CA MII Data Sharing for z/OS

CA MII Programming Guide

Release 12.0

CA Technologies Product References

This document references the following CA products:

- CA 1® Tape Management (CA 1)
- CA 7 Workload Automation (CA 7)
- CA 11™ Workload Automation (CA 11)
- CA ACF2™ (CA ACF2)
- CA Common Services for z/OS
- CA Intertest™
- CA Jobtrac™ Job Management (CA Jobtrac)
- CA MIA Tape Sharing (CA MIA)
- CA MIC Message Sharing (CA MIC)
- CA MII Data Sharing (CA MII)
- CA MIM™ Resource Sharing (CA MIM)
- CA OPS/MVS® Event Management and Automation (CA OPS/MVS)
- CA PDSMAN® PDS Library Management (CA PDSMAN)
- CA Remote Console™

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.

- Updated the Statements in the Exempt List (see page 41) section.
- Updated the Example in the Job Requeue (see page 53) section.
- Updated the Generate Reports for CA MII (see page 115) section.

Contents

Chapter 3: Advanced Topics 27

Chapter 4: Troubleshooting 79

Chapter 1: Introduction

This section contains the following topics:

CA MII Overview

CA MII is the component of the CA MIM product that improves resource integrity and helps resolve resource conflicts in multiple-system environments. You can use CA MII to protect data sets, programs, or any other file type resource. CA MII does the following:

- Ensures the integrity of resources stored on direct access storage device (DASD) volumes that are shared by multiple systems or images
- Improves performance by providing an efficient way of serializing access to resources that are stored on shared DASD volumes
- Resolves certain resource conflicts automatically
- Accelerates the resolution of conflicts by issuing notification messages to the parties involved in those conflicts
- Prevents most of the common causes of data set damage, such as data set attribute changes, simultaneous updates, partitioned data set (PDS) directory overlays, and unauthorized updates

CA MII is composed of following three facilities:

- **Global Data Integrity Facility (GDIF)**. This facility:
  - Ensures the integrity of shared resources
  - Provides an efficient method for accessing shared resources

- **ENQ Conflict Management Facility (ECMF)**. This facility:
  - Issues messages to help resolve resource conflicts
  - Automatically resolves certain data set conflicts
- **Enhanced Data Set Integrity Facility (EDIF)**. This facility:
  - Prevents programs from changing the attributes of a data set
  - Prevents simultaneous updates that can occur when tasks on the same system do not request serialization
  - Prevents programs from bypassing z/OS serialization when the wrong disposition is specified in the JCL of a job
  - Prevents unauthorized data set updates
  - Prevents unauthorized data set reads

Notation Used for Components

To aid readability, the CA MIM guides use abbreviated names when referring to the components and facilities, except in cases where the abbreviation would obscure the intended meaning. The abbreviations are as follows:

GTAF

Stands for Global Tape Allocation Facility

TPCF

Stands for Tape Preferencing and Control Facility

GCMF

Stands for Global Command and Message Facility

ICMF

Stands for Intersystem Communication Facility

Integrity of Shared Resources

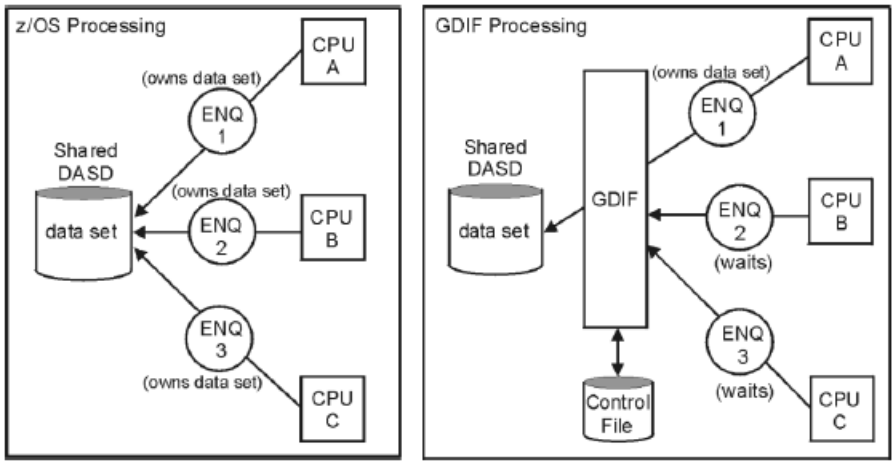
In a single system, z/OS lets only one task at a time have exclusive access to a resource and creates a queue of outstanding ENQ requests if the resource is not available. As the resource becomes available, z/OS lets the task that issued the next ENQ request obtain control of that resource. z/OS uses the RESERVE facility to protect resource integrity across system boundaries. When a task needs a global-type resource, the task issues a special type of ENQ request that identifies the resource and the unit control block (UCB) address of the DASD device on which that resource resides.

When the task obtains control of the resource, z/OS issues a *hardware reserve* for the device where the resource is located. While a hardware reserve is outstanding, tasks on other systems cannot access the reserved device even if they need resources on that device that are different from the resource held by the reserving task. This type of lockout can degrade system performance.

Note: Tasks on the system holding the reserve can still access other resources on the device.

GDIF ensures resource integrity and improves system performance by propagating ENQ and RESERVE requests as global ENQ requests to all systems in your complex. When GDIF intercepts an ENQ or RESERVE request, it sends a record of that request to the CA MIM control files where other systems can access this information. Each system has complete information about which resources are being used, so tasks on different systems cannot update the same resource at the same time.

By propagating ENQ requests to all systems, GDIF provides cross-system integrity for resources not typically protected by hardware reserves. This allows more sharing of resources between systems. By converting hardware reserves to enqueues and propagating them to all systems, GDIF improves system performance. The following figure illustrates how GDIF ENQ processing enhances z/OS ENQ processing and shows how GDIF improves integrity while ensuring serialization:



GDIF also eliminates many occurrences of the *deadly embrace*. A deadly embrace occurs when tasks on different systems cannot complete because they each hold resources or devices needed by the other to complete. When GDIF propagates RESERVE requests as global ENQ requests, z/OS serializes access to the resource on all systems without locking tasks out of the devices containing the resources. You can tell GDIF to eliminate hardware reserves so that the possibility of deadly embraces or lockouts is diminished greatly.

ENQ and RESERVE Propagation Methods

You can select either of these methods of ENQ and RESERVE propagation:

- Automatic propagation of all requests that have a scope of SYSTEMS (PROCESS=ALLSYSTEMS).
- Propagation of only the requests that you designate (PROCESS=SELECT).

Note: We recommend using PROCESS=ALLSYTEMS rather than PROCESS=SELECT: data integrity exposures may occur with PROCESS=SELECT.

No matter which method you choose, you can create a *QNAME list* to provide GDIF with detailed processing instructions. A QNAME list is a series of statements that tell GDIF how to handle ENQ/RESERVE requests based on the major name; that is, QNAME.

You also can provide GDIF with supplemental and more specific processing instructions in an *exempt list*. With the exempt list, for example, you can prevent a request from being propagated if it is issued by a certain job or is for a certain resource.

Hardware Reserves

Because GDIF propagates RESERVE requests to all systems as global ENQ requests, you can improve system performance by eliminating hardware reserves that are associated with RESERVE requests.

GDIF lets you decide which hardware reserves to retain and which to eliminate. You can set a default value that applies to hardware reserves associated with RESERVE requests that GDIF is propagating, and you can override this value for specific QNAMEs and RNAMEs.

How ECMF Manages Resource Conflicts

ECMF is primarily a reporting facility that issues messages to time-sharing option (TSO) users and operators so that they are aware of resource conflicts. Although z/OS may issue conflict messages that identify resources involved in a conflict, these messages do not identify the tasks or TSO users involved in the conflict. Also, these messages are not sent to the TSO users, which makes conflict resolution time-consuming and difficult.

ECMF issues informative conflict messages to TSO users and operators so that you can resolve conflicts before system performance is degraded. You can tell CA MII which resource conflicts you want to receive messages for by specifying ECMF=YES or ECMF=NO for the QNAMEs in the QNAME list. If ECMF detects a conflict for one of these resources, ECMF issues messages that tell the TSO user, the operator, or both what resource is involved in the conflict, what user or task controls the resource, and what system needs that resource. The TSO user or operator then can free the resource or cancel a job to resolve the conflict.

You can determine how often ECMF issues conflict messages for each class of resources. This lets you receive notification immediately for critical requests (such as data set allocations), while receiving notification for other requests only if the conflict persists. You also can modify these values dynamically to change the way ECMF handles conflicts for a class of resources.

How You Requeue Batch Jobs That Cannot Obtain Resources

You can use the ECMF REQUEUE feature to resolve SYSDSN conflicts that occur when a batch job needs a resource that is being used by another job or by a TSO user. This prevents the batch job from tying up an initiator when other jobs are waiting in the same processing queue. If a batch job cannot obtain a resource, then ECMF issues messages to notify the operator (and, if a TSO user controls the resource, to that TSO user) about the conflict. ECMF then moves the batch job to the queue with jobs awaiting execution and puts it in a hold state. When the resource becomes available, ECMF releases the held job so that it can execute.

If ECMF terminates while there are requeued jobs in a held state, then ECMF uses a preset value that you have provided to determine how to handle these jobs. You can tell ECMF to discard checkpoint information for these jobs, release the jobs, retain checkpoint information so that it can be retrieved at the next start up, or ask the operator to decide which of these choices is appropriate.

More information:

Advanced Topics (see page 27)

How You Free Unused Data Sets

You can use the ECMF AUTOFREE feature to free certain data sets that were allocated dynamically and currently are marked not in use.

Data sets that are marked not in use can cause conflicts. For example, if a TSO user issues a compile command from a session, then object and load files are allocated dynamically. These files are marked not in use as soon as the compile finishes, but the files are still allocated to that TSO user. If that TSO user submits a batch job that needs one of these files, then the job is not able to obtain the files until the TSO user issues a deallocation request or ends the TSO session.

When you activate the AUTOFREE feature, ECMF checks to see whether a data set is marked not in use during a conflict. If so, then ECMF automatically frees the data set and notifies the TSO user who allocated the data set about its actions. By doing this, ECMF enables the task that needs this data set to continue.

More information:

Advanced Topics (see page 27)

How You Bypass Integrity Processing to Resolve Conflicts

GDIF provides you with a command that lets you create controlled integrity exposures to resolve cross-system resource conflicts. As soon as the job has finished running, GDIF automatically restores global integrity processing for the affected data sets.

When a job on the local system is involved in a cross-system conflict and that job is the only local job that needs the resource, you can use the DEQJOB command to bypass integrity processing. This command tells GDIF to give the job access to the data set, even though that resource is being protected by GDIF. This lets you create a controlled integrity exposure if, for example, a local job needs to run a report that uses a data set held exclusively by CICS whenever CICS is active.

How EDIF Prevents Data Set Damage

EDIF enhances the integrity and availability of system and user data sets by eliminating most of the common causes of data set damage. EDIF prevents the following problems:

- Attribute damage that occurs when the attributes specified in the JCL of a job, a TSO command, or an application program are different from those in the data set control block (DSCB)
- Attribute damage that occurs when a PDS directory is the target of a sequential output operation
- Simultaneous updates that occur when a job does not request serialization or uses the wrong serialization technique
- Updates by unauthorized programs or by programs that direct output to the wrong file
- Read operations by unauthorized programs

EDIF provides you with a series of statements to enable or negate processing options for data sets. EDIF applies statements to data sets, based on characteristics of the dsname of a data set or on its data set organization.

You can create a processing list containing statements that apply to all data sets, a group of data sets, or a single data set. EDIF also merges statements so that you have greater flexibility in enabling and negating processing options.

How You Prevent Attribute Modification and PDS Directory Overwrites

You can use the EDIF attribute verification feature to prevent these common types of data set damage:

- Attribute damage that occurs when the attributes in the JCL of a job, on a TSO command, or in the data control block (DCB) of a program are different from those in the DSCB of the data set. Your libraries also can be damaged by attribute modification. If you specify the name of a source library that has a record format of FB on a //SYSLMOD DD statement in a LINK step, then the record format of the library is changed to U, making the library unusable.
- Overwrites to PDS directories that occur when a PDS is the target of a sequential output operation. This can happen if you forget to specify a member name for the output from an assembly program. The data set organization of the PDS is changed from partitioned to sequential, making the PDS directory unusable.

You can control exactly how EDIF handles attribute violations. If you want EDIF to detect violations for testing purposes, then you can record violations in a system management facilities (SMF) record and then use the EDIF SMF activity report to analyze statistics about attribute violations. You also can tell EDIF to abend programs to prevent attribute changes from occurring.

EDIF also lets you create lists of programs that are exempted from attribute verification. You can use these lists to identify programs that should not be checked by EDIF.

How EDIF Prevents Simultaneous Updates When Serialization Is Omitted

EDIF prevents simultaneous updates that occur when a job does not serialize access to a data set or uses the wrong technique to serialize access. Most programs rely on JCL to serialize access to data sets on their behalf. When you specify DISP=OLD on a DD statement in your JCL, z/OS issues an ENQ request that obtains exclusive access to the data set for your job. However, when you specify DISP=SHR, your job obtains shared access, which lets other tasks that request shared access use that data set at the same time.

To prevent other tasks from using the data set, the program that your JCL calls must request serialization. If serialization is not requested, then a simultaneous update can occur. To detect and prevent simultaneous updates, EDIF automatically issues an ENQ request whenever a job issues an OPEN request to update a data set. If you want to detect updates for testing purposes, then you can record updates in an SMF record and run an EDIF activity report against those records. You also can prevent simultaneous updates if you tell EDIF to make subsequent tasks wait for access when a data set is being used.

How EDIF Prevents Updates When Programs Extract Data Before Opening Data Sets

EDIF can prevent damage that occurs when a program extracts information before a data set is serialized and a second program updates that data set in the meantime. You can tell EDIF to detect and prevent this type of update. First, determine whether a program extracts information before issuing an OPEN request for a data set. If it does, then tell EDIF to check the JCL for the value DISP=OLD whenever that program tries to update a protected data set. If DISP=OLD is not specified, then EDIF records the update, prevents the update, or does both by abending the program.

How EDIF Prevents Updates by Inappropriate Programs

A data set can become unusable if a program uses the wrong utility to process a data set or writes output to the wrong file. For example, if output from an assembler program is sent to a source file instead of to an object file, then an existing source member may be destroyed. You can use the EDIF *utility verification* feature to detect inappropriate updates. During utility verification, EDIF compares the name of the program that opened the data set with a list of authorized programs that you have provided. If the program is not authorized to perform the update, then EDIF records the violation or abends the task, depending on what options you have enabled.

How EDIF Prevents Unauthorized Read Operations

EDIF also lets you prevent unauthorized programs from reading data sets. During *read verification*, EDIF examines a list of authorized programs that you have provided before letting programs read the affected data set. If a program is not authorized to read the data set, then EDIF takes an action based on other options you have specified. To monitor unauthorized read operations, record violations in the system log or in an SMF record. To prevent unauthorized read operations, you can tell EDIF to abend unauthorized programs.

Chapter 2: Planning Initial Settings

This section contains the following topics:

Overview

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

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

- MIMINIT
- MIMCMNDS
- MIMSYNCH
- MIMQNAME member
- GDIEXMPT member
- EDIPARMS member

Note: For detailed planning information, see the following:

- Advanced Topics (see page 27)
- *Statement and Command Reference Guide*
- *CA MIM Programming Guide*

General Statements

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

- IFSYS
- ENDIF
- INCLUDE
- LOG
- NOLOG

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

Facility-specific Members

Some parameter library members are used only when specific facilities have been activated. The following list describes these considerations by component, facility, and member name:

EDIPARMS

Contains a list of DEFAULT, DSORG, PREFIX, SUFFIX, PATTERN, and DATASET statements describing data set resources monitored by the Enhanced Data Set Integrity Facility (EDIF). This member name is pointed to by the EDIINIT MEMBER=*memname* statement in the MIMINIT member of MIMPARMS. This member is used when EDIF is activated. See the sample EDIPARMS member in data set CAI.CBTDPARM.

GDIEXMPT

Contains a list of DEFAULT, GLOBAL, and LOCAL statements that specifically exempt management of certain resources when the Global Data Integrity Facility (GDIF) is active. The GDIEXMPT member allows you to exclude management of certain resources. This member name is optionally pointed to by the GDIINIT EXEMPT=*memname* statement in the MIMINIT member MIMPARMS. This member is optional. See the sample GDIEXMPT member in data set CAI.CBTDPARM.

MIMQNAME

Contains a list of statements defining which resource QNAMEs you want the Global Data Integrity Facility (GDIF) or the Enqueue Conflict Manager Facility (ECMF) to manage. This member name is pointed to by the MIMINIT QNAME=*memname* statement in the MIMINIT member of MIMPARMS. You must use this member when either GDIF or ECMF is activated. See the sample MIMQNAME member in data set CAI.CBTDPARM.

Plan the MIMINIT Member

The MIMIINT member of the CA MIM parameter data set contains statements that specify initialization values for CA MIM.

Set Global Initialization Values

The MIMINIT statement lets you set initialization values that affect CA MIM regardless of the components and facilities you are using. You specify the MIMINIT statement only in the MIMINIT member of the CA MIM parameter data set.

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

Set Initialization Values for GDIF

The GDIINIT statement lets you set initialization values for the Global Data Integrity Facility (GDIF) of the CA MII component. It can only be specified in the MIMINIT member of the CA MIM parameter data set.

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

Set Initialization Values for EDIF

The EDIINIT statement lets you set initialization values for the Enhanced Data Set Integrity Facility (EDIF) of the CA MII component.

