGX file:

- Only messages that are found in the default message table files can be modified.
- Message IDs are expected to be eight characters in length. Do not change the length.
- Follow the message ID naming convention for CA MIM messages.
- All message facility syntactical rules are enforced.
- Any language can be used for message text.

Message Table Syntax Rules

The following syntax rules govern the statements that are placed in the message table files. The variable substitution capabilities that are used within the text parameter of the MSG statement are also discussed here.

- 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 which 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 with the text of each message. The "at" sign designates each substitution (@). Each substitution is numbered by @n, where n is a decimal number from 1 through 9. Optionally, special formatting can be requested in the following ways:

- A vertical bar (|) preceding the parameter number (n) indicates that leading blanks are removed.
- A vertical bar (|) following the parameter number (n) indicates that trailing blanks are removed.
- Periods (.) preceding the parameter number (n) indicate that the substituted variable is 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 is 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**

These examples show 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.'
Customizing Messages

Naming Convention for Messages

The standard format suggested for CA MIM messages is:

MIM####

Where #### is a decimal number from 0001 through 9999. t is one of the following letters that are used to designate the message type:

A
   Specifies an action message
E
   Specifies an error message
I
   Specifies an informational message
W
   Specifies a warning message

Within this naming standard, message number ranges are reserved for use by functional areas and facilities. The current ranges for message numbers are listed here:

<table>
<thead>
<tr>
<th>Range</th>
<th>Functional Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM0001 - MIM0999</td>
<td>CA MIM General Messages</td>
</tr>
<tr>
<td>MIM0500 - MIM0599</td>
<td>GCS (VM) Messages</td>
</tr>
<tr>
<td>MIM0600 - MIM0799</td>
<td>CA MIM General Messages</td>
</tr>
<tr>
<td>MIM0800 - MIM0899</td>
<td>Message Facility</td>
</tr>
<tr>
<td>MIM2000 - MIM2999</td>
<td>GTAF and TPCF</td>
</tr>
<tr>
<td>MIM3000 - MIM3999</td>
<td>GCMF</td>
</tr>
<tr>
<td>MIM5000 - MIM5999</td>
<td>Open</td>
</tr>
<tr>
<td>MIM6000 - MIM6999</td>
<td>ICMF</td>
</tr>
<tr>
<td>MIM7000 - MIM7999</td>
<td>Open</td>
</tr>
<tr>
<td>MIM8000 - MIM8399</td>
<td>Used to define the variable text segments substituted in the @n parameters</td>
</tr>
</tbody>
</table>
Message Facility Statements and Commands

This product provides statements and commands that are used exclusively by the message facility. Statements are used within message table files and are referenced during an initialization. Commands can be issued at any time to display or modify current message facility options.

TABLE Statement

The TABLE statement defines the primary message table to be used by the message facility. This table also allows you to indicate the language that is used within the message table. All subsequent MSG statements belong to the defined table. The language name can be any text string; it has no functional merit. The table and the language currently in use are displayed through the DISPLAY MSGTABLE command.

In the MIMMSGS file, the TABLE MIMMSGS,LANGUAGE=ENGLISH statement defines the MIMMSGS file as the primary message table from which all other files are called.

MSG Statement

The MSG statement is used to override the internally defined message attributes. The MSG definitions in the message tables that are shipped with CA MIM match the internally defined, default attributes. Any changes that are made to the message tables are noted during a CA MIM initialization by the MIM0866I and MIM0867I messages.

Only the following MSG statement parameter is valid in this product:

- **text**

  The content of the message you are writing in any language.

INCLUDE Statement

The INCLUDE statement is used in the primary message table to locate other message table files. During an initialization, the MSG statements found in those files are inserted at the point where the INCLUDE statement is defined.

For example, INCLUDE MICMSGS finds the message table that is named, MICMSGS, and inserts its MSG statements into the MIMMSGS file during CA MIM initialization.
DISPLAY MSGTABLE Command

The DISPLAY MSGTABLE command displays the primary message table name and language currently in use. This command also displays the MIM module, which contains the hard-coded default values for all messages that are managed by the message facility. This example illustrates this information:

```
MIM0822I MIM Message Tables
Table Language  Msg Count
MIMDRMFM  ENGLISH.intenal 1215
MIMMSGS   ENGLISH        684
MIIMSGS   ENGLISH        111
MIAMSGS   ENGLISH         77
MICMSGS   ENGLISH        108
EDIMSGS   ENGLISH         21
ICMMMSGS  ENGLISH         21
```

The message table definitions that are contained in the MIMMSGS file override the internal default values that are located in the MIMDRMFM module.

MSGTABLE Command

The MSGTABLE command is used to change the primary message table after an initialization. This product loads the message tables during an initialization. After an initialization, changes to the message table files can be implemented immediately by issuing the MSGTABLE command.
Chapter 3: Communication Methods

This section contains the following topics:

- Physical Control File Communication Methods (see page 30)
- DASD Control File Communication (see page 30)
- Virtual Control File Communication Methods (see page 31)
- Hyperstar (see page 31)
- Initial Communication Methods Overview (see page 32)
- Allocating DASD Control Files (see page 34)
- Configuring DASD Control Files (see page 35)
- Determining Control File Space Requirements (see page 36)
- Creating DASD Control Files (see page 36)
- CTCDDASD as the Initial Communication Method (see page 38)
- CTCONLY as the Initial Communications Method (see page 40)
- NONE as the Communication Method (see page 41)
- How You Define Your VCF Master and Error Recovery Environment (see page 42)
- How You Define CTC Devices for Use in VCF Environments (see page 45)
- Virtual Control File Sizing Considerations (see page 51)
- Configure CTCPATH Statements for Disaster Recovery (see page 52)
- Control and Checkpoint File Considerations (see page 53)

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

- Device allocations
- 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 77) for a better understanding of CA MIM control file internals and externals.
Physical Control File Communication Methods

This product supports the use of physical DASD files communication between systems. The CA MIM control file is allocated on a shared DASD volume. Individual CA MIM systems access the control file to communicate managed resource activity to other systems.

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.
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.

External client systems access the virtual control file through CTC devices that are connected to the VCF master system. The master system itself is a client system, and accesses the virtual control file through memory operations, without any external input/output operation.

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.

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.

This table describes the general requirements and benefits for each of the CA MIM communication methods, which are described in more detail in later sections:

<table>
<thead>
<tr>
<th>Method</th>
<th>Requirements</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>DASDONLY (See note.)</td>
<td>■ Shareable DASD volume accessible by all systems participating in the MIMplex.</td>
<td>■ The MIMplex systems can be any combination of sysplexed and non sysplexed images.</td>
</tr>
<tr>
<td></td>
<td>■ Solid State DASD provides high performance equivalent to CTC communication options.</td>
<td>■ Solid State DASD provides high performance equivalent to CTC communication options.</td>
</tr>
<tr>
<td>CTCONLY</td>
<td>■ ESCON or FICON CTC devices connecting at least one system to all others in the MIMplex.</td>
<td>■ The MIMplex systems can be any combination of sysplexed and non-sysplexed images.</td>
</tr>
<tr>
<td></td>
<td>■ Definition of at least one CA MIM system to manage the virtual control file (VCF).</td>
<td>■ High performance.</td>
</tr>
<tr>
<td></td>
<td>■ Definition of at least one checkpoint file on each system in the MIMplex.</td>
<td>■ Not subject to contention delays associated with DASD IO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Back up master system capability in case of master system failure.</td>
</tr>
</tbody>
</table>
### Initial Communication Methods Overview

#### Chapter 3: Communication Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Requirements</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTCDA<strong>SD</strong></td>
<td>DASD volume accessible by all systems participating in the MIMplex.</td>
<td>The MIMplex systems can be any combination of sysplexed and non sysplexed images. High performance. Backup DASD control file capability in case of CTC failure or if there are no available backup masters. Backup master system capability in a master system failure Not subject to contention delays associated with DASD IO when VCF is in use.</td>
</tr>
<tr>
<td></td>
<td>ESCON or FICON devices connecting at least one system to all others in the MIMplex. Definition of at least one CA MIM system to manage the virtual control file (VCF).</td>
<td></td>
</tr>
<tr>
<td>NONE</td>
<td>■ Single system only</td>
<td>■ Lets you start a CA MIM task on a single system without involving any cross-system communication media.</td>
</tr>
<tr>
<td></td>
<td>■ Generally used in new installations for testing purposes.</td>
<td></td>
</tr>
</tbody>
</table>

Although it is required to 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). At any time, you can dynamically change communication methods, after correctly configuring an alternate method.

**Note:** Using the DASDON**LY** Method

The DASDON**Ly** method uses DASD control files to communicate cross-system information about CA MIM activities. To select this method, specify MIMINIT COMMUNICATION=DASDON**Ly** in the INIT MIM file.

For the DASDON**LY** or CTCDA**SD** methods, allocate at least one DASD control file. You can allocate up to 99 backup control files. We recommend allocating at least one backup DASD control file for use encase the primary DASD control file becomes unusable.
Allocating DASD Control Files

A DASD control file is a data set residing on a DASD that is accessible to all systems running CA MIM.

Note: When using the CTCDA SD communication method, CA MIM directs most transactions to the virtual control file and suppresses I/O operations to the DASD control file. Initializing or migrating transaction are the exception and they are directed to the DASD control file.

Where to Locate Your DASD Control Files

Allocate DASD control files on devices that are accessible to all systems in the installation. Follow these guidelines when deciding where to locate your control files:

- The volume that you choose must not be the target of frequent reserve requests from any other products.
- A CA MIM control file should never reside on the same pack as a:
  - StorageTek HSM control file.
  - CA 1 Tape Management Catalog (TMC).
  - Shared RACF data set.
- Files must reside on a low-contention volume.
- Do not allocate files on a volume containing the spool, the JES checkpoint data set, a catalog, or your page or swap data sets.
- If possible, allocate files in a location that requires minimal (or no) head movement to access.