The EDIINIT statement can be specified only in the MIMINIT member of the CA MIM parameter data set. You can use the SETOPTION command to dynamically change the member that EDIF is using to obtain its processing statements.

More information:

Advanced Topics (see page 27)
Plan the EDIPARMS Member (see page 26)

Plan the MIMCMNDS Member

Commands in the MIMCMNDS member of the CA MIM parameter data set are issued at CA MII startup before the local system is synchronized with other systems; therefore, some commands (such as ADDQNAME command) should be placed in the MIMSYNCH member instead of the MIMCMNDS member.

You can set values for statements in the MIMCMDS member to set up initialization values for GDIF, EDIF, and ECMF, including tracing options, display options, status options, and settings to control messages, as well as facility-specific settings.

Note: For more information, see the discussion of the SETOPTION GDIF, SETOPTION EDIF, and SETOPTION ECMF statements in *Statement and Command Reference Guide*.

Plan the MIMSYNCH Member

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

This member name is pointed to in the PROCMIM member by the PROC SYNCH=MIMSYNCH statement. Commands found in this member execute after MIMINIT statements and MIMCMNDS commands have executed, and only after all CA MIM address spaces in the complex have synchronized.

z/VM Considerations

CA MII operates only on the z/OS operating system. There is no corresponding component for the z/VM operating system. If you share resources between both a z/OS and a z/VM system, you need to ensure that the resource is serialized through hardware reserves. You should not convert reserves issued by a z/OS system that protect a resource that is shared with a z/VM system. CA ACF2 and CA Top Secret are examples of products that can run on both z/OS and z/VM.

Plan the MIMQNAME Member

You can use the MIMQNAME member of the CA MIM parameter data set to define which resources GDIF/ECMF manage, or do not manage. The CA MIM parameter library contains two sample members: MIMQNAME and MIMQNAMS. MIMQNAME is designed for clients running with GDIINIT PROCESS=ALLSYSTEMS (recommended). The MIMQNAMS member is designed for customers running with GDIINIT PROCESS=SELECT, which we do not recommend. However, it is still the default value. The MIMQNAME and MIMQNAMS members are updated with each CA MIM release.

If you are installing CA MII for the first time, and are converting from another vendor's global ENQ serialization package, we recommend that you contact CA Technical Support for assistance with the conversion.

If you are installing CA MII for the first time, and are new to sharing DASD across multiple systems, we recommend that you run with GDIINIT PROCESS=ALLSYSTEMS and to use the sample MIMQNAME member. CA Technical Support can assist with setting up the various MIM parmlib members to help ensure the highest level of availability from the beginning.

The MIMQNAME and MIMQNAMS sample members are designed for sites who are fully sharing all DASD, security databases, tape managements systems, and so on, across *all* systems. If you have a configuration where not all DASD/databases/subsystems are to be shared, then you may need to adjust these sample members for your site. We strongly recommend that you contact CA Technical Support for assistance.

Note: For more information about the QNAME statements and associated syntax, see the *Statement and Command Reference Guide* and the chapter "Advanced Topics (see page 27)."

How You Define CA Product ENQ Management Definitions to CA MII

For ENQ management definitions for many CA mainframe products, see the sample MIMQNAME, MIMQNAMS, and GDIEXMPT parmlib members.

As recommendations change, these members are updated to reflect those changes, and the updated members are delivered to customers.

How You Define IBM or other ISV ENQ Management Definitions to CA MII

If you have installed a new IBM or ISV product that issues ENQs or hardware reserves, you should read that product's documentation or contact Technical Support to determine whether CA MII should manage that product's ENQs.

CA delivers ENQ management recommendations for many IBM and ISV products in our sample MIMQNAME, MIMQNAMS, and GDIEXMPT parmlib members. The ENQ management and exemptions statements shown in these samples are what most large scale sites run with, but you may need to customize them for your site.

Based on our experience involving thousands of sites, we have consolidated what we know as general best practice recommendations into our sample CA MII members. You may want to contact the vendor of a product that is generating a given ENQ.

For assistance in evaluating your ENQ workload to determine the optimal specifications for your DASD-sharing MIIplex, contact CA MIM Technical Support.

How You Convert RNL Specifications to QNAME/EXEMPT Statements

IBM and some ISV products specify their cross-system integrity requirements in the form of RNL specifications. You may be able to generate some or all of the QNAME statements in the MIMQNAME member by translating RNL specifications to corresponding CA MII statements.

Plan the GDIEXMPT Member

The GDIEXMPT member defines the exempt list used by GDIF. The actual member name is coded on the GDIINIT EXEMPT= statement in the MIMINIT member. You can use the GDIEXMPT member if you need to make certain datasets or jobs exempt from CA MII global ENQ processing. The sample GDIEXMPT member in the CA MIM parameter library can be used if you are installing CA MIM for the first time and all DASD /databases/subsystems are fully shared across all systems in the MIIplex.

The exempt member is only used for qnames that have GDIF=YES and EXEMPT=YES coded on the QNAME statement in the MIMQNAME member. Or, if a qname is added dynamically if GDIINIT PROCESS=ALLSYSTEMS.

Note: For detailed usage information, see “Advanced Topics“ in this guide. For syntax and additional information on the DEFAULT, LOCAL, and GLOBAL statements of the GDIEXMPT list, see the *Statement and Command Reference Guide*.

Plan the EDIPARMS Member

The EDIPARMS parmlib member lets you determine which data sets are to be processed by EDIF and how they are to be processed. Use the following statements:

- DEFAULT
- DATASET
- PATTERN
- PREFIX
- SUFFIX
- DSORG
- UTILITY

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

Chapter 3: Advanced Topics

This section contains the following topics:

z/OS Resource Serialization

z/OS provides the ENQ facility and the RESERVE facility to serialize access to resources. An ENQ macro specifies a resource name. A RESERVE macro specifies a resource name and a DASD device. The DASD device is specified by the UCB=keyword of the RESERVE macro.

When a task needs access to a resource, z/OS (or the task) issues an ENQ request to tell other tasks which resource is needed and what type of access the task needs. Tasks can request two types of access to a resource:

- *Shared access*, which typically indicates that the task does not need to update the resource. z/OS lets more than one task access a resource concurrently if all of these tasks request shared access.
- *Exclusive access*, which typically indicates that the task needs to update the resource. z/OS lets only one task at a time have exclusive access to a resource. No other task can have access to the resource, regardless of whether it needs exclusive or shared access.

z/OS makes a requester wait when:

- A task has shared access to a resource and a later request needs exclusive access to that resource
- A task has exclusive access to a resource and a later request needs shared or exclusive access

z/OS places the task that currently holds the resource at the top of the queue. Subsequent tasks are listed on a first-come, first-serve basis. z/OS makes subsequent tasks wait until they can obtain the type of access they need for that resource.

How You Identify Resources on ENQ and RESERVE Requests

A task uses a multiple-part naming convention on an ENQ or RESERVE request to identify a resource that it needs and to indicate the extent to which access should be serialized for that resource.

The following are the parts of the name used on ENQ and RESERVE requests:

qname

Indicates what class contains the resource. For example, if a task allocates a data set, then it specifies SYSDSN as the QNAME.

rname

Indicates the name of the resource needed by the task. For example, when a task needs to allocate a data set, the task usually uses the dsname of the data set as the RNAME.

scope

Indicates how z/OS serializes access to the resource:

STEP

Serializes access only in the address space of the task

SYSTEM

Serializes access in the current processor or image; that is, in the local system only

SYSTEMS

Serializes access among all of the systems in the complex

To serialize access to a resource, tasks must issue ENQ requests that match all three: QNAME, RNAME, and scope. All three parts of the name are used to identify the resource. ENQ/RESERVE requests for resources that do not match all three parts are considered by z/OS to be requests for different resources. GDIF and ECMF work with the same QNAMEs, RNAMEs, and scopes to identify resources.

RESERVE requests are always treated as having SCOPE SYSTEMS. Usually, a program makes a RESERVE request by using the RESERVE macro of z/OS. Sometimes, a program makes a RESERVE request by using the ENQ macro but also using the UCB keyword of the ENQ macro. To reduce confusion, assume that programs always use the RESERVE macro to make RESERVE requests and always use the ENQ macro to make ENQ requests; that is, assume that programs never use the UCB keyword of the ENQ macro.

CA MII has no effect on ENQ requests that specify STEP as the request scope.

Resource Serialization

You can tell GDIF and ECMF to perform these types of processing for resources:

- Using GDIF, propagate ENQ and RESERVE requests to all systems as global ENQ requests.
- Using ECMF, issue messages about resource conflicts.
- Using GDIF, eliminate hardware reserves by converting them to multisystem requests.
- Using ECMF, resolve certain conflicts automatically.

GDIF and ECMF both use a QNAME list to obtain information on how to manage resources. A QNAME list is a series of QNAME statements that defines classes of resources (by the QNAME, or major name, used on the ENQ or RESERVE request), what type of processing those classes should receive, and whether any of the resources in those classes should be processed differently.

If necessary, you can exercise refined control of GDIF propagation. The propagation can depend upon:

- The resource-name of the request, which is specified in RNAME keywords of CA MII statements
- The identity of the requester, which is specified in JOBNAME keywords of CA MII statements

To exercise refined control, you create an exempt list to specify supplemental instructions.

If you can let all requesters treat the resource in the same way, then you should avoid use of the JOBNAME keyword mentioned above.

The QNAME List and Exempt List

GDIF and ECMF are controlled by the QNAME list. The QNAME list affects both GDIF and ECMF; it has no affect on EDIF. The QNAME list is permanently specified in the MIMQNAME member and can be temporarily changed using any of the following commands:

- ADDQNAME
- ALTER
- DELQNAME

GDIF is also controlled by the exempt list. The exempt list is permanently specified in the GDIEXMPT member, which contains these statements:

- DEFAULT
- GLOBAL
- LOCAL

You can temporarily change the exempt list by using the EXEMPT command.

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

Creating a QNAME List

You create a QNAME list by placing QNAME statements in the MIMQNAME member of the CA MIM parameter data set.

A QNAME list is the primary source of processing instructions for both ECMF and GDIF. If CA MII is running in PROCESS=SELECT mode, then new QNAMEs are temporarily added to the list whenever system operators use the ADDQNAME command of CA MII. If CA MII is running in PROCESS=ALLSYSTEMS mode, then new QNAMEs are also added whenever a program uses a SCOPE=SYSTEMS ENQ macro or a RESERVE macro, and specifies a QNAME that is not already in the list. PROCESS=ALLSYSTEMS is the recommended value for the PROCESS= statement.

When ADDQNAME is used to add the new QNAME, the operator can specify any operand; that is, ECMF and REPORTCYCLE values can be specified. When the ENQ or RESERVE adds the new QNAME, default values appropriate for the current environment are chosen for some operands; for example, the default is GDIF=YES,ECMF=NO when GDIF is active

Note that QNAMEs added by an ADDQNAME command or by an ENQ/RESERVE macro are added temporarily. They are discarded when you do a global restart of all CA MII systems.

If you want to use some member name other than the CA MII default for your QNAME list, then you can choose another name. You can then use the QNAMES= keyword of the MIMINIT statement to override the default of QNAMES=MIMQNAME. For example:

```
MIMINIT QNAMES=MYQNAMES
```

We recommend that you use the same member for all systems.

CA MII generally requires that GDIF keywords of the QNAME statement must have the same value on all systems. This is not a requirement for ECMF keywords.

Reviewing the Sample QNAME List

A sample QNAME list is provided in member MIMQNAME of the CAI.CBTDPARM data set. The MIMQNAME member is to be used if you are running with GDIINIT PROCESS=ALLSYSTEMS, which we recommend. The MIMQNAMS member is to be used if you are running with GDIINIT PROCESS=SELECT.

The following example shows some of the QNAME statements in this sample MIMQNAME member:

```
SYSDSN GDIF=YES SCOPE=SYSTEM EXEMPT=YES ECMF=YES REPORTAFTER=0,
    REPORTCYCLE=60 TRACE=NONE
SYSVSAM GDIF=YES SCOPE=SYSTEMS EXEMPT=YES ECMF=NO TRACE=NONE
SYSIEWLP GDIF=YES SCOPE=RESERVES EXEMPT=NO ECMF=YES,
    REPORTAFTER=60 REPORTCYCLE=60 TRACE=NONE
SPZAPLIB GDIF=YES SCOPE=RESERVES EXEMPT=NO ECMF=YES,
    REPORTAFTER=60 REPORTCYCLE=60 TRACE=NONE
EDIDSN GDIF=YES SCOPE=SYSTEMS EXEMPT=NO ECMF=YES RPTAFTER=0,
    REPORTCYCLE=60 TRACE=NONE
SPFEDIT GDIF=YES SCOPE=SYSTEMS EXEMPT=YES ECMF=YES,
    REPORTAFTER=0 REPORTCYCLE=60 TRACE=NONE
```

Deciding Which Resources to Process

The best way to decide what resources GDIF, ECMF, or both should process is to activate GDIF, ECMF, or both to monitor resource activity through the CA MII display counts commands, and then reevaluate the QNAME list. Most sites can use the sample QNAME list provided in the MIMQNAME member to test the processing of GDIF and ECMF. Review the following guidelines when choosing QNAMEs that GDIF and ECMF should process:

- The resources in that class are used on more than one system.
- All ENQ and RESERVE requests for a specific resource use the same RNAME.
- RESERVE requests for that QNAME (if any) are long enough to degrade system performance if they are not propagated as global ENQ requests. In general, you should not place a QNAME in the QNAME list if its RESERVE requests are short.

For QNAMEs used by vendor-supplied software, you should contact the vendor for specific recommendations about QNAMEs and resources used by the product.

Changing the QNAME List

The first system active in a complex is the system that reads the MIMQNAME parameter member and builds the QNAME list for the entire complex. The second through *n*th systems activated in the complex have the QNAME list passed to them during synchronization, and the contents of the MIMQNAME member are ignored with respect to the QNAME list already in use by the complex. This implies that changing the MIMQNAME parameter member results in changing the QNAME list in use by the complex only if accompanied by a global shutdown.

The only way to change the QNAME list in use by an active complex is to use the ADDQNAME, ALTER, and DELQNAME commands. These commands make changes to the active complex dynamically, but their effect is temporary. Changes made with these commands persist until the last instance of CA MII in the complex is shut down.

We recommend the following procedures for making permanent QNAME list changes:

1. Make the desired change to the active complex using the ADDQNAME, ALTER, or DELQNAME commands.
2. Promptly make the corresponding updates in the MIMQNAME member.

   Note: This procedure assumes you have GDIINIT MISMATCHQNAME=ACCEPT specified or defaulted.

-- OR --

1. Make the desired modifications in the MIMQNAME member.
2. Shut down CA MII on all systems in the complex.
3. Restart CA MII on all systems in the complex.

Global Data Set Integrity Facility

Resource classes processed by GDIF and ECMF are known as *managed resources*. The names of these classes are specified in the QNAME list (see Creating a QNAME List in this chapter). You can tell GDIF to add classes to the list by running in PROCESS=ALLSYSTEMS mode. When GDIF adds a class while running in PROCESS=ALLSYSTEMS mode, it specifies ECMF=NO for the newly added class: ECMF gives no contention warnings for the newly added class. If you want to receive a contention warning for the newly added class, then you should include the class in the MIMQNAME member, instead of adding the class while running in PROCESS=ALLSYSTEMS mode.

When GDIF serializes a resource class, it propagates requests for that resource to other CA MII systems. The GDIF keyword of the QNAME statement controls this. The exempt list lets you specify that some resource names of the class be propagated, while other names of the class are not.

When GDIF serializes a resource class, it can optionally suppress hardware RESERVEs; the RESERVES keyword of the QNAME statement controls this. When the hardware RESERVE is suppressed, the RESERVE macro is considered to be *converted*. When you convert a RESERVE, you rely on GDIF to provide the serialization: other systems can continue to access the device. Other systems can read or update data sets that reside on that device, but GDIF ensures that other systems cannot gain ownership of the resource whose name was specified on the RESERVE macro. By converting the RESERVE, GDIF lets the RESERVE macro serialize individual data sets instead of "locking up" every data set on the device. Note that the exempt list, described later in this chapter, lets you specify that some resource names of the class should convert the RESERVE, while other names of the class should use hardware RESERVE serialization.

GDIF Processing Modes

The processing mode you select determines whether GDIF propagates all requests with a scope of SYSTEMS or only the requests you designate in the QNAME list. To select a processing mode, specify a value for the PROCESS parameter on the GDIINIT statement.

The following are valid values:

ALLSYSTEMS

Tells GDIF to process all requests with a scope of SYSTEMS, and any other requests designated in the QNAME list. When an ENQ or RESERVE request is issued with a scope of SYSTEMS, the associated QNAME is added dynamically to the QNAME list.

SELECT

Tells GDIF to propagate only the requests that you designate in the QNAME list. This is the default.

Important! With GDIINIT PROCESS=SELECT, administrators currently must monitor ENQ activity to ensure that the ENQs were properly defined to avoid integrity exposure for ENQs with SCOPE=SYSTEMS. Because of the much lower maintenance requirements associated with GDIINIT PROCESS=ALLSYSTEMS mode, and the elimination of multi-systems data integrity exposures with this mode, We recommend that you use GDIINIT PROCESS=ALLSYSTEMS.

Note: You must specify the same value on all systems. This happens automatically if you use a single member for all systems (recommended).