Allocate backup control files on a different device from the one on which the primary control file resides. A good practice is to allocate backup control files on devices that use a different channel and controller.

We recommend that you allocate a minimum of ten cylinders per file on a cylinder boundary. The cylinders must be contiguous; that is, they must be contained in a single extent.
CA MIM control files must be allocated on a shared DASD device that is accessible to all systems on which CA MIM executes. When a DASD device is shared among systems, remember the following guidelines:

- In z/OS systems, the device must be defined as shared in the SYSGEN IODEVICE macro.
- In z/VM systems, the device must support reserve/release processing. The two types of reserve/release processing are:
  - Real reserve/release processing
    Is needed when CA MIM is running on two or more real processors. That is, two or more systems that are not guests under the same z/VM system.
  - Virtual reserve/release processing
    Is needed when two or more operating systems operate as guests of z/VM.

If you have several guests running under one z/VM operating system, and you have at least two real processors in your installation, use both real and virtual reserve/release processing.

Use these guidelines, to configure DASD for your control files as follows:

- Whenever control files are shared by two or more real processors, specify SHARE YES in the system configuration file or SHARED=YES in the HCPRIO ASSEMBLE file for all RDEVICE macros that you are using to define DASD for control files. SHARED=YES or SHARE YES must be specified on each guest system, even if the first-level z/VM system does not share DASD with any other processor or LPAR.
  
  **Note:** The SHARED=YES or SHARE YES parameter is required to propagate reserve commands to the DASD hardware and upward to the host z/VM system. For details about the SHARED parameter, see Virtual Machine/Extended Architecture System Product: Planning and Administration (IBM publication number GC23-0378) or the equivalent guide for your system.

- When your installation runs two or more operating systems as z/VM guests on the same real processor, specify virtual reserve/release processing for the minidisks that are used as a control file (whether or not other processors are in use). To do so, specify mode MWV on the MDISK statement in the z/VM directory entry for one of the guest operating systems. Then, using mode MW, issue a link command to the same minidisk from each of the other guest operating systems.

  Using virtual reserve/release processing protects guests on the same real processor from each other. Also, hardware reserve still operates correctly when virtual reserve/release is invoked, thus protecting users in separate real processors from each other.
Always specify virtual reserve/release processing for the control file minidisks even if there are no guest operating systems under z/VM. Define the minidisk by specifying mode MWV on the MDISK statement. Reserve/release channel commands are not processed correctly unless virtual reserve/release is specified.

Determining Control File Space Requirements

Allocate each CA MIM control file on a cylinder boundary. We recommend that you allocate ten cylinders. The cylinders must be contiguous. After CA MIM is running, you can check how much allocated space is actually in use by issuing the DISPLAY IO command.

The default block size for a control file is 6144, but CA MIM may adjust this upward, depending on the device type and size of the file. The smallest size is 4096, and the maximum size is 32760, subject to the device-dependent maximum. If, during an initialization, you receive message MIM0025E indicating that a larger block size is required, you may specify a larger block size or you can reduce the amount of space that is allocated for the control file.

Creating DASD Control Files

Control files must be allocated before starting this product. The format of the DASD control file you use, z/OS, CMS, or CMSFP, depends upon what type of systems that are communicating through the control file. Select the appropriate procedure based on your communication requirements.

z/VM Systems Communicating with z/OS

Use z/OS format control files if one or more systems in your MIMplex is a z/OS system. Your z/VM system can, and should, use CMS format for checkpoint files, however z/OS systems cannot access CMS or CMSFP format control files.

A z/VM system cannot allocate z/OS format control files. Control files that are in z/OS format must be allocated using a z/OS allocation method.
Creating DASD Control Files

Chapter 3: Communication Methods

z/VM Systems When No z/OS System Shares the Control File

When there are no z/OS system shares, it is necessary to perform all I/O to control files to a full pack minidisk. z/VM can only propagate reserve/release orders to the real DASD. You can specify CMSFP in the DDNAMES MIM file to request this type of handling. Even though this product addresses a full pack, the control file occupies only a small minidisk area on the pack, leaving the remainder of the pack free for low priority z/VM and CMS uses.

Only one control file or checkpoint file should be allocated to a single full pack. Any pack that contains a CMSFP control file should not be used for critical CP uses, such as paging or spool files, but it is acceptable to allocate user files on such a pack. When you allocate a CMSFP file, reserve space for the file by using an ordinary MDISK statement in the MIMGR user ID definition. For example, you could define control file at MIMGR 391 for six cylinders. Format the file twice, as follows:

```
FORMAT 391 X#1#MIM391
FORMAT 391 X 1 (RECOMP
```

Provide a read/write link to the same file on the full pack from where it was allocated. For example, consider the following MDISK statement:

```
MDISK 391 3380 1242 003 VMPK01 MW
```

Provide the following full pack MDISK statement, either under MIMGR or some other owner. This example is for a triple density 3380.

```
MDISK 591 3380 000 END VMPK01 MWV
```

**Note:** Specify END, instead of the number of cylinders, on the MDISK statement for the full pack minidisk containing the control file. For more information, see the IBM Planning and Administration manual for your z/VM system.

If the MDISK statement for the full pack is owned by another user ID include a link command under MIMGR. The user ID in this example is MAINT 123.

```
LINK MAINT 123 591 MW
```

**Note:** Never define two full pack minidisks addressing the same real pack. Define one, and then link to it as needed for the applications.

Once the minidisk is formatted as previous illustrated, you do not use the 391 address to perform I/O to the disk. Instead, specify the full pack address in the DDNAMES MIM file, and the absolute address, in hexadecimal format, of the first cylinder of the minidisk, as shown. In this example, the pack address is 591.

```
CMSFP MIMTBL01 MIM391 591 4DA
```

The last token, 4DA, is the hex value of the starting cylinder 1242.
The CTCDASD initial communication method is a combination of physical control files and VCF methods.

The MIMplex can consist of sysplexed and non-sysplexed images.

A DASD control file is used as the base communication method during initialization and recovery situations. During initialization, all systems synchronize on the DASD control file. If all VCF communication requirements are met, and INITIAL=CTC is specified, then CA MIM automatically migrates from the DASD 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 ESCON or FICON 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 control file is the base communication medium. Migration to the DASD 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 if INITIAL=CTC.
Implement CTCDASD Communication

To implement the CTCDASD communication method, you customize certain parameters in the INIT MIM file.

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 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 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 control file synchronization, and after any event causing a DASD control file resynchronization.

5. Define checkpoint files for each system using the ALLOCATE statement on z/OS systems and in the DDNAMES MIM file on z/VM systems.

   **Note:** For more information, see the CA MIM Statement and Command Reference Guide.
The CTONLY initial communication method uses a control file that are 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 control file is required.

The CTONLY 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 ESCON or FICON 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 CTONLY 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.
Implement CTCONLY Communication

To implement CTCONLY as the initial communication method, you customize certain parameters in the INIT MIM file.

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 on z/OS systems and in the DDNAMES MIM file on z/VM systems.

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

NONE as the Communication Method

Setting COMMUNICATION=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 CTC devices. Therefore, the Global Tape Allocation Facility (GTAF) cannot be activated. This global facility requires a cross-system communication method.
Implement COMMUNICATION=None

To implement COMMUNICATION=None, you customize certain parameters in the INIT MIM file.

Follow these steps:

1. Define the system using the following DEFSYS statement:
   DEFSYS(SYSA,01,SYSA)
2. Specify NONE as the communications method using the MIMINIT COMMUNICATION parameter.

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 INIT MIM member.
- The GLOBALVALUE statement in the INIT MIM 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 an 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 these 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 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 that is 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 automatically migrates 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 CTC devices
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 ANYELIGIBLE=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.

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 INIT MIM file.
- Make the CTCPATH statements in the INIT MIM file 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:

![Diagram]

```
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 CTONLY 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 ADDRESS=8C4 TOSYSTEM=SYSC
CTCPATH FROMSYSTEM=SYSC ADDRESS=8C4 TOSYSTEM=SYSA
CTCPATH FROMSYSTEM=SYSB ADDRESS=2C8 TOSYSTEM=SYSC
CTCPATH FROMSYSTEM=SYSC ADDRESS=7C8 TOSYSTEM=SYSB
CTCPATH FROMSYSTEM=SYSD ADDRESS=520 TOSYSTEM=SYSC
CTCPATH FROMSYSTEM=SYSC ADDRESS=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 between SYSA and SYSB and between SYSA and SYSD makes SYSA the master eligible connection. For example, the following commands might be used to define the paths:

On SYSA:
CTCPATH ADDR=(8D1)
CTCPATH ADDR=(9F2)

On SYSB:
CTCPATH ADDR=(7D1)

On SYSD:
CTCPATH ADDR=(5F2)
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 (MIMINIT COMMUNICATION=CTCONLY), 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 files are identified by a ddname of MIMTBLnn, where nn is a numeric value from 00 to 99, inclusive. Checkpoint files are identified by a ddname of MIMCKPnn, 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.

Create Checkpoint Files

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

The checkpoint file is especially important when you are using the CTCONLY 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 when 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.

How You Allocate Control and Checkpoint Files

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

Allocate DASD checkpoint files, if you are using the CTCONLY communication method. This environment uses the checkpoint file to track system status information.
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.

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 that are 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 81).
The CA MIM control file must be large enough to accommodate the number of transactions that are generated during these transition states. The amount of space that is 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.

**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 (VCF) use a default transfer buffer of 32,768 bytes to accommodate cross-system transactions. You can change the size of the VCF transfer buffer.

**Follow these steps:**

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 MI 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.

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 when you use the DASDONLY or CTCDASD communication methods. The GMR resides in the checkpoint file when you use the CTCONLY communication method.

A FORMAT startup is required under the following circumstances:

- You are planning to reference new, unformatted DASD control file, 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
  - GTAF
  - 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 can 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 control file is allocated to CA MIM.
- At least one system is master eligible. A master eligible system has 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 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

You can migrate to a DASD control file.

Follow these steps:

1. View the list of available control files by issuing this command:
   
   `DISPLAY FILES`

2. Initiate the migration to a specific file by issuing this command:
   
   `MIGRATE CONTROLFILE=nn`

   - Specifies the control file ID of the DASD control file

You can migrate CA MIM to the next available control file.

Follow this step:

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 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.

You can migrate to a virtual control file.

Follow these steps:
1. View the list of eligible master systems by issuing this command:
   
   DISPLAY GLOBALVALUE

2. Migrate to using a virtual control file by issuing this command:
   
   MIGRATE MASTER=sysid

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

You can migrate to the next preferred master system.

Follow this step:

Issue the following command:

MIGRATE MASTER
Chapter 4: Starting and Stopping CA MIM

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 that is 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.

This section contains the following topics:

- **Startup Activities** (see page 61)
- **How You Override Initialization Values at Startup Time** (see page 62)
- **Product Termination Considerations** (see page 64)

## Startup Activities

CA MIM is designed to run on two or more systems sharing a common control file, therefore, it does 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. On the FREE command supply the name of the second system for the SYSNAME parameter. If you use a REMOVE command instead of a FREE command, then the system does not reach synchronization, because it appears that the second system was never defined.
CA MIA Startup Activities

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

For CA MIA to start successfully, at least one device must be defined to the product. When 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 d 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.

How You Override Initialization Values at Startup Time

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

For example, to override the value for the COMMANDS parameter, issue the following command:

```
MI START COMMANDS=NEWCMNDS
```

The FORMAT Override Parameter

To format control files or checkpoint files, specify the FORMAT parameter on the MI START command. The default value for FORMAT 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 parameter data sets 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 that is selected and whether checkpoint files exist.

Format DASDONLY or CTCDASD

Format DASDONLY or CTCDASD when these statements are true:

- The GMR resides in the DASD control file.
- The checkpoint files are not required, when using the DASDONLY or CTCDASD communication method.

Follow these steps:

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:

   MI START,FORMAT=CF

3. Start CA MIM on all other systems.

Follow this procedure, when checkpoint files are required, and you are using the DASDONLY or CTCDASD communication method.

Follow these steps:

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:

   MI START,FORMAT=BOTH

3. Start CA MIM on all other systems normally.
Format CTCONLY

Format CTCONLY if:
- The GMR resides in the checkpoint files.
- The checkpoint files are required and no DASD control files exist.

Then, use the following procedure:

Follow these steps:
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:

   MI START,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:

   MI START,FORMAT=CHKPT

Product Termination Considerations

You can use the MIM SHUTDOWN command to stop CA MIM. You can stop it on all systems in the complex (global) or on a single system (local).

Global Shutdown Processing

On the z/OS systems in the MIMplex, 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.

We recognize that it may not be possible to quiesce all tape activity on a system. It is possible to stop CA MIM safely 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 when 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 66)

Global Shutdown Procedure

When 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. Issue the CA MIM SHUTDOWN GLOBAL command. 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.

On z/OS systems in the MIMplex, 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 the SHUTDOWN LOCAL, SHUTDOWN DUMP, or SHUTDOWN FREE command.

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 that are held by the system that you are stopping. The devices remain stopped until you restart that system or free resources and devices through the FREE command. Issuing the SHUTDOWN FREE command, can also be done to free devices automatically.

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

CA MIA Activities During Local Shutdown

**Important!** When GTA F 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 is not serialized with the rest of the systems in the complex where CA MIM is still running, creating data integrity exposures. An example of a new tape device activity is allocations, z/OS VARY commands, or device name replies to IEF238D WTORs.

Tape resources that are held by the local system at the time CA MIM are 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:

**Follow these steps:**

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 sysname

   **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 that is identified in Step 4:

   VARY NOTAVAILABLE GLOBAL

6. Wait for all VARY NOTAVAILABLE GLOBAL commands that are issued in Step 5 to complete.
   
The completion 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 that is 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.

   **Important!** When issuing 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 can 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 the ABEND codes 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.

How You Shut Down the System in a VCF Environment

When entering 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 or CTCDASD) also affects the response of CA MIM.

When a client system is shut down in a CTCONLY environment, then follow the procedures that are outlined in Local Shutdown Processing (see page 66). 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 that is 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.

**Important!** 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 that are 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) when 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 when 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 that are 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, any managed tape resources that are held by the system at the time CA MIM becomes inactive remain unavailable to all other systems until the FREE command is issued for the inactive systems.

Resources that are 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. When tape activity continues on the system where CA MIM is no longer active, then there is a risk of integrity exposures between the inactive system and the remaining CA MIM systems. For example, Integrity exposures are corrupted data sets and dual allocations of tape drives.

**Important!** Issue the FREE command only during error recovery, if you are sure that a system has 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 that are held by a system that has become inactive, or to inform CA MIM that a system is not 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 that are 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.
Chapter 5: Advanced Topics

This section contains the following topics:

- How You Activate Features of New Releases Dynamically (see page 72)
- How You Use the Dynamic Compatibility Feature with Checkpoint Files (see page 74)
- How You Activate New Features with ACTIVATE FEATURE (see page 75)
- How You Use ACTIVATE FEATURE with Checkpoint Files (see page 76)
- How to DEACTIVATE a FEATURE (see page 76)
- Performance Considerations (see page 77)
- CA MIM Driver (see page 77)
- Control File Internals (see page 78)
- Control File Externals (see page 82)
- Control File Medium and Performance (see page 83)
- How You Tune Control File Access Rates (see page 84)
How You Activate Features of New Releases Dynamically

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 COMPATLEVEL 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
MIM0621I 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 the Dynamic Compatibility Feature 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.
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 desired features using:

- The MIMINIT FEATURE statement, which you specify in the MIMINIT file.
- 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 that you wish to activate. This must be true on all systems.
2. Enter MI 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 MIMINIT file 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 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:

```
MI 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:

```
MI DEACTIVATE FEATURE=(featurename,FORCE)
```

Command DEACTIVATE

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
Performance Considerations

CA MIM provides resource serialization services among multiple systems running z/OS or z/VM. 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 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

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
Control File Internals

The two primary structures that are 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 second, 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, or CTC) 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.

```
Block Header

Global Management Record

System Table Entries  Space Mask  Transaction
Transaction  Transaction  Transaction
```

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 store CA MIM transactions temporarily. 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 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 used with both the *physical and virtual* control file architectures.

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

1. **RESERVE** the control file
2. Read and copy the first block (GMR) into local storage
3. Check the state of the MIMplex by interrogating the GMR
4. Read and direct incoming transactions to each component driver (MIA, MII, MIC)
5. Receive outbound transactions from each component driver
6. Update the GMR residing in private storage
7. Write new GMR and transactions out to the control file
8. **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 that is 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 that 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. Some CA MIM parameter definitions can cause unnecessary CA MIM transactions to be created, or can cause unnecessary CA MIM address space overhead, which elongates the control file cycle times.
You can expect the 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 allocation activity, or command and message activity, accesses the control file more frequently than CA MIM on a system that has little activity. If you have a MIMplex comprised of systems running a range of light through heavy workloads, then CA MIM address spaces on those systems access the control file at different frequencies. For example, 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 elongate individual control file cycle times and, in turn, degrade CA MIM transaction processing times.

### CA MIM Transactions

The other main structures that are 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 that are associated with each CA MIM component:

<table>
<thead>
<tr>
<th>Component</th>
<th>Transaction Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA MIM</td>
<td>GLOBALVALUE changes</td>
</tr>
<tr>
<td>CA MIA</td>
<td>Tape device status changes, preferencing</td>
</tr>
<tr>
<td>CA MIC</td>
<td>Commands and message text</td>
</tr>
</tbody>
</table>

The size of a particular CA MIM transaction is variable in length. For example, the size of a CA MIC cross-system message transaction depends on the length of the message text. The simplest form of an individual CA MIM transaction mapping follows:

<table>
<thead>
<tr>
<th>Transaction Map</th>
<th>Size</th>
</tr>
</thead>
</table>
The routing mask field is used to direct the transaction to specific systems. For example, if a user on SYSA routes a command to SYSB, CA MIC on SYSB must be informed of this fact. In this case, CA MIC on SYSA creates a transaction having a routing mask for SYSB, and writes the transaction to the CA MIM control file. Once CA MIM on SYSB has 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. For example, 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 generation allows the newly joining system or re-synchronizing systems (in the case of a migration), to understand immediately 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) initiates a *global-copy*. The volume of transactions that are generated during a given global-copy operation depends the amount of managed resource activity at the time the global-copy was initiated. Once all global-copy transactions are placed in the control file, the newly joining CA MIM address space reads, processes, and removes 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 that are 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, provide a control file large enough to handle global-copy operations should it be initiated during a period of peak activity. The maximum number of blocks that are used during a global-copy operation in your environment can be determined by examining the MAX USED field that is shown with the DISPLAY IO command. CA MIM also provides an early warning feature that provides an alert message when you exceed 50 percent of the total number of control file blocks.

Control File Externals

CA MIM uses two fundamental, high-level architectures of cross-system communication. The two 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 the same, differences exist as to how CA MIM Driver serializes access and performs I/O operation with each type. This section looks 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. With physical control files, serializing access is achieved using RESERVE/DEQ macros and I/O operations are performed using the STARTIO macro.

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 known 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 transfers the VCF from its data space to the CA MIM client address space using CA MIM_allocated CTC devices. The client CA MIM address space then reads and updates the VCF and then transfer the updated copy back to the master. The master then services the next VCF requestor, which can 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 the master system is running on z/OS as a member of a sysplex, client systems in same sysplex can communicate with the master system through either CTC devices or XCF signaling. Client systems running zVM always communicate through CTC devices. When CA MIM presents the VCF to XCF for cross-system transport, XCF can use XCF-allocated CTC devices, or XCF-allocated coupling facility list structures to perform the cross-system transfer on the behalf of CA MIM. The transport vehicle that is 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 Medium and Performance

The hardware medium that is used to store the CA MIM Physical Control File or transmit the CA MIM virtual control file 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 media and architectures.

<table>
<thead>
<tr>
<th>Average Cycle Time</th>
<th>Control File Type</th>
<th>Control File Medium</th>
<th>CA MIM address spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 ms</td>
<td>physical</td>
<td>ESCON DASD</td>
<td>all</td>
</tr>
<tr>
<td>15 ms</td>
<td>physical</td>
<td>FICON DASD</td>
<td>all</td>
</tr>
<tr>
<td>10 ms</td>
<td>virtual</td>
<td>CTC</td>
<td>client</td>
</tr>
<tr>
<td>1 ms</td>
<td>virtual</td>
<td>VCF</td>
<td>master</td>
</tr>
</tbody>
</table>

Due to numerous environmental factors, the Average Cycle Times in a particular enterprise may be higher or lower than those values shown here. You can see from this table that the type of medium that is 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 throughput.

**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 do 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 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 or any of the virtual control file methods. The control file tuning methods that are 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. We recommend 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 command 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 that are used during the implementation phase are SETOPTION MODE, INTERVAL, and CYCLE. The following sections on control file tuning explain the processes that are involved in tuning control file cycle rates.

**Control File Tuning: Analysis**

The objective of the analysis phase is to examine your control file I/O activity to determine access rates, transaction rates, service rates, and so on. The DISPLAY IO command provides the information that is used in the analysis phase.

**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 that is 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=57.638M9 WRITTEN=27.639M
XACT READ=88.885M PROCESSED=44.093M WRITTEN=26.650M
AVG: CYC=.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 that are used in the control file tuning analysis phase have been bolded in the preceding illustration. These key fields describe follow:

**AVG: CYC=.085302**

Represents the average control file cycle completion time in seconds. The completion time 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 that are 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 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

**XACT READ=2400.309**   **PROCESSED=120.025**

These fields represent the average number of transactions that are read, and the number of transactions that are processed by this system, during each control file cycle. In CA MIM, transactions should remain in the control file for a short time. Once a transaction is read and processed by all CA MIM address spaces, that transaction is removed from the control file.
How You Tune Control File Access Rates

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 that is 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 preferred 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 this 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 acceptable. Increasing or decreasing the number of control file cycles per second on certain systems corrects this problem.

Analysis Phase Summary

By examining the DISPLAY IO statistics for every system in the MIMplex, you can identify certain systems that are not performing control file cycles often enough to keep pace with your workload-intensive systems.

Disproportionate cycle rates (access frequencies), cause CA MIM on workload-intensive systems to constantly reread and rewrite transactions that are destined for systems that are not accessing the control file often enough. The process of continuously rereading and 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
- Less than optimum throughput
Based on the data in the DISPLAY IO illustrations that are used earlier in this section, the following analysis can be made: the MIMplex is communicating across systems using a DASD control file. The MIMplex 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 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 that are 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 80 ms (for 3390 DASD control file).
- The transactions read to transactions processed ratio is more than the acceptable 10:1.

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. To determine which system is not accessing the control file often enough examine the DISPLAY IO data from the external systems.

**Note:** For help on correctly tuning a control file, see Control File Tuning: Example (see page 93).

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 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

In practice, any of the communication media perform adequately for CA MIA and CA MIC, using default values for all of the tuning parameters. However, balancing control file cycles across the MIMplex:

- May improve transaction processing efficiency
- Reduce control file block count usage
- Reduce the average control file cycle time
- Improve overall MIMplex throughput

This section provides you with 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 alter the CA MIM internal control file cycle processes dynamically. The result of these commands is to either increase or decrease control file cycles on a given system. The following parameters are used to tune CA MIM control file cycle rates.

**SETOPTION MODE=DEMAND/GROUPS**

When running in DEMAND mode, mostly local CA MIA resource activity drives local control file cycles. When CA MIA activity is not present, CA MIM Driver performs a control file cycle at the expiration of a timer value. This value is calculated based on your SETOPTION CYCLE and INTERVAL parameters. DEMAND is the default processing mode as it allows CA MIA 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 for local CA MIA or CA MIC transactions to propagate, and then performs a control file cycle. In this case, local transactions are any that have been queued or grouped while waiting for the timer to expire. 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 preferred mode of operation.

DEMAND MODE can be used to increase control file cycles on systems that do not have enough CA 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.

<table>
<thead>
<tr>
<th>Interval Value</th>
<th>Number of Control File Cycles per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>1</td>
</tr>
<tr>
<td>0.50</td>
<td>2</td>
</tr>
<tr>
<td>0.20</td>
<td>5</td>
</tr>
<tr>
<td>0.15</td>
<td>7</td>
</tr>
<tr>
<td>0.10</td>
<td>10</td>
</tr>
<tr>
<td>0.08</td>
<td>13</td>
</tr>
<tr>
<td>0.06</td>
<td>17</td>
</tr>
<tr>
<td>0.05</td>
<td>20</td>
</tr>
<tr>
<td>0.04</td>
<td>25</td>
</tr>
<tr>
<td>0.03</td>
<td>33</td>
</tr>
</tbody>
</table>

GROUPS MODE is typically used to decrease the number of control file cycles. On rare occasions, some systems dominate the control file because they have so much managed activity that they perform control file cycles at an excessive rate. This domination causes other systems to be locked out of the control file resulting in intermittent MIM0100 or MIM0200 messages. In these cases, we recommend 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 occurred when the INTERVAL parameter is set to a particular value in GROUPS mode with 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.
How You Tune Control File Access Rates

Control File Tuning: Implementation

In this section, shows you 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 to CA MIM on every system. This re-initializes the performance statistics and allows 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. CA MIM can acquire statistics for all shifts, and ideally includes your heaviest workload period.

4. Issue the DISPLAY IO command on all systems. Print the statistics.
   The display data has two parts, one containing statistics since CA MIM startup, and the other containing statistics since last reset. Use the statistics that are accumulated since the last reset command was issued.

   Analyze the statistics. Determine which systems, if any, should perform 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 ten accesses per second and is a safe starting point when it is desirable to increase the number of cycles on an inactive system. Use the SETOPTION CYCLE=1 unless otherwise directed by CA MIM Technical Support.

   More information:
   Control File Tuning: Parameters

6. Issue the DISPLAY IO=RESET command to CA MIM on every system. This command 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 command 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 data contains acceptable values for every system.
10. Once you have achieved the desired results, make a permanent change to the MIMCMNDS member of the CA MIM parmlib. Using IFSYS/ENDIF statements that you can ensure the desired SETOPTION MODE, CYCLE, AND INTERVAL values are defined for each systems.

More Information:

Control File Tuning: Parameters (see page 90)

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 IO command are shown in the following illustration:

```
A
MIM:021 GDF SERVICE DISPLAY :
  REQUESTS  16,896,706 .075 SECS
MIM:0029 CONTROL FILE I/O DISPLAY
  AVG: .085 BLOCKS READ= 3.788 WRITTEN= 4.120
  RATE: CYC= 21.571
  XACT READ= 2400.309 PROCESSED= 120.025

B
MIM:021 GDF SERVICE DISPLAY :
  REQUESTS  76,159 .060 SECS
MIM:0029 CONTROL FILE I/O DISPLAY
  AVG: .090 BLOCKS READ= 5.701 WRITTEN= 1.001
  RATE: CYC= 1.086
  XACT READ= 2199.13 PROCESSED= 2198.89
```

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 little 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 CA MIA workload or a timer expiration. Because the number of requests on SYSA is high, it is a good assumption that CA MIM control file cycles are being driven by the workload. In addition, because the number of requests on SYSB is low, 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 that are 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 80 ms (for 3390 DASD control file).
- The transactions read to transactions processed ratio is more than the acceptable 10:1.

In short, the disproportionate control file cycle rates by the CA MIM address spaces are degrading the 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 ten times per second, instead of one time per second. Then issue DISPLAY IO=RESET on both SYSA and SYSB. After 24 hours, issue DISPLAY IO command again on both systems and analyze the data.

You can see:

- Average control file cycle times have gone down
- Blocks read and written have gone down
- Transaction read to transaction processed ratio has improved
- Time per request has 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 file so that CA MIM on SYSB has those values in effect at startup. You can add the following statements to the MIMCMNDS file:

```
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 file 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, update the CA MIM parameter files to have those values in effect at CA MIM start-up.
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 6: Troubleshooting

This section contains the following topics:

- How You Resolve Problems (see page 97)
- How You Respond to Control File Access Delays (see page 99)
- How You Activate Tracing (see page 110)
- How You Obtain Dumps (see page 110)
- Product Releases and Maintenance (see page 110)

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.

Software problem occurs.

Categorize problem and collect data. See the "Collecting Diagnostic Data" section.

Try to identify problem. See Interpreting Diagnostic Data.

See if fix exists. See Accessing the Online Client Support System.

Fix Found?

NO

Collect diagnostic data and contact technical support. See Calling CA Technical Support.

YES

Apply fix and verify that problem is solved.

PROBLEM SOLVED?

YES

Keep information for future reference.

NO

Work with CA Technical Support to solve problem.
How You Respond to Control File Access Delays

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 timestamp. 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 timestamp.

   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.
How You Respond to Control File Access Delays

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 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.
Chapter 7: User Exits

CA MIM provides a common exit interface that you can use to manage any CA MIM exit routine dynamically. 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.

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.

This section contains the following topics:
- Using Communication Words in Exit Routines (see page 111)
- Managing User Exits (see page 113)
- Common Exit Interface: MIMCMDXT (see page 114)
- MIMINIXT Exit Routine (see page 119)

Using 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 during an 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 one that is unique to a particular routine. The routine initializes this word during its initialization call.
  
  When a routine is reloaded, the prior value of its communication word is made available for the reinitialization call. Reloading enables the routine to either reuse the previous data area or release it from storage. The routine is responsible for maintaining any previously allocated storage.

- 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. For a description of the MIMINIXT exit routine, see MIMINIXT Exit Routine (see page 119).
Passing the Parameter List to All Routines

All routines for CA MIM are invoked through a standard parameter list, pointed to by register 1. The offset values are:

+0
Contains the address of the routine-specific parameter list.

+4
Contains the address of the routine-specific communication word. (The initial contents are 0.)

When CA MIM is stopped, VM automatically releases all storage within the CA MIM service machine. Therefore, if the routine-specific communication word points to storage within the CA MIM service machine, the address is invalid for the initialization call to the routine when you restart CA MIM.

+8
Contains the fullword of flags that are used to pass status information to the routine.

Whenever a routine is loaded, an initialization call is made to the module. For this call, +0 of the parameter list is set to 0 and the first byte of the status flag is set to X'80'.

+C
Contains the address of the global communication word, which is shared by all routines. The contents, which are set by the MIMINIXT exit routine, are passed unchanged for all subsequent calls to a routine.

On entry, +C into the common parameter list is either zero or the value from when CA MIM was last stopped. The value that is returned at +C into the common parameter list is stored and passed to all later invocations of any other routine.

Note: For information about a specific exit routine, see the appropriate CA MIM component Programming Guide.
Managing User Exits

CA MIM provides exit routines that you can use to customize the processing of CA MIM. The following list describes each routine and where you can find detailed coding instructions for the routine:

**GCMCMDXT**

This routine prevents certain local commands from being directed to external systems. This routine is available only when you run CA MIC. See the *CA MIC Message Sharing for z/VM Programming Guide*.

**GCMDELXT**

This routine changes the way CA MIC handles DOM (Delete Output Messages) orders. This routine is available only when you run CA MIC. See the *CA MIC Message Sharing for z/VM Programming Guide*.

**GCMRCVXT**

This routine prevents certain commands from executing on the local system due to a cross-system command issued from an external system. This routine is available only when you run CA MIC. See the *CA MIC Message Sharing for z/VM Programming Guide*.

**GCMSRCXT**

This routine prevents certain local messages from being directed to external systems, which are based on routing data that is associated with the messages. This routine is available only when you run CA MIC. See the *CA MIC Message Sharing for z/VM Programming Guide*.

**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.
**Common Exit Interface: MIMCMDXT**

**Controlling Exit Routines Through Commands**

You can control any CA MIM exit routine using the following commands:

- To display information about all exit routines, issue the following command:

  ```
  DISPLAY EXIT
  ```

  The information includes:
  - The logical and module name of a routine
  - The current status
  - The address at which the module is loaded
  - Its ESTAE options

- To display information about a specific routine, issue the following command:

  ```
  DISPLAY EXIT=exitname
  ```

  For example, `DISPLAY EXIT=MIMCMDXT` displays information for only the MIMCMDXT exit routine.

- To activate or inactivate a routine, to determine the status if an ABEND occurs, or to provide ABEND protection for it, issue the following command:

  ```
  SETOPTION EXIT
  ```

  You cannot use the EXIT parameter on the SETOPTION command to modify processing for the MIMINIXT exit routine.

You can specify these commands in the SYNCH MIM or CMNDS MIM files, or you can issue these commands from a MIMGR virtual service machine console while CA MIM is running.

The CA MIM Resource Sharing for z/VM Statement and Command Reference Guide provides detailed descriptions of both commands and their parameters.

**Common Exit Interface: MIMCMDXT**

You can use the optional MIMCMDXT exit routine for two purposes:

- To prevent certain CA MIM commands from being issued on the local system
- To modify a command, before it is issued by changing the command length and contents

This product invokes this routine whenever a CA MIM command is issued on the local system, whether the command was issued on the local system or issued as a cross-system command from an external system. The MIMCMDXT parameter list indicates when a command comes from an external system. You can change the length and contents of the command text with the MIMCMDXT routine.
Coding Rules

Follow these rules when coding the MIMCMDXT routine:

- Preserve all registers except registers 0, 1, and 15.
- Review all code that you define in this routine to prevent ABENDs from occurring. This product may try to recover from ABENDs but can terminate when it experiences repeated ABENDs.
- Do not use these macros
  - GCS OPEN
  - CLOSE
  - LOAD
  - LINK
  - BLDL

Or any other GCS macro that implicitly or explicitly calls the WAIT supervisor.

- Do not use the STIMER or TTIMER macros.
- Ensure that you interpret the command name properly when intercepting a command with this routine. Command names are converted to the new name (and parameters). Therefore, base your comparisons on the true command name rather than any alias.
- If you want to alter the command text, 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. For the mandatory code for this subroutine, see the example in Issuing Messages from MIMCMDXT (see page 118) 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, 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.
- To assemble an exit, use High-Level Assembler.

Assemble and link edit the completed routine as an authorized program into a load library and include the load library on the GLOBAL LOADLIB command in the PROFILE GCS for the MIMGR ID. Use the module name that you provided during the assemble and link edit to activate the exit through the SETOPTION EXIT command.
The sample file named MIMCMDXT ASSEMBLE on your distribution tape contains a sample exit that you can modify. A LINKEXIT EXEC is provided on the tape to link the sample routine into a load library called MIMEXITS LOADLIB.

For more information on the SETOPTION command, see the CA MIM Statement and Command Reference Guide.

Entry Environment:

Supervisor state, problem program key, addressing mode (AMODE) 31. Preserve this entry environment upon entry and restore it upon return.

Register Usage at Entry:

Common Parameter List

R1
This register contains the address of the parameter list, in the following format:

+0
Contains the address of the MIMCMDXT exit routine parameter list. This list is not used for initialization calls.

+4
Contains the MIMCMDXT routine communication word. The content of this word is set by the routine during its initialization call and is passed unchanged for all subsequent calls.

+8
Contains the fullword of flags that are used to pass status information to the routine. The first byte x'80' bit is set on for the initialization call.

+C
Contains the global communication word.

R10
Reserved. Do not alter it if you are using a message subroutine to issue messages.

R11
Reserved. Do not alter it if you are using a message subroutine to issue messages.

R13
Contains the address of the standard save area.

R14
Contains the return address for CA MIM.
R15

Contains the entry point address for the MIMCMDXT routine.

All other registers are undefined.

MIMCMDXT Parameter List
+0

Contains the address of a two-word vector. The first word of the vector contains the address of a command name. The second word of the vector contains the true length of the command name. Do not alter this information.

+4

Contains the address of the command text buffer. You can change the command text at this offset.

+8

Contains the true length of the command text buffer.

+A

Contains the origin of the command. If the high-order bit is on, the command was issued by a CMS user on the local system or by a user on an external system. If it is off, the command was issued at the MIMGR console on the local system.

* 0000 If command is from logged on console
* 0001 If command sent by SMSG from VM ID
* 8001 If command sent by IUCV (MI MODULE) from VM ID

+C

Contains the address of the message subroutine that you can use to send a message to the console or CMS user who issued this command (as described here).

+10

Contains the address of console authority flags.

CMDXAINF EQU B'00000000' ..INFO authority - all off
CMDXASYS EQU B'10000000' ..SYSTEM authority
CMDXAIO EQU B'01000000' ..IO authority
CMDXACON EQU B'00100000' ..CONSOLE authority
CMDXAALL EQU B'11100000' ..ALL (SYS, IO, CONS), but not MASTER

+14

Contains the pointer to an 8-byte CMS user ID.

+18

Reserved

+4F

Reserved
Return Code, Register 15:

0

Processes this command normally.

4

Rejects this command and issues message MIM0140 which notifies the issuing console or CMS user.

8

Rejects this command but does not issue a message to notify the issuing console or CMS user.

When you specify a value other than 0, 4, or 8, CA MIM rejects this command and issues MIM0141. This message notifies the issuing console or CMS user.

Note: All registers (except 0, 1, and 15) upon return must be restored.

Issuing Messages from MIMCMDXT

When sending a message from the MIMCMDXT routine to the console or CMS user who issued a command, use the message subroutine that is provided at offset +C of the exit-specific parameter list. Here is the 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
```

This message is recorded in the TRACE MIM file if the TRACE feature is active.
MIMINIXT Exit Routine

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 shutdown CA MIM. 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.

Coding Rules for the MIMINIXT Exit Routine

Follow these rules when coding the MIMINIXT routine:

■ Preserve all registers except registers 0, 1, and 15.
■ All CA MIM exit routines execute under GCS. Therefore, use GCS macros when coding the routines. Also, assemble the routines using the GCS macro libraries.
■ Acquire storage in a key that is usable to the exit routines you plan to invoke later.

Note: 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, CA MIM tries to load and activate module MIMINIXT.

Note: Specifying MIMINIT INITEXIT=NONE causes CA MIM to bypass all MIMINIXT processing.
■ To assemble an exit, use High-Level Assembler.

If you write your own MIMINIXT exit routine, assemble and link edit the completed routine with the module name you specify on the INITEXIT parameter of the MIMINIT statement. Link edit the module into a load library and add the load library to the GLOBAL LOADLIB command in the PROFILE GCS file for the MIMGR user ID.

The sample file MIMINIXT ASSEMBLE on your distribution tape contains a sample MIMINIXT exit routine. A LINKEXIT EXEC is provided on the tape to link the sample MIMINIXT routine into a load library named MIMEXITS LOADLIB.

Entry Environment:

Supervisor state, problem program key, addressing mode (AMODE) 31. Preserve this entry environment and restore it upon return.
Register Usage at Entry:

Common Parameter List

The common parameter list is mapped by the UXPARM. DSECT found in the UXPARM COPY file.