For example, if you want GDIF to propagate all requests that have a scope of SYSTEMS, then you could specify this GDIINIT statement in the initialization member:

```
GDIINIT PROCESS=ALLSYSTEMS
```

GDIF uses the following values for the parameters on the dynamically added QNAME statement:

- GDIF=YES
- SCOPE=SYSTEMS
- EXEMPT=YES
- TRACE=NONE
- ECMF=NO

Use the DISPLAY QNAMES operator command of CA MII to show all the classes of the QNAME list, including the dynamically added classes that result when you run in ALLSYSTEMS mode.

How You Eliminate Hardware Reserves

You need to indicate whether GDIF should eliminate or retain hardware reserves by default when GDIF propagates RESERVE requests as global ENQ requests. You can do this by specifying one of the following values on the RESERVES parameter on the GDIINIT statement:

CONVERT

Tells GDIF to eliminate hardware reserves for managed resources unless you specify otherwise in the QNAME or exempt lists.

KEEP

Tells GDIF to retain hardware reserves for managed resources unless you specify otherwise in the QNAME or exempt lists.

Default: CONVERT

These values have no effect on whether GDIF propagates RESERVE requests to other systems as global ENQ requests. For example, to retain hardware reserves by default, you could specify this GDIINIT statement in the initialization member:

```
GDIINIT RESERVES=KEEP
```

Important! The RESERVES value of the GDIINIT statement can be overridden by the RESERVES value of the QNAME statements. The QNAME statements, in turn, can be overridden by the RESERVES value of the exempt-list (GLOBAL statements). When the operator enters an ADDQNAME command, a QNAME statement must be provided (and a RESERVES value is provided, either explicitly or by omission).

How You Control How GDIF Handles Requests

You can use the following QNAME statement parameters to control the way GDIF handles requests for a class of resources:

GDIF

Tells GDIF whether to propagate ENQ and RESERVE requests for that class.

SCOPE

Tells GDIF which ENQ and RESERVE requests to propagate for that class. You can specify:

ALL

Propagates all (RESERVES, SYSTEM, and SYSTEMS) ENQ and RESERVE requests for the specified class of resources

RESERVES

Propagates requests that produce hardware reserves

SYSTEM

Propagates SYSTEM ENQ requests for resources used by tasks on the same system

SYSTEMS

Propagates requests that produce hardware reserves and SYSTEMS ENQ requests for resources that may be used by several systems

Default: SYSTEMS

EXEMPT

Tells GDIF whether to use the exempt list to obtain supplemental and more specific processing information for that class.

RESERVES

Tells GDIF whether to eliminate or retain hardware reserves for this QNAME. If you specify this parameter, then this value overrides the value you specify on the RESERVES parameter on a GDIINIT statement.

Note: You can use the RESERVES parameter on a GLOBAL statement in the exempt list to override this value, based on the name of a resource.

For example, the following QNAME statement tells GDIF to propagate reserve requests and eliminate the reserve for resources requested under SYSIEWLP, unless otherwise directed in the exempt list:

```
SYSIEWLP GDIF=YES SCOPE=RESERVES EXEMPT=YES RESERVES=CONVERT
```

By default, GDIF propagates ENQ and RESERVE requests for a class of resources whenever you specify a QNAME statement for those resources. If you do not want to propagate requests for those resources, then specify GDIF=NO on the QNAME statement.

Typically, GDIF propagates all requests for a QNAME whenever you specify a QNAME statement for that class of resources. However, you can tell GDIF to propagate only certain types of requests for that QNAME.

Hardware Reserves for a Class of Resources

To change the way GDIF handles hardware reserves for a class of resources, you can specify a value for the RESERVES parameter on a QNAME statement. If you specify this parameter, then this value overrides the value that you specified on the RESERVES parameter on a GDIINIT statement.

To eliminate hardware reserves for a QNAME, specify RESERVES=CONVERT on that QNAME statement. This value tells GDIF to eliminate hardware reserves for that class of resources, no matter what value you specified on the GDIINIT statement. If you want to retain hardware reserves, then specify RESERVES=KEEP instead.

PDSE Type Data Set Considerations

The most important thing to remember is that serialized sharing of PDSE type data sets is limited to systems that belong to a single sysplex when GRS is active. There is no serialized protection when PDSE type data sets are shared with systems outside of a single sysplex. You can successfully share PDSE type data sets in a monoplex with the following settings:

- GRS=NONE
- PDSESHARING(NORMAL)
- CA MIM manages SYSZIGW0 and SYSZIGW1

Important: Attempts to share PDSEs across sysplex boundaries may result in data integrity exposures, corrupted PDSEs, or loss of data.

PDSE Resource Sharing

CA MII is fully compatible with PDSE resource sharing. CA MII does not modify, enhance, or restrict PDS/E processing in any way. PDSE sharing is limited to a single sysplex in a pure global GRS environment, and the same restrictions apply, even when you are running CA MII. In this regard, CA MII neither enhances nor restricts PDSE sharing.

However, customers have shared DASD across sysplex (GRSPLEX) boundaries for years using CA MII. You cannot assume that CA MII provides the same enhanced sharing capabilities with PDSEs.

While CA MII propagates ENQ and RESERVE requests outside sysplex boundaries, and thus allows sharing of DASD outside the sysplex boundaries, PDSE ENQ requests are a notable exception. Because the PDSE code issues its SYSZIGW0 and SYSZIGW1 ENQs with RNL=NO, CA MII does not process these ENQs, and thus does not propagate these requests outside the sysplex. However, CA MIM propagates RNL=NO type ENQs running in a monoplex with GRS=NONE and PDSESHARING(NORMAL), including SYSZIGW0 and SYSZIGW1.

When using PDSEs, there are two possible modes of sharing:

- PDSESHARING(NORMAL)
- PDSESHARING (EXTENDED)
- You specify the sharing mode in the IGDSMSxx member of SYS1.PARMLIB or in a data set in the logical PARMLIB. You can use the MVS "DISPLAY SMS,OPTIONS" command to determine how the PDSESHARING option has been set in your active SMS address space on a given system.

PDSESHARING(NORMAL)

When sharing PDSEs in NORMAL mode, the PDSE can be shared by all systems for input only. This is the default mode setting. Only one writer is allowed per complex. Subsequent concurrent openers for output are terminated with a S213-70 ABEND.

When you specify PDSESHARING(NORMAL), the SYSZIGW0 and SYSZIGW1 ENQs are used in order to maintain PDSE integrity, and XCF services are not used. However, the PDSE code issues its SYSZIGW0 and SYSZIGW1 ENQs with RNL=NO specified. This precludes CA MII from managing these ENQs when they are issued on a system that is a member of a sysplex. This can cause integrity exposures in some environments.

PDSESHARING(EXTENDED)

When you specify PDSESHARING(EXTENDED), a PDSE can be shared for input and output in a single sysplex.

In EXTENDED mode, XCF services are used to pass information between systems in a single sysplex to maintain read and write integrity. Therefore, the sharing of PDSEs in EXTENDED mode requires a sysplex and is limited to the systems in the sysplex by definition.

With PDSESHARING(EXTENDED), the sharing of PDSEs is limited by XCF connectivity, not by the scope of the SYSZIGW0/1 ENQs. Therefore, the sysplex restriction in this case is due to the IBM PDSE architecture, not by the global serialization product (for example, CA MII).

RNL=NO and GRSRNL=EXCLUDE

When IBM introduced basic sysplex, they required that at least a bare-bones global GRS be active on the systems that are part of a sysplex. In order for GRS and CA MIM to peacefully co-exist in a sysplex, IBM added the EXCLUDE value to the GRSRNL= parameter in IEASYSxx, and added the RNL=YES/NO parameter to the ENQ macro and parameter list.

The GRSRNL=EXCLUDE parameter setting caused GRS to demote all SYSTEMS scope ENQs to SYSTEM scope ENQs. This allowed GRS to be active in a system with CA MII also active, and prevented GRS from propagating ENQ requests that were already being propagated by CA MII.

However, IBM required serialization of resources that were sysplex-specific (ENQs with a scope of the local sysplex). Also, IBM required this resource serialization, particularly during sysplex initialization, when CA MII serialization might not be available.

To satisfy these requirements, IBM added the RNL=YES/NO parameter on the ENQ macro and parameter list. The default is RNL=YES. RNL=YES specified that an installation's RNL lists and their GRSRNL=EXCLUDE value, if specified, would be honored.

Specifying RNL=NO overrides both the installation's RNL specifications and their GRSRNL=EXCLUDE specification. Therefore, if an ENQ is issued with a scope of SYSTEMS and RNL=NO, GRS propagates this ENQ to all of the systems in the GRSPLEX (that is. all of the sysplex systems), even when GRSRNL=EXCLUDE is specified.

RNL=NO was intended to be used only on ENQ requests for resources that were sysplex-specific (that is, the scope of the local sysplex), or for resources that needed to be serialized during system and sysplex initialization.

To determine the GRSRNL value you are running with on a given system, issue the following z/OS command:

```
DISPLAY GRS,ALL
```

When an ENQ is issued on a sysplex system, and that ENQ has RNL=NO coded, CA MII bypasses those ENQ requests and passes them along to GRS. Recall that, in a sysplex, global GRS must be active, and all members of the sysplex must be members of the same GRSPLEX. This guarantees that the RNL=NO requests are propagated to all sysplex systems by GRS if the ENQ is scope SYSTEMS.

Unfortunately, the original IBM RNL=NO documentation did not express this intended function and was somewhat misleading, telling the user to "Use RNL=NO only when you are sure that you never want the scope value to change". IBM has since created APAR OW30729 to clarify the documentation on the RNL parameter.

Integrity Exposure Issues

For CA MII customers who are sharing PDSEs in a MIMplex that does not equal a single sysplex, integrity exposures can occur between the sysplex systems and the systems or sysplexes outside the local sysplex. Because the SYSZIGW0 and SYSZIGW1 ENQs are issued with RNL=NO, CA MII bypasses those ENQs and passes them on to GRS. All of the systems in the originating sysplex are aware of these ENQs because GRS propagates them to the systems in the local sysplex. However, because CA MII bypasses these requests, the systems outside the originating sysplex are unaware of these ENQs.

The following table illustrates in what environments it is safe to share PDSEs across all systems in the MIMPLEX:

DASD Sharing Environment	PDSESHARING Mode	
	Normal	Extended
Non-sysplex systems	Yes	n/a
Single sysplex systems	Yes	Yes
Single sysplex plus non-sysplex systems	No (see Notes)	No
Multiple sysplexes plus non-sysplex systems	No (see Notes)	No
Multiple sysplexes	No (see Notes)	No

Notes:

- To share PDSEs in non-sysplex environment, do one of the following:
  - Specify GDIINIT PROCESS=ALLSYSTEMS
  - Specify the following statements in the MIMQNAME member:

    ```
    SYSZIGW0 SCOPE=SYSTEMS ECMF=NO GDIF=YES EXEMPT=NO
    SYSZIGW1 SCOPE=SYSTEMS ECMF=NO GDIF=YES EXEMPT=NO
    ```
- PDSE sharing in mixed sysplex environments is not safe, due directly to the inappropriate use of RNL=NO on the SYSZIGW0 and SYSZIGW1 ENQs. See the IBM APAR OW30729 for clarification of this issue.

At the present time, we recommend that you not manage PDSEs with EDIF.

The Exempt List

The exempt list affects only GDIF; it affects neither EDIF nor ECMF. The exempt list is ignored if every statement of the QNAME list specifies EXEMPT=NO. In this case, the QNAME list provides all the information that controls GDIF. The exempt list is not required to be identical on every system; however, system management is easier if every system has the same exempt list.

The exempt list provides supplemental processing instruction for GDIF. If you want to use the exempt list to give GDIF supplemental and more specific information about processing a QNAME, then you need to specify EXEMPT=YES on that QNAME statement. Doing this allows you to use RNAMEs and job names to determine whether an ENQ request should be propagated. You also can change the way GDIF handles hardware reserves, based on the QNAME and RNAME on the RESERVE request.

You can use the exempt list to give GDIF special processing instructions, based on the RNAME of a resource or on the name of the job that issued the ENQ request. If you want to use other criteria (such as device name, volume serial number) to give GDIF special processing instructions, you can code the optional GDIXMPXT exit routine.

How You Create an Exempt List

If you want to propagate ENQ and RESERVE requests for only some of the resources that are requested through the same QNAME, or if you want to handle hardware reserves for some of these resources differently, then you can create an exempt list. You need to specify the name of this member on the EXEMPT parameter on the GDIINIT statement.

By default, GDIF uses the member GDIEXMPT of the CBTDPARM data set, which contains an exempt list. GDIF uses the exempt list only when one of the following conditions exists:

- GDIF=YES and EXEMPT=YES were specified on a QNAME statement.
- The default values GDIF=YES and EXEMPT=YES are being used for QNAME SYSDSN.
- GDIF is running in ALLSYSTEMS mode, and GDIF dynamically adds a QNAME statement to the QNAME list.

If you do not want to exempt anything or create an exempt list, then specify the statement GDIINIT EXEMPT=NONE.

When coding exempt list statements, keep the following in mind:

- Each exempt list statement begins with a statement name, such as DEFAULT, and contains additional parameters.
- You can specify parameters in any order and use non-ambiguous truncations for parameters.
- You can specify a QNAME or an RNAME in either character or hexadecimal format. If you use character format and the QNAME or RNAME contains embedded blanks, you need to enclose that QNAME or RNAME in single quotation marks.
- All standard rules on CA MIM syntax, commenting, and continuation apply.

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

Requests That Should Not Be Propagated

You may not want to propagate requests for the following data set-type resources:

- System data sets that are replicated on different systems. Because each system has a unique copy of these data sets, you may not want to issue global ENQ requests for them.
- Shared data sets that need to be accessed concurrently from different systems and are serialized across systems by the tasks that use them. If you are in doubt, check with the product vendor.
- ISPF LIST, LOG, and PROFILE data sets that are associated with a TSO session for a specific user. You may not want to propagate requests for these resources because TSO users usually log on to one system at a time.
- Resources for which you want controlled integrity exposures. For example, the CICS or IMS products can keep data sets open indefinitely if you run these products around the clock. Some sites then run reports or other read-only jobs on other systems because this is the only alternative to stopping the product, running the job, and then restarting the product. If you are controlling integrity exposures manually and you do not want to change the procedures of your site, then you need to tell GDIF not to propagate requests for those resources.
- Resources that are serialized through RESERVE by both z/OS and z/VM. You should not convert reserves issued by a z/OS system that protect a resource that is shared with a z/VM system. CA ACF2 and CA Top Secret are examples of products that can run on both z/OS and z/VM.

Statements in the Exempt List

If you specify EXEMPT=YES on a QNAME statement, then GDIF uses the statements in the exempt list to decide whether to propagate requests for that QNAME. GDIF looks at two types of statements in the exempt list: a statement that tells GDIF how to process requests by default and statements that override processing for specific resources or jobs.

You can use the following exempt list statements to indicate how GDIF can process resources:

DEFAULT

GDIF decides whether to propagate the ENQ and the RESERVE requests by default when a task requests a QNAME with EXEMPT=YES.

GLOBAL

GDIF propagates the ENQ and RESERVE requests for a certain resource or job. You also can use this statement to change the way GDIF handles hardware reserves for a resource.

LOCAL

GDIF does not propagate the ENQ requests for a certain resource or job. The LOCAL statements do not apply to RESERVE requests, except when EXEMPTRESERVES=YES has been specified. Even when this option is set to YES, RESERVES=CONVERT/KEEP cannot be specified on a LOCAL statement (and RESERVES=KEEP is implied).

You can specify only one DEFAULT statement, and it must be the first statement in the exempt list. You can specify as many GLOBAL and LOCAL statements as required to override processing for specific resources or jobs.

Each DEFAULT statement parameters affect how GDIF propagates requests by default:

JOB

Tells GDIF whether to propagate requests by default, no matter what job issued the request.

RESOURCE

Tells GDIF whether to propagate requests by default, no matter what RNAME is specified on the request.

Each parameter affects propagation. However, the combined effect of the parameters determines whether GDIF propagates requests by default and how you can override this default processing:

- When you specify GLOBAL on both parameters, GDIF propagates requests globally. You can override this default processing for ENQ requests using the LOCAL statements.
- When you specify LOCAL on one or both parameters, GDIF treats requests as local requests and GDIF does not propagate the requests to other systems. You can override this default processing using the GLOBAL statements.

GDIF uses a maximum of two matching LOCAL statements, GLOBAL statements, or both for each request: one statement that contains a matching RNAME, and another statement that contains a matching job name. If more than one statement contains a matching RNAME or job name, then GDIF selects one of these matching statements and GDIF ignores all other matching statements. Therefore, understanding how GDIF chooses among matching statements is a necessity.

NOTE: If an ENQ/RESERVE request matches both a LOCAL job statement and a GLOBAL resource statement, then the LOCAL job statement takes precedence. If an ENQ/RESERVE request matches both a GLOBAL job statement and a LOCAL resource statement, then the LOCAL resource statement takes precedence. This logic is true regardless of what is coded on the DEFAULT statement.

Propagate QNAME Requests

To propagate most QNAME requests, place the following statement in the GDIEXMPT member:

```
DEFAULT RESOURCE=GLOBAL JOB=GLOBAL
```

This statement tells GDIF to propagate requests by default. Use LOCAL statements to indicate which ENQ requests should not be propagated.

Propagate Requests for Selected Resources or Jobs

To propagate requests as a general rule and use RNAMEs or job names to identify requests that should not be propagated, specify one of the following DEFAULT statements:

- DEFAULT RESOURCE=GLOBAL JOB=LOCAL

  This statement tells GDIF to propagate requests for only the jobs named on GLOBAL statements.

- DEFAULT RESOURCE=LOCAL JOB=GLOBAL

  This statement tells GDIF to propagate requests for only the resources named on GLOBAL statements.