R1

Contains the address of the parameter list, in the following format:

+0

Not used; therefore, the value is unpredictable.

+4

Contains the communication word of the MIMINIXT routine. The content of this word is set by the routine during its initialization call and is passed, unchanged, during the next restart whenever possible.

+8

Contains the fullword of flags that are used to pass status information to the routine. The first byte should always be set to X'80'.

+C

Contains zeros or the contents of the global communication word from a previous execution of CA MIM.

R13

Contains the address of the standard save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the MIMINIXT routine.

All other registers are undefined.

Note: No return codes are supported.
Chapter 8: Maintaining LMP Keys

Each CA MIM component (CA MIA or CA MIC) is identified by an assigned two-character LMP key. You are issued an LMP Product Key Certificate for each LMP-enabled component for which you are licensed. This Key Certificate contains an execution key for the component that consists of the LMP key code for the component, an expiration date, and the CPU model and serial number on which the component is being licensed to run. Enter this key as part of the component installation process.

This section contains the following topics:

CALMP KEYS File (see page 121)
Validity of LMP Keys (see page 121)
Changing CPUs (see page 122)
Adding New LMP Keys (see page 122)
About the CALMP KEYS File (see page 123)

CALMP KEYS File

LMP keys are maintained in the CALMP KEYS file. The file is fixed format with a record length of 128 and can reside on either the 192 disk of the VMRMAINT ID (CA VM:Manager Management Suite ID) or a local disk on the MIMGR ID. The disk containing the CALMP KEYS file must have virtual address 1FF on the MIMGR ID. For more information regarding the placement of the CALMP KEYS file, see the CA MIM Installation Guide.

Validity of LMP Keys

The existence and validity of the LMP key is checked at these times:

- At CA MIM initialization
- Whenever a CA MIM AUTHCHK command is entered
- Whenever the interval specified on the CA MIM SETOPTION MIM AUTHCHECK parameter expires.

LMP verification determines whether a component is authorized to execute at your site on that specific CPU. If CA LMP cannot find a valid LMP key for the model and serial on which the component is running, warning messages are issued. These messages do not affect CA MIM performance, and the product continues to run.
Each LMP key has an expiration date. However, CA MIM does not terminate nor is its operation affected in any way for not having a valid LMP key. Warning messages are issued for each component whose LMP key is about to expire. When you start getting messages that your LMP key is due to expire, contact CA-TLC: Total License Care support as soon as possible to avoid potential license expiration. Place any new LMP code in the CALMP KEYS file, and delete any expired codes.

Changing CPUs

If you plan to change CPUs in the future, contact CA Technical Support. Be prepared to provide the following information:

- Current CPU make, model, and serial number
- New CPU make, model, and serial number.

Adding New LMP Keys

When you install a new component or change CPUs, add an LMP key for each installed component in the CALMP KEYS file. The LMP key for the component is valid for all subsequent releases of that component. When upgrading to a new release of CA MIM, the same LMP keys are used; you do not need new LMP keys. An LMP key for a component becomes invalid after the expiration date is reached, or the CPU model and serial number changes.

Add a new component LMP key to the CALMP KEYS file and delete the expiring key. The new LMP key is effective immediately. You can add a new LMP key to this file in one of these ways:

- Using an option from the VMIMAINT Main Menu (if you have a common CALMP KEYS file on the CA VM:Manager VMRMAINT ID)
- Manually editing the CALMP KEYS file

For information on using the VMIMAINT Main Menu to enter LMP keys, see the CA VM:Manager Management Suite Installation Guide. For information on manually entering LMP keys in the CALMP KEYS file, see Editing the CALMP Keys File (see page 126).
About the CALMP KEYS File

The CALMP KEYS file can reside on the VMRMAINT 192 minidisk (CA VM:Manager ID) or on the local MIMGR ID. This file is fixed record format and is 128 bytes in length. The disk containing the CALMP KEYS file must have virtual address 1FF on the MIMGR ID. For more information regarding the placement of the CALMP KEYS file, see the CA MIM Installation Guide.

Types of Keys

The CALMP KEYS file can contain three types of keys:

- Product
- EKG
- Site ID

Product Keys

A product key is valid for a specific component for a specific CPU (or a specific site) until the expiration date. Each product key begins with a PROD statement and specifies the product, expiration date, and associated LMP key.

Product keys have the following format:

PROD(pp) DATE(ddmmmyy) CPU(tttt-mmmm/sssss) LMPCODE(kkkkkkkkkkkkkkk)

Or

PROD(pp) DATE(ddmmmyy) CPU(SITE—****/sssss) LMPCODE(kkkkkkkkkkkkkkk)

pp

Is the two-character component code.

Values for pp are:

- CS – CA MIA (GTAF and TPCF)
- CT – CA MIC (GCMF and ICMF)

ddmmmyy

Is the CA LMP licensing agreement expiration date. For example, 15JUL05 specifies an expiration date of July 15, 2005.

Note: CA LMP licensing supports the twenty-first century date format. However, only the last two digits of the years in the new century are displayed.
tttt-mmmm/ssssss

or

SITE--****/ssssss

Specifies the CPU type, model, and serial number on which the component will run. If the CPU type and model require fewer than four characters, insert blank spaces for the unused characters. If your site has been issued a site license for a component, this specifies the site ID that must appear on the SITEID record for this key to be valid, and the CPU serial number.

kkkkkkkkkkkkkkkk

Specifies the LMP key for the component.

EKG Keys

In emergency situations such as disaster recovery, an Emergency Key Generator (EKG) can issue a temporary key for the components. An EKG key is valid for ten days from the date of activation, and it is valid for all components and on all CPUs.

Each EKG key begins with the EKG statement, and is in the following format:

EKG(nnnnnnnn)

nnnnnnnn

Specifies the emergency key.

It is not necessary to implement EKG keys to activate components in a disaster recovery situation, because CA LMP allows the components to run uninterrupted regardless of the CPU on which they are running.

Note: When bypassing EKG, warning messages are issued at any time CA MIM performs authorization checking. These messages do not impact component operation, but they are visible on the console and to operators.

To suppress these messages, contact CA-TLC support and, in an emergency, request an EKG key. Once you have obtained an EKG key, edit the CALMP KEYS file and add the key information as shown in the example.
Site ID Keys

A site ID key is used in combination with an LMP key to provide non-CPU specific key support. Site ID keys are typically generated by CA-TLC support for large sites running many components.

If a site ID exists, it must be the first statement in the CALMP KEYS file.

Each site ID key begins with the SITEID statement and is in the following format:

```
SITEID(xxxxxxx) SITECODE(xxxxxxxxxxxxxx) NAME(name)
```

- `nnnnnnn` Specifies the site ID.
- `xxxxxxxxxxxxxxxxxxxx` Specifies the site ID key.
- `name` Specifies the company name.

Sample CALMP KEYS File

A sample CALMP KEYS file follows:

```
PROD(AA) DATE(15MAY03) CPU(1234-000 /123ZZ2) LMPCODE(ABCD1234H678J098)
PROD(BB) DATE(01FEB03) CPU(1234-000 /123ZZ2) LMPCODE(1234GH586K79H86L)
PROD(1C) DATE(01FEB03) CPU(1234-000 /123ZZ2) LMPCODE(8760PO19GF6D65S44)
PROD(DA) DATE(01FEB03) CPU(1234-000 /123ZZ2) LMPCODE(854QWE567YU8900)
PROD(DD) DATE(01FEB03) CPU(1234-000 /123ZZ2) LMPCODE(2298GJK80LPP4555)
PROD(3L) DATE(01FEB03) CPU(1234-000 /123ZZ2) LMPCODE(EGG6789IOP99999)
PROD(5F) DATE(01FEB03) CPU(1234-000 /123ZZ2) LMPCODE(NMV89IO005HHT44)
PROD(HH) DATE(01FEB03) CPU(1234-000 /123ZZ2) LMPCODE(DFTY67890HJK9999)
PROD(K9) DATE(01FEB03) CPU(1234-000 /123ZZ2) LMPCODE(FB89COR92VET97SS)
```
About the CALMP KEYS File

Editing the CALMP KEYS File

Typically you edit the CALMP KEYS file only when you are adding a new LMP key, or if you make a mistake entering the LMP key. Routine maintenance is not required. It is not necessary to bring CA MIM down when editing the CALMP KEYS file. CA MIM detects the updates to the file the next time it performs authorization checking.

To edit the CALMP KEYS file, either use the CA:VM Manager menu interface or use XEDIT to edit the file directly. You can use the VMIMAINT Main Menu if you have a common CALMP KEYS file on the CA VM:Manager VMRMAINT ID. For information on using the VMIMAINT Main Menu to enter LMP keys, see the CA VM:Manager Management Suite Installation Guide. To edit the file directly using XEDIT:

Follow these steps:

1. Log on to the ID that owns the CALMP KEYS file. This is either the VMRMAINT ID or the MIMGR ID.

2. Enter the following command where m is the file mode of the disk containing the CALMP KEYS file:
   
   XEDIT CALMP KEYS

3. Enter the LMP key for each component exactly as it appears in the Product Key Certificate you received. If you are entering a new LMP key to replace an expiring key, be sure to delete the expiring key. If you do not delete the expiring key, warning messages are issued until the key expires.

4. When you are finished, save and exit the file. The new LMP key and any other changes are effective immediately and are used the next time CA MIM performs authorization checking. You can issue the CA MIM AUTHCHK command to invoke immediate authorization checking to verify the updated contents of the CALMP KEYS file.

CA LMP Error Handling

You may encounter error messages whenever CA MIM attempts authorization. These attempts can occur at startup, when the AUTHCHK command is issued, or when the SETOPTION MIM AUTHCHECK interval expires. The CA LMP error messages are documented in the CA MIM Message Reference Guide.

For assistance with LMP key issues, contact CA Technical Support (see page 3).
Appendix A: Ensuring Integrity When Sharing DASD Under z/VM

When sharing DASD under z/VM, you should:

- Verify 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.

This section contains the following topics:

- Serializing Access in z/VM Environments (see page 127)
- Serializing Access Through Real Reserve/Release Processing (see page 128)
- Serializing Access Through Virtual Reserve/Release Processing (see page 129)

Serializing Access in z/VM Environments

The type of reserve/release processing that you need depends on whether you have several real processors, several guest systems, or a combination of both.

- 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.

Use both real and virtual reserve/release processing if you have several guests running under one z/VM operating system and you have at least two real processors in your complex.

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.
Serializing 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 group to access a device, real reserve/release can prevent integrity exposures among real processors.

When a guest operating system running under z/VM performs I/O to DASD, the CP component of z/VM takes responsibility for issuing a channel program to the real hardware on behalf of the guest. During this process, some portions of the channel program may be modified by z/VM.

If the device is not dedicated to a processor already, then the DASD hardware executes the reserve channel command word (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, an integrity exposure can occur when several real processors are sharing a device.

Determining Whether z/VM is Sending Reserve CCWs to Devices

To see if z/VM is sending reserve CCWs to a certain device, specify that device address as a parameter on the QUERY command. For example, to see if z/VM is sending reserve CCWs to device 123, issue this 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, 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.
Serializing Access Through Virtual Reserve/Release Processing

Telling z/VM to Send Reserve CCWs to Devices

If z/VM is not sending reserve and release CCWs to your devices, take extra 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 device address 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 SHARE 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.

Be certain your control file is defined as a full pack minidisk if you want to send real reserve/release orders to the hardware. z/VM suppresses reserve release orders, even if you set SHARED ON, if the minidisk is not a full pack.

Serializing 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.

Invoke virtual reserve/release in this situation because guests under the same z/VM operating system share the same path group to a device. Real reserve/release cannot ensure integrity among these guests because it has no effect on guests that share a path group 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)
- An entire device to a guest (if you defined that device as a full-pack minidisk).
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. 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.

**Note:** You may need to tell z/VM systems to send reserve CCWs to the hardware. For information, see *Telling z/MV to Send Reserve CCWs to Devices* (see page 129).

### Invoking 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 END 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 END), 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
```
Appendix B: Using the VMPROOF Feature

The VMPROOF feature performs two important functions for verifying the proper DASD control file operation:

1. Identify configuration errors that might cause failure of reserve/release processing.
2. Simulate control file I/O to test reserve/release processing between two or more systems.

VMPROOF can run in any IBM supported release of z/VM. VMPROOF can be used on a z/VM system, along with MIMPROOF on a z/OS system, to verify that a control file shared between VM and z/OS is being properly serialized. For more information about MIMPROOF, see the CA MIM Resource Sharing for z/OS Systems Programmer Guide.

This section contains the following topics:

- Using VMPROOF for Configuration Checking (see page 131)
- Using VMPROOF to Simulate Control File I/O (see page 133)
- Verifying Serialization for a z/OS Control File (see page 134)
- Verifying Serialization for a CMS Control File (see page 135)
- Verifying Serialization for a CMSFP Control File (see page 136)
- VMPROOF Notes (see page 138)

Using VMPROOF for Configuration Checking

It is not necessary to allocate a control file for this function. Under CMS, on the ID where you plan to run CA MIM, enter the following command:

VMPROOF vdev

vdev

Specifies the virtual address of the DASD where the control file resides, or will reside.
VMPROOF displays several lines of information about the DASD similar to the following example. The information varies according to the kind of system.

```
vmproof 945
VERSION 2.0
Controller type 3880, Model 05
DASD type 3380, Model 1E
00885 cyls (0375 hex)
   Device characteristics:
0000 38800533 B01E8100 0000200E 0375000F
0010 DE00BB60 04400120 01EC00EC 00000000
0020 00000000 00000000 23230900 BB740000
0030 00000000 00000000 00000000 00000000
Real device address: 0227, Real VOLSER: "MVS340"
Starting real cylinder: 06EA (hex) Extent: 0375 (hex)
Size of real pack: 02655 (decimal) 0A5F (hex)
WARNING: Not a full pack minidisk. Does not start on real cylinder 0.
WARNING: Not a full pack minidisk. Extent is less than full pack. Ready;
```

The preceding example shows the result you might see on a z/VM system. The results show that the DASD at 945 is a minidisk on the MVS340 pack at real address 227. VMPROOF warns that the DASD is not a full pack minidisk. This is not necessarily an error, but does mean that CP does not transmit reserve and release orders to the real device.

If the control file shares multiple processors, serialization does not work properly. If the control file is only shared among guests on the same z/VM system, however, virtual reserve/release processing enables the control file to work correctly, even in the presence of this warning.

In this example, 945 is not a full pack because it fails both tests that CP uses to define a full pack; to be qualified by CP as a full pack minidisk, cylinder 0 of the minidisk must correspond with cylinder 0 of the real disk, and the size of the minidisk must be equal to or larger than the size of the real pack. VMPROOF also reports the size of the real pack, so you can update your MDISK statement if necessary.

VMPROOF also issues a warning if the virtual DASD does not support reserve/release. This warning is usually an indication that 'V' needs to be added to the MDISK mode to invoke virtual reserve/release processing.

When a DASD device is attached to the user instead of being defined as a minidisk, VMPROOF reports that the device is DEDICATED. Dedicated DASD handles RESERVE/RELEASE correctly if the DASD controller itself handles the orders.

The display also includes device characteristics in four lines of hexadecimal data. This is the 64-byte response to a "read device characteristics" order sent to the DASD. This information may be useful in debugging difficult configuration problems.
Using VMPROOF to Simulate Control File I/O

For this test, a control file must be allocated on the DASD before you begin. CA MIM can use three supported kinds of control files, and VMPROOF can verify them all. Valid types are MVS, CMS, and CMSFP. VMPROOF needs slightly different information for each type of file, and an example is given for each.

This test functions by writing data to disk, then reading the same data back later, and testing for corruption. A single disk record is used by all systems participating in the test. Each system modifies a different eight-byte field within the record. If serialization works correctly, each system consistently retrieves the same data from the record that it writes, but if serialization fails, one or more systems detects an overlay of its eight-byte area.

This test is most valuable when it is run on multiple systems at the same time, but even running on a single system the test verifies that DASD I/O is working properly, and that RESERVE orders do not cause a unit check.

Once the test begins operation, it continues until an error occurs or until it is manually stopped. The VMPROOF program recognizes two console commands, STATUS and STOP.

- The STATUS command can be issued at any time, and produces a display of the test data. By using this command, you can observe how test data changes while the test is running. Data from all systems is visible, and an asterisk is placed beside the data from the system on which you issue the command. Two full words of data, or eight bytes, are displayed for each system. The first word is a decimal counter, and shows how many times the control file has been written by the corresponding system. The second word is a time stamp.

- The STOP command shows the same information as STATUS, and then terminates the test on the local system. Do not use HX to stop VMPROOF because this can result in leaving a RESERVE in effect on the control file.

You may observe a slight delay between the time you enter a STATUS or STOP command and when it is processed. This is because these commands are processed only when no reserve is held on the control file, and a full write/read cycle has been completed.

The general form of the VMPROOF command is:

```plaintext
VMPROOF vdev
VMPROOF vdev[MVS parm dsn]
VMPROOF vdev[CMS parm volser]
VMPROOF vdev[CMSFP parm volser cyl]
```

MVS, CMS, or CMSFP specifies which type of control file you want to test. Specify other parameters for each type of control file.
Specify values for these parameters:

**vdev**
Indicates the virtual address of the DASD on which the control file resides.

**parm**
Specifies a four-character word that sets parameters for the VMPROOF program. The first character is the letter F or N, meaning respectively format or do not format the control file. In this context, *format* refers to an initialization procedure that VMPROOF performs. It is not the same as the CMS FORMAT command, which must be used to allocate a CMS or CMSFP control file.

The first system to begin VMPROOF or MIMPROOF operations should use F and all others should use N. The next character is a number, 0 through 7. Choose a different number for each system running VMPROOF or MIMPROOF. If you use the same number on two systems, VMPROOF incorrectly reports a RESERVE failure. The last two characters of *parm* are the time period, in 1/10 second intervals, that VMPROOF waits between alternate reads and writes.

**dsn**
Specifies the z/OS data set name of the control file.

**volser**
Specifies the minidisk label of the CMS minidisk that is used for a CMS or CMSFP format control file.

**cyl**
Specifies the absolute cylinder address, in hex, of the CMS minidisk being used for the CMSFP control file.

---

**Verifying Serialization for a z/OS Control File**

First, allocate the control file on the pack you will share between z/VM and z/OS. Allocation of a z/OS format file can only be done from z/OS, either by using TSO or by submitting a job. A z/OS format control file should be allocated as a sequential file, and usually should be ten (10) cylinders long. Allocate only a single extent.

Log on to MIMGR and IPL CMS. Link in write mode to the DASD containing the file you allocated on z/OS. The link could be established by putting an MDISK or LINK statement in the directory entry for MIMGR or it could be done by using the CP link command. Alternatively, you could attach the z/OS pack to the MIMGR user ID or could use an equivalent DEDICATE statement in the directory.
Verifying Serialization for a CMS Control File

CMS control files are used when no z/OS system is present. They are not addressable from a z/OS system. In z/VM systems, you may need to use a CMSFP format file, which is described in more detail in the next section.

If the CMS format control file is to be shared between a host z/VM system and its z/VM guest systems, you should use a single MDISK statement at the first level, and as many LINK statements as you need. Mode MWV should be specified on the MDISK statement, and MW on the LINK statements. Always use MWV mode when two or more guests on the same z/VM system will share the same control file, even if the control file will also be shared with a z/VM system running on another processor.

A CMS control file is placed at the top end of a CMS minidisk, in an area not used by the CMS file system. The MIMGR 191 minidisk should not be used for a CMS control file. Instead, define a minidisk that you will use only for control file purposes if you choose to use a CMS format control file.

Suppose you define a MIMGR 291 minidisk for a control file. The minidisk should be defined one cylinder larger than the size you will need for the control file. For this example, we will define a ten-cylinder control file, so the minidisk should be 11 cylinders long. To make the disk usable for control file use, perform the formatting steps shown here:

```plaintext
FORMAT 291 B#1#CTFL1
FORMAT 291 B 1 (RECOMP)
```

This results in a CMS minidisk that is formatted for CMS files in the first cylinder only. The remaining ten cylinders are not available for CMS files on the minidisk. (The control file is not stored in CMS file format.) "CTFL1" is the minidisk label you are assigning to MIMGR 291. You may choose any label of six characters or less. The label you specify here will have to be matched on VMPROOF.

Start the VMPROOF program. In the example, we assume that the z/OS pack is available to MIMGR at virtual address 945, and that the control file data set is named 'MIM.CF.01'.