Propagate Certain Requests from Certain Jobs

To propagate requests only if a certain job needs a certain resource, specify the following statement in the GDIEXMPT member:

```
DEFAULT RESOURCE=LOCAL JOB=LOCAL
```

This statement tells GDIF to propagate requests only if you specified two GLOBAL statements: one for the resource and another for the job requesting that resource. Therefore, you must specify both the RNAME and job name on separate GLOBAL statements before GDIF propagates a request.

Override the Default Processing of GDIF

You can use the LOCAL and GLOBAL statements to override the values on your DEFAULT statement. Use the LOCAL statement to treat certain ENQ requests as local requests and the GLOBAL statement to propagate certain ENQ and RESERVE requests globally.

The number of LOCAL or GLOBAL statements you need to use depends on what values you specified on your DEFAULT statement. If you specified GLOBAL on both DEFAULT statement parameters, then you can specify a single LOCAL statement to stop GDIF from propagating an ENQ request.

However, to propagate requests when you specified LOCAL for both parameters, you need to specify two GLOBAL statements: one statement that overrides the RESOURCE parameter, and a second statement that overrides the JOB parameter.

Override Default Processing for a Certain Resource

To change the way GDIF handles requests for a certain resource, specify the RNAME of that resource on the RNAME parameter of a LOCAL or GLOBAL statement. When that RNAME is specified on a request, GDIF overrides the value for the RESOURCE parameter on your DEFAULT statement.

For example, if you specified RESOURCE=GLOBAL on a DEFAULT statement, then you can treat all requests for the SYS1.LINKLIB data set as local requests, no matter what QNAME is used, by placing the following statement in the GDIEXMPT member:

```
LOCAL RNAME=SYS1.LINKLIB
```

To override the default processing of GDIF if a certain QNAME is specified on the request, specify that QNAME on the QNAME parameter. For example, you can tell GDIF to treat requests for the SYS1.LINKLIB data set as local requests only if the QNAME SYSDSN is specified by placing the following statement in the GDIEXMPT member:

```
LOCAL QNAME=SYSDSN RNAME=SYS1.LINKLIB
```

Override Default Processing for a Certain Job

To change the way GDIF handles requests issued by a certain job, specify the name of that job on the JOB parameter of a LOCAL or GLOBAL statement. When that job issues a request, GDIF overrides the value for the JOB parameter on your DEFAULT statement.

For example, if you specified RESOURCE=GLOBAL and JOB=LOCAL on your DEFAULT statement, then you can propagate requests from job MAINT by placing this GLOBAL statement in the GDIEXMPT member:

```
GLOBAL JOB=MAINT
```

Wildcard Characters

You can use the following wildcard characters to affect QNAME processing:

- Use the # character in the RNAME, job name, or QNAME. This character matches a single character in the name.

  For example, to match a LOCAL statement for the SYS1.LINKLIB data set with ENQ requests that use the QNAMEs DATAMNGR, DATAMNRR, or DATAMNSR, you could specify this statement in the GDIEXMPT member:

  ```
  LOCAL QNAME=DATAMN#R RNAME=SYS1.LINKLIB
  ```

- Use the * wildcard character with an RNAME or a job name. This character matches the remainder of the name.

  For example, to match a GLOBAL statement with ENQ and RESERVE requests issued by the jobs TEST1, TEST2, and TESTPAY, you could specify the following statement in the GDIEXMPT member:

  ```
  GLOBAL JOB=TEST*
  ```

  Note: If you specify TEST without the asterisk, you still get all jobs that begin with the letters TEST, since there is an implied asterisk after the job name. Therefore, if you do not want all jobs matching the name TEST with any suffix, then you should explicitly specify the job name on your statement. This is also true for RNAMEs. If you specify a "*" before the end of the RNAME, it is automatically replaced by a "#" (single character): "*" is invalid anywhere but at the end of the RNAME.

- Use the "?" character in an RNAME. This character matches the remainder of the current index level. However, the "?" character does not *wor*k if specified at the end of the RNAME.

  For example, suppose that you want to match a LOCAL statement with ENQ requests in which SYS1 is specified in the first index level of the RNAME and LOAD in the third index level of the RNAME. To do this, you could specify the following statement in the GDIEXMPT member:

  ```
  LOCAL RNAME=SYS1.?.LOAD
  ```

Notes:

- GDIF reorders the LOCAL and GLOBAL statements in the exempt list according to how many non-wildcard characters each name contains and in what position wildcard characters are specified. Therefore, you should specify as many leading, non-wildcard characters as possible in a job name, RNAME, or QNAME.
- For information on how GDIF reorders the exempt list, see How GDIF Reorders the Exempt List at Startup Time (see page 47).

Change the Way GDIF Handles Hardware Reserves for a Resource

If you want to change the way GDIF handles hardware reserves for a specified resource, then you can specify a value for the RESERVES parameter on a GLOBAL statement. You can use this parameter to override the values you specify for the RESERVES parameter on a QNAME or GDIINIT statement.

Specify one of these values on the GLOBAL statement to indicate how GDIF should handle hardware reserves for this resource:

- RESERVES=CONVERT

  Tells GDIF to eliminate hardware reserves associated with RESERVE requests for this resource.

- RESERVES=KEEP

  Tells GDIF to retain hardware reserves associated with RESERVE requests for this resource. RESERVES=KEEP on a GLOBAL statement can cause deadlocks because GDIF still propagates the ENQs for this resource.

  For example, suppose you specified RESERVES=KEEP on a QNAME statement. To eliminate hardware reserves when the QNAME/RNAME combination SYSIEWLP/SYS1.LINKLIB appears on a RESERVE request, you could specify this statement in the GDIEXMPT member:

  ```
  GLOBAL QNAME=SYSIEWLP RNAME=SYS1.LINKLIB RESERVES=CONVERT
  ```

Apply LOCAL Statements to RESERVE Requests

Do not attempt to specify RESERVES=CONVERT or RESERVES=KEEP on a LOCAL statement. The RESERVES keyword is not supported on a LOCAL statement. Instead, use the SETOPTION GDIF EXEMPTRESERVES command to control whether the LOCAL statements should be applied to RESERVE requests.

For example, assume that NO is specified on SYSA, and JOB1 issues the first RESERVE request for an exempted resource. This results in the RESERVE being converted to a global enqueue. Assume that YES is specified on SYSB, and JOB2 issues a subsequent request on SYSB for the same exempted resource. The request of JOB2 is not propagated. As a result, both JOB1 and JOB2 could have concurrent ownership of the resource, possibly corrupting the resource.

Important! The value of the RESERVES=attribute command must be set consistently across all systems in your complex, or an integrity exposure can occur.

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

Prevent Statements from Being Negated

By default, GDIF lets you dynamically negate any LOCAL or GLOBAL statement. You can use the EXEMPT command to do this. This causes GDIF to ignore that statement when processing requests.

To prevent a statement from being negated, you need to specify PERMANENT=YES on that statement. For example, to prevent the LOCAL statement for the SYS1.LINKLIB data set from being negated, you could specify this statement in the GDIEXMPT member:

```
LOCAL RNAME=SYS1.LINKLIB PERMANENT=YES
```

You can specify PERMANENT=NO if you want to be able to negate that statement.

How GDIF Reorders the Exempt List at Startup Time

At startup time, GDIF handles the LOCAL and GLOBAL statements by concatenating the QNAME first, and then the RNAME for any statement on which the RNAME parameter is specified. GDIF appends blank characters to QNAMEs that are less than 8 characters long and uses 8 pound (#) characters to represent a QNAME if a QNAME is not specified on that statement. GDIF then reorders the LOCAL and GLOBAL statements according to these rules:

- By the number of significant characters in the QNAME/RNAME string or in the job name. For example, GDIF orders a statement that contains the value ########/SYS1.LINKLIB (with 12 significant characters) before a statement that contains the value SYSDSN##/SYS* (with 9 significant characters).

  Note: Significant characters are non-wildcard characters, including blanks.

- By the number of leading significant characters (for statements that have the same number of significant characters). For example, GDIF orders a statement that contains the value SYSDSN##/SYS* (with six leading significant characters) before a statement that contains the value ########/SYS1.LINK* (with no leading significant characters).

  Note: Use the DISPLAY EXEMPT command to view the current reordered list.

How GDIF Matches Requests with Exempt List Statements

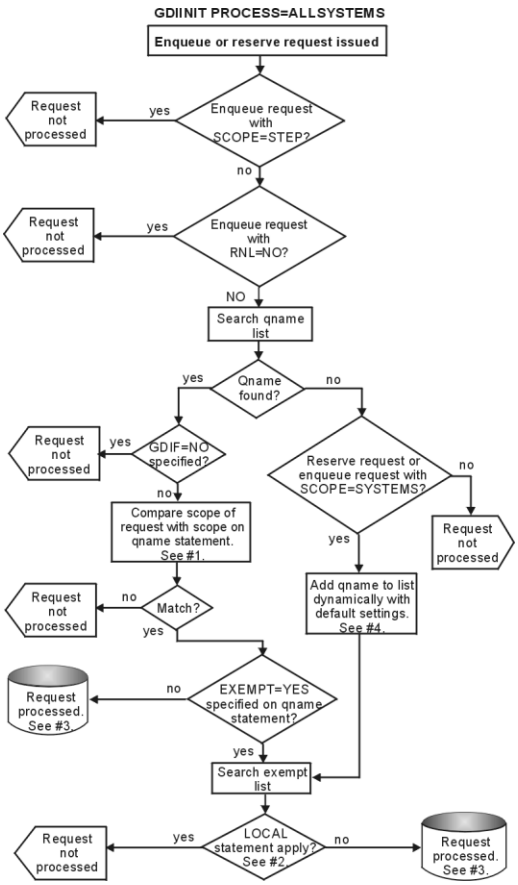
When a task issues a request, GDIF compares the QNAME/RNAME string on the request with the string on the first statement in the reordered list. If these values match, then GDIF selects the statement and ignores other statements that contain QNAME/RNAME strings. If they do not match, GDIF looks at the next statement and performs the same comparison. If GDIF cannot find a matching LOCAL or GLOBAL statement, it selects the DEFAULT statement.

GDIF also performs a job name comparison to select a statement that tells GDIF how to handle requests issued by that job. Based on the statements it selects, GDIF then determines whether to propagate the request. If you specify GLOBAL through a DEFAULT statement, a GLOBAL statement, or both for both the resource and the issuing job, then GDIF propagates the request. Otherwise, GDIF does not propagate the request.

MIMQNAME/GDIEXMPT Processing Flowcharts

The flow charts in this section illustrate how CA MII determines whether to propagate enqueue or reserve requests. Some steps require further explanation, which is provided in notes # 1, # 2, #3, #4, and #5 after the flow charts.

Equation 1: GDIINIT PROCESS=SELECT flowchart



Notes on the Flowcharts

The following notes further explain items in the flow charts:

#1

Use the following table to determine if the SCOPE value you specify on the QNAME statement (row) matches the scope of the ENQ or RESERVE request (column):

SCOPE Values	SYSTEM	SYSTEMS	RESERVES	ALL
Enqueue request issued with SCOPE=SYSTEM	YES	NO	NO	YES
Enqueue request issued with SCOPE=SYSTEMS	NO	YES	NO	YES
Reserve request issued with SCOPE=SYSTEMS	NO	YES	YES	YES

#2

If a reserve request was issued and a LOCAL statement (including the DEFAULT statement) applies to this request (if it is not overridden with a GLOBAL statement), the action taken depends on GDIF EXEMPTRESERVES.

With EXEMPTRESERVES=NO, the LOCAL statement does not apply. Thus the enqueue is propagated and the reserve is either kept or converted, depending on the keyword RESERVES. See #3.

With EXEMPTRESERVES=YES, the LOCAL statement does apply, and the hardware reserve is not converted. If the request is not a reserve and a LOCAL statement applies (if it is not overridden with a GLOBAL statement), then the enqueue is not propagated.

#3

The enqueue is propagated to external systems. If this was a reserve request, then the hardware reserve is either kept or converted, depending on the keyword RESERVES.

RESERVES can be specified on the GDIINIT, QNAME, and GLOBAL statements. If specified on the QNAME statement, then this overrides anything specified on the GDIINIT statement. If specified on the GLOBAL statement, then this overrides anything specified on either the GDIINIT or QNAME statement.

#4

The default settings for a dynamically-added QNAME are:

```
ECMF=NO,EXEMPT=YES,GDIF=YES,REPORTAFTER=0,REPORTCYCLE=60,
SCOPE=SYSTEMS,TRACE=NONE
```

#5

To determine how the exempt list is searched, see How GDIF Reorders the Exempt List at Startup Time (see page 47) and How GDIF Matches Requests with Exempt List Statements in this chapter (see page 47).

Enqueue Conflict Management Facility

You can use the following parameters on a QNAME statement to control the way the ECMF handles conflicts for resources:

ECMF

Tells ECMF whether to issue conflict messages for resources requested on this QNAME.

REPORTAFTER

Tells ECMF when to issue the first message about a conflict for one of these resources.

REPORTCYCLE

Tells ECMF how long to wait before issuing subsequent conflict messages for one of these resources.

For example, the following QNAME statement tells ECMF to issue conflict messages for resources in conflict under SYSIEWLP. The first message is issued after the conflict has persisted for 60 seconds. Then, messages are issued every 60 seconds until the conflict is resolved.

```
SYSIEWLP ECMF=YES REPORTAFTER=60 REPORTCYCLE=60
```

Note that you cannot use the exempt list to tell ECMF to perform special processing for resources. If you do not want ECMF to process some of the resources with the same QNAME, then you need to code one or more of the optional ECMF exit routines.

Note: For information about coding these routines, see the chapter "User Exits (see page 87)."

How You Tell ECMF Whether to Issue Conflict Messages

To receive messages when two or more tasks need access to the same resource at the same time, specify ECMF=YES on a QNAME statement. This tells ECMF to issue conflict messages if a task specifies that QNAME on an ENQ or ISGENQ request and the requested resource already is being used. ECMF conflict messages show the following information about a conflict:

- The QNAME and RNAME associated with the resource
- The name of the task holding the resource
- The name of the system on which the holding task is executing

How ECMF Handles Requeued Jobs When CA MIM Terminates

ECMF maintains checkpoint information for requeued jobs in the CA MIM checkpoint files. By default, ECMF releases requeued jobs if CA MIM terminates and accumulates new checkpoint information when you restart CA MIM. You can change the way ECMF handles requeued jobs and checkpoint information by specifying one of these values on the SETOPTION REQCKPT command:

ASK

Tells ECMF to issue the MIM1049 WTOR, which asks the operator how to handle requeued jobs and checkpoint information.

DISCARD

Tells ECMF to leave the jobs in a held state and discard checkpoint information. Release these jobs manually.

RELEASE

Tells ECMF to release requeued jobs and accumulate new checkpoint information when you restart ECMF.

This is the default.

USE

Tells ECMF to leave the jobs in a held state until you restart ECMF. When ECMF is restarted, it rebuilds the checkpoint information and releases the jobs. If the job needs a data set that is still unavailable, then ECMF reinstates the job to the control of ECMF. If the data set is available, then ECMF releases the job immediately.

How You Determine When Conflict Messages Are Issued

The following table shows who receives ECMF conflict messages.

Type of Conflict:	Who ECMF Informs:
A batch job or system task is holding a resource that a TSO user needs	The TSO user only
A TSO user is holding a resource that a batch job or system task needs	The TSO user and the operator with the job waiting
A batch job or system task is holding a resource that another batch job or system task needs	The operator with the job waiting
A TSO user is holding a resource that another TSO user needs	Both TSO users

To tell ECMF how many seconds to wait before issuing the first conflict message, you need to specify a value for the REPORTAFTER parameter on the QNAME statement. By default, ECMF issues the first message immediately. ECMF automatically reissues conflict messages every 60 seconds until the conflict is resolved. To change this interval, specify the number of seconds ECMF should wait through the REPORTCYCLE parameter on the QNAME statement. For example, if you want ECMF to issue the first conflict message for resources in the SYSIEWLP QNAME class after 5 seconds, and have these messages repeat every 30 seconds, you would specify the following QNAME statement in the MIMQNAME member:

```
SYSIEWLP ECMF=YES REPORTAFTER=5 REPORTCYCLE=30
```

Note: For the QNAME SYSDSN, the REPORTCYCLE parameter also affects the way ECMF requeues batch jobs that cannot obtain resources.

Receive a Single Set of Conflict Messages

ECMF uses the values you set through the REPORTAFTER and REPORTCYCLE parameters only for *unconditional* ENQ requests: that is, ENQ requests that specify RET=NONE or RET=HAVE. If a conflict involves one or more *conditional* ENQ requests in which RET=USE is specified, then ECMF issues a single set of messages to notify the participants about the conflict.

Note: For more information about the RET parameter on an ENQ request, see the IBM documentation.

Suppression of Redundant Conflict Messages

ECMF may suppress redundant ENQ conflict messages. If the same victim ASID and TCB contend more than once message within 60 seconds for the same QNAME/RNAME, and the ENQ requests are conditional, then ECMF will suppress any message beyond the first one, until the 60 seconds expires. The 60 second window is internally hard-coded, and cannot be customized through CA MIM parameters.

Job Requeue

The Problem: When a batch job is submitted, the initiator task issues a SYSDSN RET=ECB type ENQ request for every data set on all DD cards in the JCL. All ENQ requests must be obtained before the batch job begins execution. If any data set is not available, then the initiator goes into a WAIT state until all data sets become available. This prevents another job from using this initiator until the first job completes. This is true even if the data sets of other job are currently available.

ENQs Eligible for Job Requeue

Specifying ECMF REQUEUE requeues only batch jobs that contend for SYSDSN RET=ECB type ENQ requests issued from an initiator task at job initiation. SYSDSN ENQ requests that result from dynamic allocation or at the step level are not eligible to be REQUEUED.

GDGs

When a batch job is submitted, and relative GDGs are referenced in the JCL, the initiator issues SYSDSN ENQ requests for the base GDG name. For example, if STEP3 of a batch job contains a DD for DSN=SYSPROG.DUMP(+2), then an ENQ request is issued for QNAME=SYSDSN RNAME=SYSPROG.DUMP at job initiation. Later, when STEP3 runs, another ENQ request is issued by STEP allocation on the resolved GDG name, or G0000V00 (SYSPROG.DUMP.G0005V00).

If this ENQ request is not available, then the MIM1038 and MIM1039 ECMF messages are issued, but the job will not be requeued. If ECMF requeued the job, and then later released it, previous steps would be rerun, and this could cause data to be corrupted.

SYSZVOLS ENQ

ECMF does not requeue a batch job that contends for the SYSZVOLS QNAME. This ENQ is issued during step initiation after the job has already started. If ECMF requeued the job, and then later released it, previous steps would be rerun, and this could cause data to be corrupted.

Aliases

Aliases are handled the same as GDGs. These are not resolved until STEP initiation, and so are not eligible for REQUEUE.

First Job Step

If an ENQ request is issued for the first step of a job, but after job initiation has completed, then ECMF will not requeue the job.

Requirements

- The ECMF REQUEUE feature requires the use of a checkpoint file on each system. This is not a shared file, and is unique to each system.

  Note: For more information on allocating a checkpoint file, see the *CA MIM Programming Guide*.

- SYSDSN must be in the QNAME member with ECMF=YES. The REPORTCYCLE parameter on this QNAME statement is used in conjunction with the SET ECMF REQAFTER parameter to determine when the job should be requeued.
- SET ECMF REQUEUE=ON must be coded in the MIMCMNDS member.

How It works

First, ECMF detects whether all requested resources are currently available to a batch job. If not, then ECMF issues MIM1038/MIM1039/MIM1040 messages until the REQAFTER value expires. At this time, CA MIM executes the JES2 commands to requeue the batch job. Specify the REQAFTER option on the SET ECMF command in the commands member. The value for this option is the number of REPORTCYCLEs on the SYSDSN QNAME that ECMF must wait before it requeues the batch job.

Example:

In the MIMQNAME member, issue the following statement:

```
SYSDSN GDIF=YES SCOPE=SYSTEM REPORTAFTER=0 RPTCYCLE=60 ECMF=YES EXEMPT=YES
```

In the MIMCMNDS member, issue the following statement:

```
SET ECMF REQAFTER=3
```

ECMF waits 3(REQAFTER) X 60(REPORTCYCLE) = 180 seconds before it requeues the batch job. You can change these values dynamically using the ALTER and SET ECMF commands.

After this timer expires, the following JES2 commands are executed:

```
$HJxxxxx
$EJxxxxx
$CJxxxxx
```

The JES2 command character in this example is $. ECMF dynamically determines the appropriate JES2 subsystem command character. Also, these JES2 commands are executed using the UTOKEN of the address space of the batch job. If you want these commands to be executed using the UTOKEN of the CA MIM address space, then use the SET ECMF REQSECUR=ON command.

After these commands are issued, the job is removed from the initiator, and stays in the HELD queue until the resource becomes available. The following message displays:

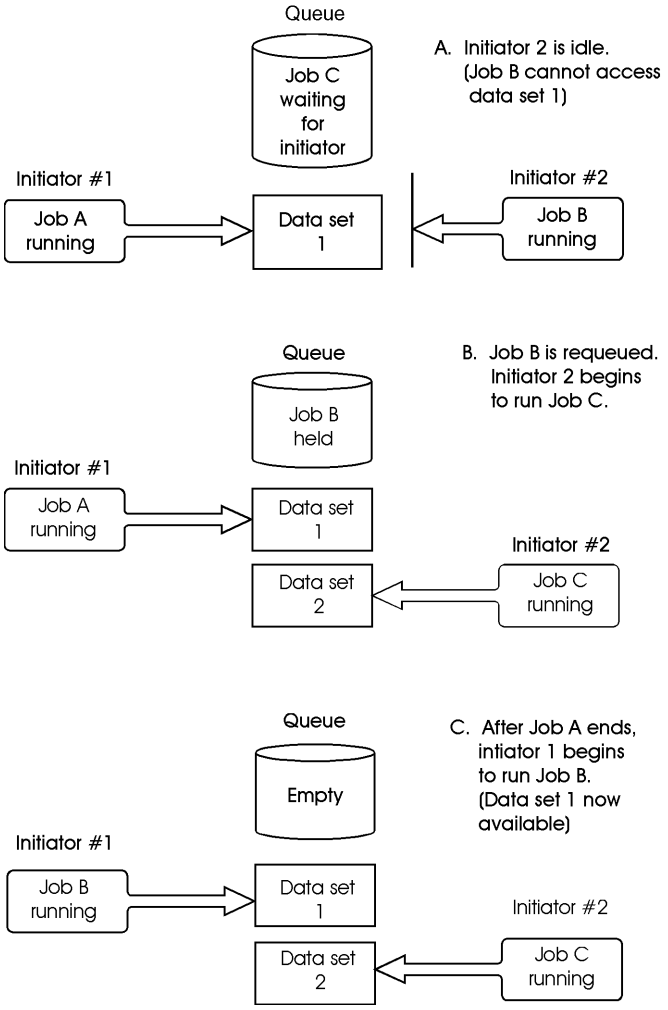
```
MIM1083 jobname(jesid) REQUEUED - AWAITING DATASETS
```

ECMF determines resource availability, both locally and globally, of all required resources for all DD cards in the JCL of the batch job. This check is performed every REQCYCLE, as coded on the SET ECMF REQCYCLE command. The default is 30 seconds. After ECMF determines that ALL resources are now available, it executes the $Ajxxxxx JES2 command to release the batch job.

The following message is issued:

```
MIM1048 jobname(jesid) RELEASED - DATASETS NOW AVAILABLE
```

The following graphic illustrates this process:



How ECMF Handles Requeued Jobs When CA MIM Terminates

ECMF maintains checkpoint information for requeued jobs in the CA MIM checkpoint files. By default, ECMF releases requeued jobs if CA MIM terminates and accumulates new checkpoint information when you restart CA MIM. You can change the way ECMF handles requeued jobs and checkpoint information by specifying one of these values on the SETOPTION REQCKPT command:

ASK

Tells ECMF to issue the MIM1049 WTOR, which asks the operator how to handle requeued jobs and checkpoint information.

DISCARD

Tells ECMF to leave the jobs in a held state and discard checkpoint information. Release these jobs manually.

RELEASE

Tells ECMF to release requeued jobs and accumulate new checkpoint information when you restart ECMF.

This is the default.

USE

Tells ECMF to leave the jobs in a held state until you restart ECMF. When ECMF is restarted, it rebuilds the checkpoint information and releases the jobs. If the job needs a data set that is still unavailable, then ECMF reinstates the job to the control of ECMF. If the data set is available, then ECMF releases the job immediately.

Compatibility with Job Scheduling Packages

CA MII is compatible with all scheduling packages. CA 7 customers should specify REQUE=YES in CA 7. The default for this parameter is NO.

However, ECMF REQUEUE was not designed to aid in job scheduling and does not replace the need for a job scheduling package. After a job is requeued, resource availability is only determined at fixed intervals (SET ECMF REQCYCLE). So, it is possible that other jobs can “sneak in” and obtain resources that were needed by the requeued job, during the interim.

For example, with REQCYCLE=30:

mm:ss

00:00 JOBA requeued, 2 of 7 data sets were not available

00:30 REQCYCLE timer pops

- ECMF checks availability of all 7 data sets
- Now only 1 data set is not available (DATASET1) JOBB owns this
- JOBA not released

01:00 REQCYCLE timer pops

- ECMF checks availability of all 7 data sets
- Data set 1 still owned by JOBB
- JOBA not released

01:05 JOBB ends and releases DATASET1

01:10 JOBC starts and obtains DATASET1

01:30 REQCYCLE timer pops

- ECMF checks availability of all 7 data sets
- JOBC now owns DATASET1
- JOBA still not released

01:40 JOBC ends and releases DATASET1

02:00 REQCYCLE time pops

- ECMF checks availability of all 7 data sets
- All 7 are now available
- JOBA is released

Though JOBC was started *after* JOBA, it finished before JOBA. Thus, ECMF REQUEUE cannot be used as a scheduling package, and the order in which the jobs run may or may not be first in, first out.

Restrictions

JES3

JES3 customers should not use the REQUEUE option because JES3 determines resource availability prior to job initiation, which makes this feature redundant. JES3 customers should specify SET ECMF REQUEUE=OFF.

Poly-JES Environments

ECMF REQUEUE works on a system where more than one JES2 address space is running.

Exits

For information on using ECMF user exits to customize ECMF processing, see the chapter "User Exits (see page 87)."

AUTOFREE Feature

The ECMF AUTOFREE feature can be used to dynamically deallocate a data set that was dynamically allocated to a TSO user, but is currently marked Not-in-Use. Data sets that are marked Not-in-Use can cause local or global ENQ conflicts for the SYSDSN QNAME. For example, if a TSO user issues a compile command, object and load files are allocated dynamically. These files are marked Not-in-Use after the compile completes, but the files are still allocated to the TSO user. Non-temporary data sets that are allocated through SVC 99 in a TSO task and are still allocated at task termination (normal/abnormal) are marked Not-in-Use through SVC 99 Verb Code 5.

To determine which data sets are currently marked Not-in-Use, issue the following TSO command:

```
LISTA ST SY H
```

The following information displays:

```
--DSORG--CREATED-----EXPIRES-----SECURITY--DDNAME---DISP--
SYS1.LINKLIB
  PO     06/25/1998  00/00/0000  NONE      RES      KEEP
SYS00066.T143549.RA000.USER001.R0128686
  PS     03/06/2000  00/00/0000  NONE      ISPCTL0  DELETE,DELETE
SYS1.LINKLIB
  PO     06/25/1998  00/00/0000  NONE      SYSLMOD  KEEP
*USER001.SDSITEST.OBJ
  PS     03/06/2000  00/00/0000  NONE      SYS00045 KEEP,KEEP
*USER001.SDSITEST.LINKLIST
  PS     03/06/2000  00/00/0000  NONE      SYS00047 CATLG,CATLG
*USER001.SDSITEST.LOAD(TEMPNAME)
  PO     03/06/2000  00/00/0000  NONE      SYS00049 CATLG,CATLG
```

The data sets marked with an * character are currently marked Not-in-Use, and are eligible for AUTOFREE.

Note: GDG and temporary data sets are not eligible to be automatically freed by this feature of ECMF.

If a batch job is submitted that needs one of these files, the job cannot allocate these files until the TSO user deallocates them or ends the TSO session.

When the AUTOFREE feature is active, ECMF checks to see whether a data set is marked Not-in-Use when the conflict occurs. ECMF checks the DSABNUSE bit in the DSAB, if this bit is not on, then the data set is considered Not-in-Use. If the bit is not on, then ECMF automatically frees the resource through SVC 99, the TSO user is notified with the MIM1043 messages, and the operator is notified with MIM1038, MIM1039, and MIM1040 messages. By doing this, ECMF enables the task that needs the data set to continue.

When the conflict is first detected, the following messages are sent to the operator:

```
MIM1038 USER001D CONTENTION WITH USER001 OWNS SHR  ON XE44
MIM1039 USER001D NEEDS EXCL SYSDSN   USER001.SDSITEST.OBJ
MIM1039 USER001D NEEDS EXCL SYSDSN   USER001.SDSITEST.LINKLIST
MIM1039 USER001D NEEDS EXCL SYSDSN   USER001.SDSITEST.LOAD
MIM1040 USER001D WAITING FOR RESOURCES
```

When the resource is freed, the following messages are sent to the TSO user:

```
MIM1043 SYSDSN   USER001.SDSITEST.LOAD FREED FOR USER001D ON XE44
MIM1043 SYSDSN   USER001.SDSITEST.LINKLIST FREED FOR USER001D ON XE44
MIM1043 SYSDSN   USER001.SDSITEST.OBJ FREED FOR USER001D ON XE44
```

Enhanced Data Set Integrity Facility

This section discusses the Enhanced Data Set Integrity Facility (EDIF).

How You Apply EDIF Processing to Data Sets

EDIF lets you apply processing options to all data sets, to a specific group of data sets, or to an individual data set. You can tell EDIF to perform the following types of processing to prevent accidental data set damage:

- Prevent simultaneous updates that occur because serialization is not requested
- Prevent unauthorized programs from updating data sets
- Prevent unauthorized programs from changing the attributes of data sets
- Prevent PDS directories from being overlaid
- Prevent unauthorized programs from reading data sets

To tell EDIF what type of processing you want, you need to create an *EDIF processing list*. An EDIF processing list is a series of EDIF processing statements that indicate what type of processing EDIF should perform based on characteristics of the data set name or the data set organization.

You can create an EDIF processing list by specifying EDIF processing statements in the EDIPARMS member of the CA MIM parameter data set. If you want to use a different member, then you need to identify the member that you are using through the MEMBER parameter on the EDIINIT statement. For example, you can use a member named DSPARMS by specifying EDIINIT MEMBER=DSPARMS.

While EDIF is running, you can change the member you are using or tell EDIF to obtain more up-to-date information from this member.

A sample EDIF processing list is provided in member EDIPARMS of the CAI.CBTDPARM data set.

More Information:

How You Dynamically Modify the EDIF Processing List (see page 76)
CA Product EDIF Statements (see page 133)

How You Use EDIF Processing Statements

Each EDIF processing statement contains a list of processing options that apply to a specific data set or to a group of data sets. Use the following statements to tell EDIF how to process data sets:

DATASET

Applies processing options to a specific data set.

DEFAULT

Applies processing options to all data sets.

DSORG

Applies processing options to all data sets that have a specified data set organization.

PATTERN

Applies processing options to all data sets that have a specified pattern in their data set names.

PREFIX

Applies processing options to all data sets that have a specified prefix in their data set names.

SUFFIX

Applies processing options to all data sets that have a specified suffix in their data set names.

UTILITY

This special EDIF processing statement lets you identify a group of programs to be treated similarly when EDIF processes a data set. Specify the name that you assigned to a UTILITY statement on other EDIF processing statements. This lets you quickly identify programs that should be treated the same way, instead of listing these programs individually. For a description of the UTILITY statement, see Listing Programs Receiving the Same Processing in this chapter.

These statements are not mandatory, including the DEFAULT statement. Note that if you do not specify a DEFAULT statement, then all options are prefixed with NO; for example, NOABEND or NOATTRIBUTE.

Rules for Coding EDIF Processing Statements

Each EDIF processing statement begins with a statement name that indicates what data set characteristics identify the data sets to which the statement applies. For example, a PREFIX statement indicates that a data set prefix identifies the appropriate data sets. Each statement also contains parameters that you can specify in any order. You can use non-ambiguous truncations for these parameters, but not for the statement name. None of the examples in this chapter use truncations.

The order in which you place similar statements in your processing list can determine which statement is chosen.

Notes:

- For more information on ordering statements, see How EDIF Merges and Overrides Statements (see page 63).
- All standard rules on syntax, commenting, and continuation apply. See the *Statement and Command Reference Guide.*

How EDIF Merges and Overrides Statements

If several statements affect a data set, EDIF merges these statements to determine which processing options to use. This allows you to accomplish the following tasks:

- Perform *additional* processing for selected data sets. For example, you can specify one statement that applies to all data sets and then another that performs special additional processing for the SYS1.LINKLIB data set.
- *Override* processing that is in effect for a data set. For example, you can prevent unauthorized updates for all data sets and then override this processing for data sets that start with the prefix SYS1.

When overriding processing options for a data set, the order in which EDIF merges its processing statements is important. EDIF builds a list of applicable options for a data set by prioritizing processing statements as follows:

1. DEFAULT
2. DSORG
3. PREFIX
4. SUFFIX
5. PATTERN
6. DATASET

You can use statements at the end of this list to override processing from a previous statement. For example, you can set options for all data sets on a DEFAULT statement and override that processing for certain data sets through PATTERN and DATASET statements.

EDIF uses, at the most, one of each type of statement when it processes a data set. For example, if the data set name on three PREFIX statements matches, then EDIF uses only one of these PREFIX statements for that data set. If you specify more than one of the same type of statement, keep the following guidelines in mind to ensure that EDIF uses the correct statement:

- For PREFIX and SUFFIX statements, EDIF uses the statement that has the longest matching data set name. Therefore, make sure that the data set name or partial data set name you specify is as exact as possible.
- For PATTERN statements, EDIF uses the first statement that matches the name of the data set. Therefore, place the most specific PATTERN statements first and the least specific last.

How You Turn Off Processing Options for Data Sets

Many EDIF processing options are specified as values on the OPTION parameter. You can use the EDIF merge process to turn off any processing option specified using the OPTION parameter. To turn off a certain processing option, specify the prefix NO in front of that option on the appropriate statement. To negate all processing options, specify OPTION(NONE) instead.

For example, to detect and prevent attribute changes for partitioned data sets, you would specify OPTION(ABEND,ATTRIBUTES) on a DSORG statement. To detect changes for the MY.DATA data set without preventing these changes, you then would specify OPTION(NOABEND) on a DATASET statement for this data set.