```
VMPROOF 945 MVS F007 MIM.CF.01
```

VMPROOF verifies the configuration of 945, formats the area beginning at cylinder 1, and then begins writing and reading simulated control file data.

You can issue a similar command on other z/VM systems or on other guests that share the control file to verify that serialization is working as you expect. As described, the parameter F007 would be changed for each additional system you add to the test. Only the first system uses the letter "F" (others all use "N"), and each system uses a different second digit. Also, each system uses a different delay time.
Note: The operation performed by the CMS FORMAT command only formats space for CMS file operations and reserves space for the control file. After this is done, it is still necessary for the first VMPROOF or MIMPROOF that starts operations to request a format in its parm field.

It is not necessary to access the minidisk before running VMPROOF because VMPROOF sends its own I/O to the minidisk without using the CMS file system. The same is also true for the CA MIM production program. Therefore, there is no need to have an ACCESS statement for control file disks in PROFILE GCS or PROFILE EXEC.

You can store CMS files in the CMS portion of the minidisk if you want to, and you can access this disk to use the files for CMS or GCS purposes, but remember that the disk will be unavailable at times when CA MIM is running, due to RESERVE contention. Files that CA MIM accesses, such as AUTHUSER MIM and UNITS MIM, should not be stored here.

To run VMPROOF tests of this control file, enter the following command:

VMPROOF 291 CMS F007 CTFL1

VMPROOF verifies the configuration of 291, formats the area beginning at cylinder 1, and then begins writing and reading simulated control file data.

You can issue a similar command on other z/VM systems or on other guests that share the control file to verify that serialization is working as you expect. As described above, the parameter F007 would have to be changed for each additional system you add to the test. Only the first system will use the letter "F" (others all use "N"), and each system uses a different second digit. Also, each system should use a different delay time.

Verifying Serialization for a CMSFP Control File

The CMSFP control file differs from the CMS control file only in the way VMPROOF and MIMGR address it. For a CMSFP control file, VMPROOF performs I/O on the full pack on which the minidisk resides. It is not even necessary for MIMGR to have a link or an MDISK statement to the minidisk.

The following sample lines from a directory entry illustrate the way to define a CMSFP control file:

USER MIMGR

... 

MDISK 291 3390 447 011 VMVOL1 MW
MDISK 555 3390 000 885 VMVOL1 MWV
In this example, the first MDISK statement defines a minidisk for MIMGR at virtual address 291. The minidisk is 11 cylinders in length, and the first cylinder is at decimal real cylinder 447 on the real pack labeled VMVOL1. The second line defines a full pack disk on the same real pack, at virtual address 555. It is not necessary to know the real address of pack VMVOL1.

VMPROOF will need three pieces of information about the minidisk to use this file in CMSFP mode. First, it needs the vdev address of the full pack minidisk. Second, it needs the VOLSER of the minidisk (not the VOLSER of the full pack), and finally, the absolute cylinder, in hex, where the 291 minidisk is allocated on the full pack.

The following commands would be used to prepare the minidisk for control file testing, and then to start the test:

```
FORMAT 291 B#1#CTFL1
FORMAT 291 B 1 (RECOMP
VMPROOF 555 CMSFP F007 CTFL1 1BF
```

**Note:** The value "1BF" is derived by converting the origin of the 291 disk, as stated on the MDISK statement, to hex.

When you are running VMPROOF on two or more z/VM systems and each system addresses the same CMSFP format control file, it is only necessary to define the minidisk that corresponds to the control file on one of the systems. The minidisk is only used to perform the CMS FORMAT and CMS FORMAT RECOMP functions, and this is only needed one time. All systems that will run VMPROOF must define or link the full pack minidisk to the user who will run VMPROOF.

CMSFP and CMS control files have exactly the same data structure. Therefore, some systems can refer to a control file as a CMS control file and others can refer to the same file as a CMSFP control file. Those referring to the CMS control file will perform I/O to the minidisk and those referring to the control file as CMSFP will perform all I/O to the full pack.

In z/VM, reserves are sent to the real DASD only if the RESERVE order was originally sent to a full pack minidisk, and only if the SHARED attribute is on for the real pack. This makes it necessary to address DASD as a full pack in z/VM to assert RESERVE on a real pack.
The information contained in this section provides some additional considerations when using VMPROOF. Please note the following:

- VMPROOF cannot find some configuration problems. A common one is specifying multiple MDISK statements for the same full pack minidisk. This subverts correct operation of virtual reserve/release (VRR). For VRR to work, there must be a single MDISK specification using mode MWV and all other references to the same full pack must link to the MDISK using mode MW.

- When z/OS or z/VM runs under z/VM, the guest definition at the first level can cause problems that are hard to find at the second level. For example, if a full pack minidisk is defined as read/only (R/O) at the first level, but read/write (R/W) at the second level, second level applications will see the disk as R/W. In this case, MIMPREG or VMPROOF running at the second level will fail when it attempts to write.

- VMPROOF does not check for the SHARED setting in z/VM systems. Even for a full pack minidisk, z/VM will not forward RESERVE and RELEASE orders to the hardware unless SHARED is set on. You can check this setting yourself by using the QUERY rdev command from any class B user.

- Never run two or more applications that might issue RESERVE to different minidisks on the same real pack. If you allow this to occur, the status of RESERVE on the real pack becomes unpredictable. Fortunately, not many applications issue reserves to a CMS minidisk. It never happens in CMS, for example.

- VMPROOF is compatible with MIMPREG. You can run VMPROOF on z/VM and MIMPREG on z/OS at the same time and sharing the same control file to test for correct operation. MIMPREG cannot access CMS or CMSFP control files, however. MIMPREG is distributed with CA MIM for z/OS and documented with that product.

- If VMPROOF detects corrupted data, it issues a "RESERVE ERROR DETECTED" message, dumps memory, and stops running with return code 24. The dump will be sent to the reader of the user running VMPROOF.

- VMPROOF stops running when you issue a STOP command or when it detects corrupted data. If VMPROOF is running on two or more user IDs, either on the same system or on different systems, stopping one copy of VMPROOF does not affect the status of the others. Each must be stopped independently of the other.
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.

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.
assignable tape devices

Assignable tape devices refers to the following types of devices: 3480, 3490, 3590, and any other type of compatible tape device.

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.

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.

checkpoint files

Checkpoint files are non-shared DASD files used to track system status information when using CTONLY communication method. 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, CTONLY, CTCDASD, and NONE.

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 CTONLY 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.

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.

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 MIMUNIT 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 MIMUNIT.

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 preferred the device.

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.

dual allocation

A dual allocation is a situation in which two or more jobs concurrently allocate the same device.

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.

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.

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**.

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 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.

**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.
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

**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 CA1.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 CA1.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 CA1.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 non-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 non-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 non-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**

Overgenned 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 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.

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.

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.

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, GTA F, 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.

**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.

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.
**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.

**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 RESERVE facility**

See RESERVE facility.

**z/VM**

*z/VM* is a generic term for the *z/VM* and VM/ESA operating systems.
Index

C
CA L-Serv • 10, 41
CA MIA
   CA MIA, starting • 62
description • 9
CA MIC
description • 10
CA MIM
   starting • 61
   stopping • 64
CA MIM Driver • 10, 77
checkpoint files
   creating • 53
   formatting • 56
communication methods
   CTCONLY • 40
COMPATLEVEL • 72
control file blocks, and global copy process • 81
control files • 78
   contrasting with checkpoint files • 53
   formatting • 56
   migrating • 57
   physical • 82
   placement • 54
creating checkpoint files • 53
CTC devices
   configuring for VCF • 45
CTCDASD communication method • 57
CTCONLY communication method • 40
CTCPATH statement
   CTCPATH statement, using to configure CTC
devices • 45

D
DASD control files
   migrating • 58
   devices, freeing • 69
diagnostic procedures • 98
dumps • 110

E
external system, problems with inactive • 100

F
features, activating dynamically • 72
formatting checkpoint files • 56
formatting control files • 56

G
GCMF, description • 10
Global Management Record (GMR) • 56
global shutdown • 65
GLOBALVALUE command, using to select master
   systems • 43
GTAF, description • 9

I
ICMF, description • 10
inactive system, troubleshooting • 103
initialization values, overriding at start-up • 62

L
local shutdown • 66

M
MIMplex
   removing a system from • 61

P
performance • 77
   control file • 83
problems accessing • 103

R
resources, freeing • 69

S
starting CA MIA • 62
starting CA MIM
   overriding initialization values • 62
   stopping CA MIM • 64
   in a VCF Environment • 68
   using global SHUTDOWN commands • 65
   using local SHUTDOWN commands • 66
   systems

Dumps • 110

Index 157
problems with inactive • 100, 103
removing inactive • 69

T
TPCF, description • 9
tracing
  activating and configuring • 110
troubleshooting • 97

V
Virtual Control Files • 82
  migrating to • 59
  problems accessing • 107

Z
z/OS START command • 62