How You Override Lists of Programs

When you tell EDIF to detect or prevent instances of data set damage, you also can provide lists of programs that are exempted from that type of checking or are allowed to perform that type of update. For example, when you are detecting or preventing unauthorized update operations, you can use the AUTHORIZED parameter to identify programs that are authorized to update a data set.

Because you may want to use different lists of programs for different data sets, EDIF lets you override these program lists during the EDIF merge process. To use a different list of programs, specify the names of those programs on the appropriate EDIF processing statement.

For example, suppose you specify AUTHORIZED(PROGRAM(TEST1,TEST2)) on a PREFIX statement. To allow only the program named TEST1 to update the TEST.PAYROLL data set, specify AUTHORIZED(PROGRAM(TEST1)) on a DATASET statement for the TEST.PAYROLL data set.

How You Identify the Data Sets EDIF Should Process

You can use the NAME parameter on DATASET, PREFIX, SUFFIX, PATTERN, and DSORG statements to identify the data sets to which a statement applies. You can specify the NAME parameter on a UTILITY statement, but for a different purpose.

Use the NAME parameter to identify data sets in these ways:

- On the DATASET statement, specify the name of the data set that EDIF should use. For example, DATASET NAME=SYS1.LINKLIB.
- On the PREFIX, SUFFIX, and PATTERN statements, specify the part of the data set name that EDIF should use to identify applicable data sets; for example, PREFIX NAME=SYS1.
- On DSORG statements, specify the data set organization of the data sets that the statement applies to:
  - DIRECT (or DA)
  - ISAM
  - PARTITIONED (or PO)
  - SEQUENTIAL (or PS)
  - UNMOVABLE-PARTITIONED (or POU)
  - UNMOVABLE-SEQUENTIAL (or PSU)
  - VSAM

  For example, to apply a DSORG statement to all partitioned data sets, specify DSORG NAME=PO.

EDIF lets you use these wildcard characters on PATTERN statements:

*****

Matches the remainder of the data set name.

?

Matches the remainder of the current index level.

#

Matches a single character in the data set name.

You can use combinations of wildcard characters to match a pattern. For example, the pattern #.?.* matches data set names A.DATA.SET or A.B.C, but not A.DATA or MY.DATA.SET.

How You Detect and Prevent Attribute Modification

You can use the *attribute verification* feature of EDIF to see whether inappropriate programs are changing the attributes of a data set and to prevent attribute changes from occurring. You can enable this feature by specifying OPTION(ATTRIBUTES) on a DEFAULT, DSORG, PREFIX, SUFFIX, PATTERN, or DATASET statement. However, EDIF does not prevent the attribute change unless the ABEND option also is in effect. We recommend that you monitor attribute changes before specifying the ABEND option on any of these statements.

Note: EDIF does not detect attribute violations when DISP=MOD is coded in the JCL.

To detect attribute violations for testing purposes, we recommend that you specify OPTION(ATTRIBUTES,SMF,SUPPRESSMESSAGES). EDIF then records attribute violations in an SMF record and suppresses the MIM4004 message. This lets you identify data sets that are being damaged by attribute changes, without sending notification messages to TSO users and operators. Based on this information, you can decide which programs to abend and which programs to exempt from attribute verification.

For example, to detect attribute violations for the SYS1.LINKLIB data set and record violations in an SMF record, specify the following statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION(ATTRIBUTES,SMF,SUPPRESSMESSAGES)
```

To prevent attribute changes, specify OPTION(ATTRIBUTES,ABEND). If EDIF detects an attribute violation, then EDIF issues message MIM4004 to notify you about the violation and abends the program to prevent the update. EDIF also issues message MIM4007 to tell you that it abended the program.

For example, to prevent attribute changes for the SYS1.LINKLIB data set and record update violations in an SMF record, specify the following statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION(ATTRIBUTES,ABEND,SMF)
```

Be careful when specifying the ABEND option on the DEFAULT statement or on any other statement that affects a large number of data sets. We recommend that you do not specify the SUPPRESSMESSAGES option when the ABEND option is in effect.

By default, EDIF uses abend code U913 when abending a program. To use a different abend code, specify a new code through the ABENDCODE parameter on the EDIINIT statement.

EDIF Attribute Values

Use the following parameters to indicate what attribute values EDIF should use during attribute verification:

BLKSIZE

Indicates what block size EDIF uses

DSORG

Indicates what data set organization EDIF uses

LRECL

Indicates what logical record length EDIF uses

RECFM

Indicates what record format EDIF uses

If you want EDIF to use a value other than the default in the data set control block (DSCB), specify a new value on the appropriate parameter.

If you do not want EDIF to check a certain attribute, then specify ANY on the appropriate parameter. For example, EDIF accepts any logical record length if you specify LRECL=ANY.

If you want EDIF to use a block size value of 6144, a data set organization value of PARTITIONED, and whatever logical record length and record format are in the DSCB when it performs attribute verification for the SYS1.LINKLIB data set, then you would specify this statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION(ATTRIBUTES,ABEND), BLKSIZE=6144
    DSORG=PARTITIONED
```

By default, EDIF checks output data for the presence of carriage control characters and verifies that the data set organization matches the data. For example, if carriage control characters are found in the data, then DSORG=FB would not be valid, but DSORG=FBA would. To suppress this check, include the OPTION(IGNORECC) entry on the processing statement.

How You Identify Exempted Programs

You can identify exempted programs, that is, programs for which attribute verification should be suspended, using the EXEMPT parameter. For example, specify EXEMPT(PROGRAMS(SYS1PAY,SYS2PAY)) on the appropriate statement to suspend attribute checking for the SYS1PAY and SYS2PAY programs.

If you have listed authorized programs on a UTILITY statement, then you can specify the name of that UTILITY statement instead of listing the programs individually.

Example: Preventing Attribute Changes

Suppose that you want EDIF to perform the following types of processing for partitioned data sets:

- Detect attempts to change the block size or record format attributes, using those in the DSCB of the data set for comparison
- Ignore the logical record length attribute
- Ignore carriage control characters in the record format
- Exempt programs listed on the UTILITY statement named ISPF from attribute verification
- Record attempted updates in an SMF record
- Abend programs that try to change these attributes for the SYS1.LINKLIB data set only

To do this, specify the following statements in the EDIPARMS member:

```
DSORG NAME=PARTITIONED OPTION=(ATTRIBUTES,SMF,IGNORECC),
    EXEMPT(UTILITY(ISPF)) LRECL=ANY

DATASET NAME=SYS1.LINKLIB OPTION=ABEND
```

How EDIF Detects and Prevents Simultaneous Updates

EDIF can check whether several programs are updating a data set simultaneously and can prevent this type of simultaneous update from occurring. To tell EDIF to check for simultaneous updates, specify OPTION(ENQUEUE) on a DEFAULT, DSORG, PREFIX, SUFFIX, PATTERN, or DATASET statement. To detect and prevent these updates, EDIF issues an ENQ request that uses the QNAME EDIDSN. In a multiple-system environment, use GDIF or a similar product to propagate these ENQ requests to all systems.

To detect simultaneous updates for testing purposes, we recommend using the following statement:

```
OPTION=(ENQUEUE,SMF,SUPPRESSMESSAGES)
```

This lets you identify data sets that are victims of simultaneous updates without sending notification messages to TSO users and operators. You can then decide whether to make programs wait for access to a data set.

For example, to detect simultaneous updates for the SYS1.LINKLIB data set and record these updates in an SMF record, you would specify the following statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION(ENQUEUE,SMF)
```

To prevent simultaneous updates, specify the following statement:

```
OPTION=(ENQUEUE,WAIT,CONFLICTMESSAGES)
```

This tells EDIF to check for simultaneous updates and issue conflict messages when two or more jobs attempt to update a data set simultaneously.

Because we also specified the WAIT option, EDIF prevents the update from occurring by issuing an ENQ request for QNAME EDIDSN. This ENQ request serializes access to the data set. We recommend that you do not specify the SUPPRESSMESSAGES option to suppress notification messages when the WAIT option is in effect.

For example, to prevent simultaneous updates for the SYS1.LINKLIB data set and record update violations in an SMF record, specify the following statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION=(ENQUEUE,WAIT,SMF,CONFLICTMESSAGES)
```

VSAM Data Sets

We do not recommend EDIF OPTION=ENQ processing for VSAM data sets that have SHAREOPTIONS of 1 or 2. If the SHAREOPTIONS are 3 or 4, then you can use EDIF to prevent simultaneous updates. If EDIF ENQ processing is in effect for a VSAM data set with SHAREOPTIONS of 1 or 2, deadlocks could result between the SYSVSAM and EDIDSN ENQs. We also recommend that you not code DSORG=VSAM OPTION=(ENQ,WAIT) because this would extended EDIF ENQ processing to all VSAM data sets.

Example: Preventing Simultaneous Updates

Suppose that you want EDIF to perform the following tasks for partitioned data sets:

- Prevent simultaneous updates for all partitioned data sets except the TEST1.MYDATA data set
- Record update violations in an SMF record
- Issue conflict messages when an update violation occurs

To do this, specify the following statements in the EDIPARMS member:

```
DSORG NAME=PARTITIONED OPTION=(ENQUEUE,WAIT,SMF,CONFLICTMESSAGES)

DATASET NAME=TEST1.MYDATA OPTION=NOWAIT
```

How You Detect and Prevent DISP=SHR Updates

The EDIF DISP=SHR verification is useful if any of the programs at your site extract information from DSCBs before issuing OPEN requests for data sets. In these cases, programs can damage a data set even though the ENQUEUE option is in effect. Because EDIF issues ENQ requests when a data set is opened, the ENQ request will be issued too late to prevent damage from occurring. You can tell EDIF to monitor a designated group of programs to see if they are updating certain data sets when DISP=SHR is specified in the JCL of the job.

You also can tell EDIF to prevent these types of updates from occurring. To detect or prevent these updates, specify the CHECKEXCLUSIVE parameter on a DEFAULT, DSORG, PREFIX, SUFFIX, PATTERN, or DATASET statement. EDIF does not check all programs that update a data set. Identify the programs that you want EDIF to check in one of the following ways:

- Specify program names on the CHECKEXCLUSIVE parameter
- Specify names on the UTILITY statement that identifies these programs
- To detect this type of update violation for testing purposes, We recommend that you specify OPTION=(SMF,SUPPRESSMESSAGES). EDIF records update violations in an SMF record without broadcasting the MIM4006 message.

For example, to see whether the programs listed on the UTILITY statement named TEST are updating the SYS1.LINKLIB data set without proper serialization, you could specify the following statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION=(SUPPRESSMESSAGES,SMF),
    CHECKEXCLUSIVE(UTILITY(TEST))
```

To prevent updates that occur when DISP=SHR is specified, you need to specify OPTION(ABEND). EDIF abends programs that attempt to update the data set when DISP=SHR is specified and issues message MIM4006 to notify you about the update violation.

EDIF also issues message MIM4008 to notify you that the program has been abended. You should use extreme care when specifying the ABEND option on the DEFAULT statement or on any other statement that affects a large number of data sets.

You also can record update violations in an SMF record by specifying the SMF option. We recommend that you do not specify the SUPPRESSMESSAGES option to suppress notification messages when the ABEND option is in effect.

For example, to prevent the programs listed on the UTILITY statement named TEST from updating the SYS1.LINKLIB data set when DISP=SHR is specified, you would place this statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION=(SMF,ABEND),
    CHECKEXCLUSIVE(UTILITY(TEST))
```

By default, EDIF uses abend code U913 when abending a program. To use a different abend code, specify a new code through the ABENDCODE parameter on the EDIINIT statement.

Example: Preventing DISP=SHR Updates

Suppose that you want EDIF to perform the following type of processing for partitioned data sets:

- Determine whether programs TESTPAY1 or TESTPAY2 are updating partitioned data sets when DISP=SHR is specified.
- Record update violations in an SMF record.
- Prevent these programs from updating the PAYROLL data set, but not other partitioned data sets, when DISP=SHR is specified.

To do this, specify these statements in the EDIPARMS member:

```
DSORG NAME=PO CHECKEXCLUSIVE(PROGRAMS(TESTPAY1,TESTPAY2)),
    OPTION=SMF

DATASET NAME=PAYROLL OPTION=ABEND
```

The abbreviation PO in the above example stands for PARTITIONED OPTION.

How You Detect and Prevent Unauthorized Updates

You can use the *utility verification* feature of EDIF to see whether unauthorized programs are updating data sets and to prevent unauthorized updates from occurring. You can enable this feature by specifying OPTION=UTILITY on a DSORG, PREFIX, SUFFIX, PATTERN, or DATASET statement. During utility verification, EDIF compares the name of the program that issues an OPEN request to a list of programs authorized to update that data set. If that program is not on the list, then EDIF considers the OPEN request to be an update violation.

You can identify authorized programs using the AUTHORIZED parameter. For example, you can authorize programs IEWL and ISRUDA to update the SYS1.LINKLIB data set by specifying AUTHORIZED=(PROGRAMS=(IEWL,ISRUDA)) on that processing statement.

If you have listed authorized programs on a UTILITY statement, then you can specify the name of that UTILITY statement instead of listing the programs individually. For example, suppose that you assigned the name LOADLIB to a UTILITY statement for these programs. To authorize these programs, specify AUTHORIZED=(UTILITY(LOADLIB)) on that processing statement.

To detect update violations for testing purposes, We recommend that you specify OPTION=(UTILITY,SMF,SUPPRESSMESSAGES). This lets you identify data sets that are being damaged by unauthorized updates, without sending notification messages to TSO users and operators. You can then decide which programs to abend and which programs to authorize.

For example, to detect update violations for the SYS1.LINKLIB data set and record violations in an SMF record, specify the following statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION=(UTILITY,SMF,SUPPRESSMESSAGES)
```

To prevent unauthorized updates, specify OPTION=(UTILITY,ABEND). EDIF then abends programs that attempt to perform unauthorized updates. If EDIF detects an unauthorized update, then EDIF issues message MIM4002 to notify you about the update violation and message MIM4003 to notify you that the program has been abended.

Note: You should use extreme care when specifying the ABEND option on the DEFAULT statement or on any other statement that affects a large number of data sets. We recommend that you do not specify the SUPPRESSMESSAGES option to suppress notification messages when the ABEND option is in effect.

For example, to prevent unauthorized updates for the SYS1.LINKLIB data set and record update violations in an SMF record, specify the following statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION=(UTILITY,ABEND,SMF)
```

By default, EDIF uses abend code U913 when abending a program. To use a different abend code, specify a new code through the ABENDCODE parameter on the EDIINIT statement.

How You Use EDIF to Determine Authorized Programs

If you do not specify the AUTHORIZED parameter, then EDIF decides which programs should update a data set by examining the UTILITY statements you have provided. EDIF compares the record format and data set organization values on each UTILITY statement with the values in the DSCB of the data set. If these values match, then EDIF lets any program named on that UTILITY statement update that data set.

For example, suppose that you specified the following statements in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION(UTILITY,ABEND)

UTILITY NAME=LOADLIB PROGRAMS=(IEWLF128,ISRUDA,IEBCOPY),
    RECFM=UNDEFINED DSORG=PARTITIONED
```

Because the SYS1.LINKLIB data set has a record format value of UNDEFINED and a data set organization value of PARTITIONED, EDIF lets any of the programs named on this UTILITY statement update the SYS1.LINKLIB data set.

How You Allow EDIF to Identify Authorized Programs During Utility Verification

EDIF can identify authorized programs during utility verification if you omit the AUTHORIZED parameter on all statements that affect a data set. This lets you create lists of authorized programs and make EDIF determine which list to use, based on data set characteristics.

EDIF compares the data set organization and record format values in the DSCB of the data set with the values on the UTILITY statement to see if they match. If they match, then all programs listed on that UTILITY statement are authorized to update the data set. The following UTILITY statement parameters provide EDIF with the data set characteristics that it uses to identify authorized programs:

DSORG

Tells EDIF what data set organization to use

RECFM

Tells EDIF what record format to use

Note: When you omit the RECFM parameter, but specify the DSORG parameter, EDIF will still match the UTILITY statement, based on the DSORG parameter only.

For example, suppose that you specified the following statements:

```
DSORG NAME=PO OPTION(UTILITY,ABEND)
UTILITY NAME=SOURCE RECFM=FB DSORG=PO PROGRAMS=(EDIT,IEBUPDTE)
UTILITY NAME=CLIST RECFM=VB DSORG=PO PROGRAMS=EDIT
UTILITY NAME=LOAD RECFM=UNDEFINED DSORG=PO PROGRAMS=(IEWL,SPZAPLIB)
```

In this example, EDIF uses the UTILITY statement named SOURCE for the SYS1.MACLIB data set (because its record format and data set organization match this statement), the statement named LOAD for the SYS1.LINKLIB data set, and the statement named CLIST for the TSO.CLIST data set. The abbreviation PO stands for PARTITIONED OPTION.

Example: Preventing Unauthorized Updates

Suppose that you want EDIF to perform the following tasks for data sets that have the suffix .PAYROLL:

- Detect unauthorized updates for these data sets and record update violations in an SMF record
- Allow programs S1PAY and S2PAY to update the SYS1.PAYROLL data set
- Abend all other programs that try to update a data set that has this suffix

To do this, specify the following statements in the EDIPARMS member:

```
SUFFIX NAME=.PAYROLL OPTION=(UTILITY,SMF,ABEND)
DATASET NAME=SYS1.PAYROLL AUTHORIZED=(PROGRAMS=(S1PAY,S2PAY))
```

How You Prevent Unauthorized Programs from Reading Protected Data Sets

You can use the *read verification* feature of EDIF to see whether unauthorized programs are reading data sets and to prevent unauthorized read operations from occurring by specifying OPTION=ACCESSCHECK on a DSORG, PREFIX, SUFFIX, PATTERN, or DATASET statement. During read verification, EDIF compares the name of the program that tries to read the data set to a list of programs authorized to read that data set. If that program is not on the list, then EDIF considers the request to be a read violation.

To identify programs authorized to read a data set, use the ACCESSLIST parameter. For example, authorize programs ISREDIT and ISRUDA to read SYS1.LINKLIB by specifying ACCESSLIST(PROGRAMS(ISREDIT,ISRUDA)) on that processing statement.

If you have listed authorized programs on a UTILITY statement, then you can specify the name of that UTILITY statement instead of listing the programs individually. For example, suppose that you assigned the name ISPF to a UTILITY statement for these programs. You could authorize these programs by specifying ACCESSLIST(UTILITY(ISPF)) on that processing statement.

To detect read violations for testing purposes, We recommend that you specify OPTION=(ACCESSCHECK,SMF,SUPPRESSMESSAGES). This lets you identify data sets that are victims of unauthorized read operations, without sending notification messages to TSO users and operators. You then can decide which programs to abend and which programs to authorize.

For example, to detect read violations for the SYS1.LINKLIB data set and record violations in an SMF record, you would specify the following statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION=(ACCESSCHECK,SMF,SUPPRESSMESSAGES)
```

To prevent unauthorized read operations, specify OPTION=(ACCESSCHECK,ABEND). When you do this, EDIF abends programs that attempt to read a data set without authorization. Use caution when specifying OPTION=(ACCESSCHECK,ABEND) on any statement that affects a large number of data sets. We recommend that you do not specify the SUPPRESSMESSAGES option to suppress notification messages when the ABEND option is in effect.

If EDIF detects an unauthorized read operation, then EDIF issues message MIM4069 to notify you about the read violation and message MIM4070 to notify you that the program has been abended.

For example, to prevent unauthorized read operations for the SYS1.LINKLIB data set and record read violations in an SMF record, specify the following statement in the EDIPARMS member:

```
DATASET NAME=SYS1.LINKLIB OPTION=(ACCESSCHECK,ABEND,SMF)
```

By default, EDIF uses abend code U913 when abending a program. To use a different abend code, specify a new code through the ABENDCODE parameter on the EDIINIT statement.

Example: Preventing Unauthorized Read Operations

Suppose that you want EDIF to perform the following tasks for data sets that have the suffix .PAYROLL:

- Allow programs SYS1PAY and SYS2PAY to read these data sets
- Abend all other programs that try to read these data sets
- Record read violations in an SMF record

To do this, specify the following statement in the EDIPARMS member:

```
SUFFIX NAME=.PAYROLL OPTION=(ACCESSCHECK,ABEND,SMF),
    ACCESSLIST=(PROGRAMS=(SYS1PAY,SYS2PAY))
```

How You List Programs Receiving the Same Processing

When you specify the AUTHORIZED, EXEMPT, CHECKEXCLUSIVE, or ACCESSLIST parameters on an EDIF processing statement, you need to identify the programs to which that parameter applies. One way to do this is to list these programs individually. Another way is to list these programs on a UTILITY statement and then specify the name of that statement on the appropriate parameter. This lets you create a single list of programs that should receive the same type of processing and then reference that list on more than one statement.

You need to assign each UTILITY statement a name through the NAME parameter. This is the name that you can substitute on other EDIF processing statements to identify the UTILITY statement that contains the list of programs you want to use. For example, if you specify NAME=ISPF on a UTILITY statement, then you can specify the value AUTHORIZED(UTILITY(ISPF)) on a DATASET statement to tell EDIF to use this UTILITY statement to identify authorized programs.

You also need to list the programs that should be treated the same way. You can use the PROGRAMS parameter to do this. For example, suppose that you want only the programs IEWLF128, ISRUDA, ISRURS, and IEBCOPY to update the SYS1.LINKLIB and PROD1.MYDATA data sets. To do this, specify the following statements in the EDIPARMS member:

```
UTILITY NAME=ISPF PROGRAMS=(IEWLF128,ISRUDA,ISRURS,IEBCOPY)

DATASET NAME=SYS1.LINKLIB OPTION=(UTILITY,ABEND),
    AUTHORIZED=(UTILITY(ISPF))

DATASET NAME=PROD1.MYDATA OPTION=(UTILITY,ABEND),
    AUTHORIZED=(UTILITY(ISPF))
```

Because you identified these programs on a UTILITY statement named ISPF, you can specify UTILITY(ISPF) instead of listing all of these programs on each DATASET statement.

How You Dynamically Modify the EDIF Processing List

You can dynamically modify the EDIF processing list in the following ways:

- You can refresh the processing list so that EDIF uses the current contents of that list rather than the contents in place when you started CA MIM. This lets you change information in the processing list and make those changes take effect immediately.
- You can change the member that EDIF is using. This, for example, lets you replace a test-processing list with a production-processing list while EDIF is running.

You can use the MEMBER parameter on the SETOPTION command to perform either task while CA MIM is running.

Product Activation and Termination Considerations

For information about starting and stopping CA MII, see the CA MIM *Programming Guide*.

Performance Considerations

If you are experiencing ENQ delays or bottlenecks on your system, follow these steps:

1. Examine ENQ/RESERVE activity.
2. Verify that you are not converting hardware reserves to global ENQs or propagating system or systems ENQs against the QNAME recommendations of the vendor. Use the display F MIMGR,DISPLAY COUNTS command to examine ENQ/RESERVE activity. Do not propagate ENQs for non-shared resources.
3. Refine the QNAME List.

   For customers running with GDIINIT PROCESS=SELECT, delete QNAMEs that should not be managed from the QNAME list, rather than coding GDIF=NO. Do not code EXEMPT=YES if there are no entries in the EXEMPT list that apply. Also code TRACE=NONE for all QNAMEs, unless there is a specific need to do tracing.
4. Refine the EXEMPT List.

   If you are running with DEFAULT RESOURCE=GLOBAL JOB=GLOBAL, remove any unnecessary GLOBAL statements, which increase search time. Similarly, with either RESOURCE=LOCAL OR JOB=LOCAL, remove any unnecessary LOCAL statements.
5. Dispatch priority and performance of the CA MIM address space

   Note: For detailed information on the CA MIM dispatching priority, control file tuning, and performance tips, see the chapter “Advanced Topics” in the CA MIM *Programming Guide*.

Sysplex Considerations

CA MIM provides support for sites with z/OS images configured in XCFLOCAL, MONPLEX, or Multi-System sysplex modes.

If none of the images in your CA MIM complex are in Multi-System sysplex mode, CA MII requires no changes. (Multi-System sysplex mode is indicated by the PLEXCFG=MULTISYSTEM specification in the IEASYSxx member of SYS1.PARMLIB.)

If any of the images are in Multi-System sysplex mode, then you need to activate GRS ENQ propagation by specifying GRS=START, JOIN, TRYJOIN, or STAR in the IEASYSxx member of SYS1.PARMLIB.

CA MII can co-exist with GRS ENQ propagation, but CA MII does not propagate any ENQ request that GRS handles. We highly recommend that you specify GRSRNL=EXCLUDE in the IEASYSxx member of SYS1.PARMLIB. Failure to do so causes unpredictable results.

This specification directs GRS to bypass RNL processing and propagate *only* the ENQ requests that specify RNL=NO. This causes GRS to bypass processing of most ENQ requests, making them eligible for CA MII processing.

If your site is running a multi-system sysplex with CA MII, it is important that you do not specify SCOPE=ALL for any QNAMEs in your QNAME list unless specifically recommended by the vendor responsible for the QNAME.

Review the QNAMEs in your QNAME list that have SCOPE=ALL specified. These QNAMEs should have a SCOPE of SYSTEMS, unless specifically recommended otherwise by the vendor responsible for the QNAME. SYSDSN and SYSZVOLS are exceptions. SYSDSN and SYSZVOLS should have a SCOPE of SYSTEM.

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

z/VM Considerations

CA MII operates only on the z/OS operating system. There is no corresponding component for the z/VM operating system. If you share resources between both a z/OS and a z/VM system, you need to ensure that the resource is serialized through hardware reserves. You should not convert reserves issued by a z/OS system that protect a resource that is shared with a z/VM system. CA ACF2 and CA Top Secret are examples of products that can run on both z/OS and z/VM.

Chapter 4: Troubleshooting

This section contains the following topics:

Global Data Integrity Facility Problems

This section discusses problems with the Global Data Integrity Facility (GDIF).

ENQ Delays and System Performance Degradation

If you experience ENQ delays or bottlenecks on your system, follow these steps:

1. Examine ENQ/RESERVE activity.

   Verify that you are not converting hardware reserves to global ENQ requests, or propagating SCOPE=SYSTEM or SCOPE=SYSTEMS ENQs against the QNAME recommendations of the vendor. Use the display F MIMGR,DISPLAY COUNTS command to examine ENQ/RESERVE activity. Do not propagate ENQ requests for non-shared resources.

2. Refine the QNAME list.

   QNAMEs that should not be managed should be deleted from the QNAME list for customers running with GDIINIT PROCESS=SELECT rather than coding GDIF=NO. Do not code EXEMPT=YES if there are no applicable entries in the EXEMPT list. Also code TRACE=NONE for all QNAMEs, unless there is a specific need to do tracing.

3. Refine the EXEMPT list.

   Customers running with DEFAULT RESOURCE=GLOBAL JOB=GLOBAL should remove any unnecessary GLOBAL statements. This only increases search time. Similarly, with either RESOURCE=LOCAL OR JOB=LOCAL remove any unnecessary LOCAL statements.

CA MIM Space Performance Considerations

For information on addressing problems with CA MIM address space performance, see the CA MIM *Programming Guide*.

ENQ and RESERVE Hangs

If you experience a “hang” rather than a delay, we recommend that you gather the following information:

1. Issue the following commands on all systems:

   ```
   F MIMGR,DISPLAY CONFLICTS
   F MIMGR,DISPLAY RESERVES
   ```

   Note: For examples of the CA MIM displays that result from issuing DISPLAY commands, see the *Command and Statement Reference*.

2. Issue the following z/OS command on all systems:

   ```
   D GRS,C
   ```

3. Issue the following command to verify the complex is synchronized:

   ```
   F MIMGR,DISPLAY SYSTEMS
   ```

4. After analyzing the output from the previous displays, determine which systems are involved. Then take SVC dumps on each system of:

   - CA MIM address space
   - Any job or started task that appears involved

   Notes:

   - Multiple address spaces can be dumped using the MIM SYSDUMP command, and this is the preferred method.
   - For more information on the SYSDUMP command, see the *Statement and Command Reference Guide.*

5. Issue the following command on all systems:

   ```
   F MIMGR,DISPLAY QNAMES
   F MIMGR,DISPLAY EXEMPT
   F MIMGR,DISPLAY COUNTS
   F MIMGR,DISPLAY FACILITIES
   ```

CA MII Diagnostic Commands

Use the following commands to display information that can help you diagnose ENQ conflict conditions:

- F MIM, D RES=HARD
- F MIM, D RES=CONV
- F MIM, D CONFLICTS
- F MIM, D ENQR=(*qname*, *rname*)

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

z/OS Diagnostic Commands

Descriptions of the z/OS diagnostic commands follow.

DISPLAY GRS,CONFLICT

Displays information about a LOCAL ENQ conflict. The GRS component of z/OS is responsible for maintaining integrity in a single z/OS system. CA MII does not manage LOCAL ENQ conflicts, only GLOBAL conflicts. This command returns a display similar to the following:

```
D GRS,C
S=SYSTEM  SYSDSN   KEEWI01.TEST
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
XE03      TESTENQ            0056      007D6D90 EXCLUSIVE    OWN
XE03      KEEWI01A           003D      007E6858 EXCLUSIVE    WAIT
XE03      KEEWI01B           003E      007E6858   SHARE      WAIT
XE03      MIM119WK           0060      0064E840   SHARE      WAIT
NO LATCH CONTENTION EXISTS
```

You can also use this command to identify a global ENQ conflict. You can tell that it is a GLOBAL ENQ conflict because the CA MIM address space owns the resource on the local z/OS system. This is normal and is referred to as the "Blocking ENQ." CA MIM raises this ENQ on the local system to prevent access to a resource that is currently held on an external system as shown.

```
D GRS,C
ISG343I 07.46.53 GRS STATUS 148
S=SYSTEM  SYSDSN   KEEWI01.TEST
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
XE13      MIM119WK           0091      0064E838 EXCLUSIVE    OWN
XE13      BATCHJ01           003A      007FF3A0   SHARE      WAIT
NO LATCH CONTENTION EXISTS
```

To find out which system actually owns the resource, issue the following CA MIM command:

```
DISPLAY CONFLICT
```

This command returns a display similar to the following:

```
F MIM119WK,D CON
MIM0067I Command DISPLAY 212
MIM1028I CONFLICT display
  QNAME     RNAME                          Requestor      Type
  SYSDSN    KEEWI01.TEST                   S=XE03      HOLDS EXCL
                                           J=BATCHJ01  WAITS SHR
```

DISPLAY GRS,RES=(*qname,rname*)

Used to determine who owns a resource on the local system. The resource can or cannot be involved in a local or global ENQ conflict. You can use the * character to mask the remainder of the QNAME or RNAME. This command displays the following:

```
D GRS,RES=(SYSDSN,SYS2.CLIS*)
ISG343I 10.46.16 GRS STATUS 787
S=SYSTEM  SYSDSN   SYS2.CLIST
SYSNAME       JOBNAME         ASID     TCBADDR   EXC/SHR  STATUS
XE07          USER001         001A     008FDE88  SHARE    OWN
```

GDIF User Abends

User abend U0095 RC=1708 or RC=1635 occurs whenever CA MII detects that it has an ENQ control block (GQB) for a resource that is not currently outstanding.

Try the following solutions:

- If Global-GRS is active, as a requirement for implementing SYSPLEX, then verify that the QNAME on the MIM1059 message does not have SCOPE=ALL specified on the QNAME statement. We no longer recommend SCOPE=ALL, unless specifically recommended by the vendor.

  Note: For more information on using CA MII in a sysplex, review the sysplex information in the chapter "Advanced Topics (see page 27)."

  Look for any ISG* messages in the syslog that may indicate that GRS is having problems. Contact IBM for problem resolution.

Enhanced Data Set Integrity Facility Problems

To diagnose Enhanced Data Set Integrity Facility (EDIF) problems, use the EDITEST command.

The EDITEST command shows you what processing options are in effect for a data set. You need to specify the data set name on the DSNAME parameter and, if you have specified any DSORG statements in the EDIPARMS member, then you also need to specify the appropriate data set organization on the DSORG parameter.

For example, suppose that you want to see what options are in effect for the SYS1.LINKLIB data set and you know that you have specified at least one DSORG statement in the EDIPARMS member. To display processing options for this data set, you could issue this EDITEST command:

```
EDITEST DSNAME=SYS1.LINKLIB DSORG=PO
```

When you issue this command, CA MIM displays the processing options in effect for this data set on message MIM4063.

Activate Event Tracing

You can activate event tracing for the GDIF, EDIF, and ECMF facilities using the following commands. To activate tracing, you must first issue the SETOPTION MIM TRACE command.

Note: For more information about the SETOPTION GDIF, SETOPTION EDIF, and SETOPTION ECMF commands, see the *Statement and Command Reference Guide.*

Activate Trace Feature

To activate and deactivate the TRACE feature, issue the following command:

```
SETOPTION MIM TRACE=options
```

This command allows the limit of some types of tracing by job name.

By default, tracing is off.

Set Tracing Options

To activate the tracing function for the specified facility and selected options, use the command SETOPTION *facility* SETTRACE=(*options*) . The SETTRACE parameter enables the recording of specific program events in the CA MIM internal trace table.

Activate the Printing Function

To activate the printing function for the specified facility and selected options, use the command SETOPTION *facility* SETPRINT=(*options*).

The SETPRINT parameter controls whether events that are recorded in the CA MIM internal trace table are also sent to the MIM trace data set. SETPRINT is dependent upon SETTRACE, because only operands that have been specified in both SETTRACE and SETPRINT are sent to the MIM trace data set.

For example, to record SVC20 and OPEN processing trace events, issue:

```
SETOPTION EDIF SETTRACE=(SVC20,OPEN)
SETOPTION EDIF SETPRINT=(SVC20,OPEN)
```

Tracing is activated and trace records are then recorded in the MIM trace data set or spool file.

Reset Tracing Options

To reverse or turn off settings that were previously made using SETTRACE, use the command SETOPTION *facility* RESETTRACE. After RESETTRACE is issued, the options you specify are no longer recorded in the CA MIM internal trace table.

Reset Printing Options

To reverse or turn off settings that were previously made using SETPRINT, use the command SETOPTION *facility* RESETPRINT. After RESETPRINT is issued, the options specified are no longer recorded to the MIM trace data set. The options continue to be stored in the CA MIM internal trace table, when RESETTRACE has not been issued.

Dumps

Use the following commands to obtain dumps. For details, see the *Statement and Command Reference Guide*.

Obtain a System Dump

The SYSDUMP command lets you obtain a system dump of CA MIM, and optionally, a list of other named address spaces. You can request the dump on the issuing system, all systems in the CA MIM complex, and all external systems to the issuing system, or a list of selected systems in the complex.

The SYSDUMP command must be issued from a console or a TSO session. You cannot issue this command from the parameter data set.

You must be authorized to issue system control commands to issue the SYSDUMP command. TSO users generally are not authorized to issue system control commands.

To dump the CA MII and the IBM GRS address spaces on all MIIplex system, enter the following:

```
SYSDUMP JOBNAME=(GRS),SYSID=ALL
```

Chapter 5: User Exits

This section contains the following topics:

CA MII User Exits

You can use the following exit routines to customize CA MII processing:

EDIABNXT

Keeps EDIF from abending tasks when it detects inappropriate updates. This exit routine is available only when you run EDIF.

EDIATRXT

Modifies the EDIF attribute verification processing for data sets. This exit routine is available only when you run EDIF.

EDIOPTXT

Sets the status of EDIF processing options for various programs or data sets. This exit routine is available only when you run EDIF.

GDIXMPXT

Exempts resources from global ENQ propagation. This exit routine is available only when you run GDIF.

XCMCMDXT

Issues installation-specific commands in place of commands that ECMF issues when requeuing batch jobs or sending messages to a JOBLOG data set. This exit routine is available only when you run ECMF.

XCMCNFXT

Changes the way unconditional resource conflicts are processed. This exit routine is available only when you run ECMF.

XCMMSGXT

Changes the way resource requesters are notified of a resource conflict when they request a resource that is not available. This exit routine is available only when you run ECMF.

XCMNAVXT

Changes the way conditional resource conflicts are processed. This exit routine is available only when you run ECMF.

XCMNFYXT

Changes the way TSO resource owners are notified when they are involved in a resource conflict. This exit routine is available only when you run ECMF.

XCMPGMXT

Changes the way conditional resource conflicts are processed. This exit routine is available only when you run ECMF.

XCMREQXT

Changes the way batch job requeue processing occurs for batch jobs that are waiting for control of data sets. This exit routine is available only when you run ECMF.

SETOPTION EXIT Command

CA MII provides optional exit routines that allow you to do the following:

- Change the way GDIF processes ENQ requests
- Control how ECMF handles data set conflicts
- Modify EDIF processing

You use SETOPTION commands to manage CA MII exit routines. The EXIT parameter lets you identify the exit routine you want to control. Additional parameters let you designate whether routines should remain active under normal and abend conditions, and let you provide a means to trace data and recover from abends.

For example, to control the GDIXMPXT exit routine, you could issue the following command:

```
SETOPTION EXIT=(GDIXMPXT)
```

Display Exit Routine Information

To see information about exit routines you are using, issue the following command:

```
DISPLAY EXIT
```

CA MIM displays each logical exit routine name with its load module, address, status, and values set for protection and dump generation. The next screen shows the display for exit routine information:

```
F MIMGR,D EXIT

MIM0264I MIM EXIT display:
  XCMCNFXT USRCNFXT 8CB6E938 ACTIVE YES  YES  NO
  XCMNAVXT USRNAVXT 8CB0E028 ACTIVE YES  YES  NO
  XCMNFYXT USRNFYXT 8CB04040 ACTIVE YES  YES  NO
  XCMMSGXT USRMSGXT 8CB6E598 ACTIVE YES  YES  NO
```

MIMINIXT (Common Exit Interface)

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

Note: For more information, see the *CA MIM Programming Guide*.

How You Code Exit Routines

Follow these rules when coding any of the CA MII exit routines:

- Adhere to the restrictions as documented by each exit point.
- Make each routine reentrant.
- Routines are called in supervisor state, key zero or key eight. Be extremely careful not to violate storage boundaries.
- These routines can be called in either 24-bit or 31-bit mode. You can switch addressing modes (AMODEs) in a routine as long as the routine returns to the invoking program in the original addressing mode.

For additional rules for these routines, see Exit Routine Programming Considerations (see page 90).

To code a user exit

1. Use the EXIT parameter on the SETOPTION command to identify the appropriate load module for your routine.

   For example, specify SETOPTION EXIT=GDIXMPXT when the module name is GDIXMPXT. If you use a module name different from the default module name, then specify your name on the LOAD parameter. For example, to use the load module USRXMPXT for the GDIXMPXT routine, specify SETOPTION EXIT=(GDIXMPXT LOAD=USRXMPXT).

2. Customize the sample JCL in the ASMEXITS member of the CAI.CBTDJCL file to assemble and link-edit each completed exit routine as an authorized program, using the module name you specified on the SETOPTION command. Link-edit the routines into the authorized library that contains the CA MIM load modules.

Sample Exit Routines

Samples of the CA MII exit routines are contained in the CAI.CBTDSAMP data set. You can use these sample routines, or you can create your own routines. The CAI.CBTDMAC data set also contains mapping macros for the exit-specific parameter lists.

Exit Routine Programming Considerations

Use the guidelines in this section when coding your exit routines using the samples.

EDIABNXT Exit

The EDIABNXT exit routine lets you keep EDIF from abending a task for one of the following causes:

- A read-access violation
- A DISP=SHR violation
- A utility violation
- An exempt violation
- An attribute violation

EDIF calls this exit routine directly before EDIF abends the task.

Entry Environment

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

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this standard parameter list. It has the following format:

+0

Fullword containing the address of the parameter list of the EDIABNXT exit routine. This is not used for initialization calls.

+4

Fullword containing the communication word of the EDIABNXT routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during the second initialization, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the routine. The high order bit of the first byte is set on X'80' for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of one standard save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the EDIABNXT routine.

All other registers are undefined.

EDIATRXT Exit

The EDIATRXT exit routine lets you modify the EDIF attribute verification processing for data sets. It is called after EDIF has performed two different attribute merges but before the results of these two merges are compared. The two attribute merges are:

- Attributes from EDIF processing statements and the DSCB of the data set
- Attributes from the DCB, DCB exit (if one exists), system JFCB, and JFCB exit (if one exists)

Entry Environment

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

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this standard parameter list. It has the following format:

+0

Fullword containing the address of the parameter list of the EDIATRXT exit routine. This is not used for initialization calls.

+4

Fullword containing the communication word of the EDIATRXT routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during the second initialization, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the routine. The high order bit of the first byte is set on X'80' for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of one standard save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the EDIATRXT routine.

All other registers are undefined.

EDIATRXT Exit-specific Parameter List

+0

Fullword containing the address of an EDIF exit control block (EDIXCB). This control block contains various fields that you can modify.

All other registers are undefined.

Return Codes

There is no return code. You tell EDIF what to do by modifying fields in the EDIF exit control block (EDIXCB).

Note: You must restore all registers (except register 15) upon return.

EDIOPTXT Exit

The EDIOPTXT exit routine lets you set the status of EDIF processing options for various programs or data sets. EDIF calls this exit after it has determined the processing options it should perform but before it actually processes any of them.

Entry Environment

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

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this standard parameter list. It has the following format:

+0

Fullword containing the address of the parameter list of the EDIOPTXT exit routine. This is not used for initialization calls.

+4

Fullword containing the communication word of the EDIOPTXT routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during the second initialization, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the routine. The high order bit of the first byte is set on X'80' for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of one standard save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the EDIOPTXT routine.

All other registers are undefined.

EDIOPTXT Exit-specific Parameter List

+0

Fullword containing the address of an EDIF exit control block (EDIXCB). This control block contains various fields that you can modify.

All other registers are undefined.

Return Codes

There is no return code. You tell EDIF what to do by modifying fields in the EDIF exit control block (EDIXCB).

Note: You must restore all registers (except register 15) upon return.

GDIXMPXT Exit

The GDIXMPXT exit routine is called for a resource any time that GDIF needs to evaluate the exemption/conversion status for the resource. Therefore, the GDIXMPXT exit may be called multiple times for the same resource.

The processing in the GDIXMPXT exit routine can take place in the resource requester's address space or in the GRS address space. Therefore, the exit must reside in common storage.

The GDIXMPXT exit routine can influence resource exemption/reserve conversion processing based on criteria such as QNAME, RNAME, job name, VOLSER, etc. You can customize the exemption/conversion actions by setting the appropriate flags in GDIXMPXP parameter list (field XMPX_AFLGS). The GDIXMPXT exit routine should pass a return code value of 4 back to its caller when altering the exemption/conversion actions.

Entry Environment

Supervisor state; Storage Protect Key Zero; Enabled; Locks may or may not be held; AMODE 31; May be called with an EUT FRR held; Primary ASC mode; one of the following cross-memory environments:

- PASN = SASN = HASN (requester's address space)
- PASN=GRS SASN=HASN=requester
- PASN = SASN = HASN (CA MIM address space)

.

Programming Restrictions

- The GDIXMPXT exit routine cannot implicitly or explicitly cause the current task to be placed in a WAIT.
- The GDIXMPXT exit routine may not call any system services that conflict with the execution environment.
- The GDIXMPXT exit routine must reside in common storage.

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list for the exit. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this parameter list. It has the following format:

+0

Fullword containing the address of the exit-specific parameter list for the GDIXMPXT exit routine. The exit-specific parameter list is mapped by the GDIXMPXP mapping macro that is contained in the CAI.CBTDMAC data set.

Note: This is not used for exit initialization calls.

+4

Fullword word containing the communication word of the GDIXMPXT exit routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during any subsequent initialization calls, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the exit routine. The high order bit of the first byte is set on (X'80') for the initialization call.

+C

Fullword word containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of a standard 18-word register save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the GDIXMPXT exit routine.

All other register contents are undefined upon entry to the GDIXMPXT exit routine.

GDIXMPXT Exit-specific Parameter List

Offset +0 in the common exit parameter list (mapped by UXPARM) points to the GDIXMPXT exit routine-specific parameter list. See member GDIXMPXP in the CAI.CBTDMAC data set for a mapping macro for this parameter list. You can set various flags and values in the GDIXMPXP parameter list to influence conflict actions for the requester involved in the resource conflict.

Return Codes

R15

Contains one of the following return codes when control is returned to CA MIM:

0

Continue normal exemption / conversion processing for this resource based on the policies set in the QNAME list and RNAME list and the TEMPORARYDSN parameter.

4

Perform exemption / conversion actions for this resource according to the actions selected by the exit routine in the GDIXMPXP parameter list.

XCMCMDXT Exit

Use the XCMCMDXT exit routine when a JES2 command is to be issued to perform job requeue processing. The installation user exit can modify the command text when using modified JES2 command syntax.

Upon entry to the XCMCMDXT exit routine, the required command text has been built in the CMDX_CMDTX field of the CMDX_PL parameter list. If necessary, the installation exit can alter or replace the command text in the CMDX_CMDTX field.

The exit can manipulate the command action flags in the CMDX_FLG1 byte. The CONSOLE name and the UTOKEN can be customized (replaced) by the exit; however, the replacement values must be valid. For example, if the XCMCMDXT exit routine supplies a different CONSOLE name, that CONSOLE name must reflect a valid, previously defined MCS or extended MCS CONSOLE.

Entry Environment

Supervisor state; Storage Protect Key Zero; Enabled; No locks held; Task Mode; AMODE 31; PASN = SASN = HASN; Primary ASC mode.

Programming Restrictions

The XCMCMDXT exit routine cannot implicitly or explicitly cause the current task to be placed in a WAIT; therefore, the XCMCMDXT exit routine should not issue the command directly through MGCRE (SVC 34). Instead, it should let CA MII schedule the command in the appropriate execution environment.

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list for the exit. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this parameter list. It has the following format:

+0

Fullword containing the address of the exit-specific parameter list for the XCMCMDXT exit routine. The exit-specific parameter list is mapped by the XCMCMDXP mapping macro that is contained in the CAI.CBTDMAC data set.

Note: This is not used for exit initialization calls.

+4

Fullword containing the communication word of the XCMCMDXT exit routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, the value set during the first initialization call is passed during any subsequent initialization calls, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the exit routine. The high order bit of the first byte is set on (X'80') for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of a standard 18-word register save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the XCMCMDXT exit routine

All other register contents are undefined upon entry to the XCMCMDXT exit routine.

XCMCMDXT Exit-specific Parameter List

Offset +0 in the common exit parameter list (mapped by UXPARM) points to the XCMCMDXT exit routine-specific parameter list. See member XCMCMDXP in the CAI.CBTDMAC data set for a mapping macro for this parameter list. You can set various flags and values in the XCMCMDXP parameter list to influence conflict actions for the requester involved in the resource conflict.

Return Codes

R15

Contains one of the following return codes when control is returned to CA MIM:

0

Continue normal processing.

4

Ignore this request and do not issue the command.

XCMCNFXT Exit

The XCMCNFXT exit routine is called once when it is first detected that a resource requester is in conflict for resource ownership and is queued for control of the target resource. This exit is called for unconditional ENQ/ISGENQ requests when the resource requester is queued to control the target resource. For information on processing conflicts generated by conditional ENQ/ISGENQ requests, see the XCMNAVXT exit routine.

The processing in the XCMCNFXT exit routine takes place in the CA MII address space and is based on the job name or TSO user ID of the 'victim' in the resource conflict and not the job name or TSO user holding the resource in conflict.

The XCMCNFXT exit routine can influence the conflict processing for a specific resource requester by overriding the values of REPORTAFTER seconds and REPORTCYCLE seconds in the XCMCNFXP parameter list (fields CNFX_RPTA and CNFX_RPTC, respectively). You can customize additional conflict actions by setting the appropriate flags in XCMCNFXP parameter list (field CNFX_FLG1). The XCMCNFXT exit routine should pass a return code value of 4 back to its caller when altering the conflict actions.

If the goal of the installation user exit is to customize conflict message processing, you may want to use the XCMMSGXT or the XCMNFYXT exit points instead.

Entry Environment

Supervisor state; Storage Protect Key Eight; Enabled; No locks held; Task Mode; AMODE 31; PASN = SASN = HASN (CA MII address space); Primary ASC mode.

Programming Restrictions

The XCMCNFXT exit routine cannot implicitly or explicitly cause the current task to be placed in a WAIT.

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list for the exit. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this parameter list. It has the following format:

+0

Fullword containing the address of the exit-specific parameter list for the XCMCNFXT exit routine. The exit-specific parameter list is mapped by the XCMCNFXP mapping macro that is contained in the CAI.CBTDMAC data set.

Note: This is not used for exit initialization calls.

+4

Fullword containing the communication word of the XCMCNFXT exit routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during any subsequent initialization calls, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the exit routine. The high order bit of the first byte is set on (X'80') for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of a standard 18-word register save area

R14

Contains the return address for CA MIM

R15

Contains the entry point address for the XCMCNFXT exit routine.

All other register contents are undefined upon entry to the XCMCNFXT exit routine.

XCMCNFXT Exit-specific Parameter List

Offset +0 in the common exit parameter list (mapped by UXPARM) points to the XCMCNFXT exit routine-specific parameter list. See member XCMCNFXP in the CAI.CBTDMAC data set for a mapping macro for this parameter list. You can set various flags and values in the XCMCNFXP parameter list to influence conflict actions for the requester involved in the resource conflict. Return to the caller with a return code of 4 in R15 when requesting any specific actions in the XCMCNFXP parameter list.

Return Codes

R15

Contains one of the following return codes when control is returned to CA MIM:

0

Continue normal processing.

4

Continue processing this requester conflict using the actions specified in the XCMCNFXP parameter list.

8

Ignore this requester conflict.

XCMMSGXT Exit

The XCMMSGXT exit routine is called before messages are generated on behalf of a resource requester that is in conflict and is waiting for ownership of the target resource. This processing takes place on the same system that the resource requester resides. This exit is called for both conditional and unconditional ENQ conflicts.

The processing in the XCMMSGXT exit routine takes place in the CA MIM address space and is provided data on the conflicting resource name, the resource requester that is in conflict, and the resource owner.

The XCMMSGXT exit routine can influence the conflict message processing for a specific resource requester by setting the appropriate flags in XCMMSGXP parameter list (field MSGX_FLG1). The XCMMSGXT exit routine should pass a return code value of 4 back to its caller when altering the conflict actions.

Entry Environment

Supervisor state; Storage Protect Key Eight; Enabled; No locks held; Task Mode; AMODE 31; PASN = SASN = HASN (CA MII address space); Primary ASC mode.

Programming Restrictions

The XCMMSGXT exit routine cannot implicitly or explicitly cause the current task to be placed in a WAIT.

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list for the exit. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this parameter list. It has the following format:

+0

Fullword containing the address of the exit-specific parameter list for the XCMMSGXT exit routine. The exit-specific parameter list is mapped by the XCMMSGXP mapping macro that is contained in the CAI.CBTDMAC data set.

Note: This is not used for exit initialization calls.

+4

Fullword containing the communication word of the XCMMSGXT exit routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during any subsequent initialization calls, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the exit routine. The high order bit of the first byte is set on (X'80') for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of a standard 18-word register save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the XCMMSGXT exit routine.

All other register contents are undefined upon entry to the XCMMSGXT exit routine.

XCMMSGXT Exit-specific Parameter List

Offset +0 in the common exit parameter list (mapped by UXPARM) points to the XCMMSGXT exit routine-specific parameter list. See member XCMMSGXP in the CAI.CBTDMAC data set for a mapping macro for this parameter list. You can set various flags and values in the XCMMSGXP parameter list to influence conflict actions for the requester involved in the resource conflict. Return to the caller with a return code of 4 in R15 when requesting any specific actions in the XCMMSGXP parameter list.

Return Codes

R15

Contains one of the following return codes when control is returned to CA MIM:

0

Continue normal processing.

4

Continue processing this requester conflict using the actions specified in the XCMMSGXP parameter list.

8

Suppress all conflict message processing for this resource requester.

XCMNAVXT Exit

The XCMNAVXT exit routine is called a resource requester has made a conditional request for resource ownership for which control cannot be immediately granted. These are conditional ENQ/ISGENQ requests for which resource control is not immediately granted and the requester is not queued for resource control. For information on processing conflicts generated by unconditional ENQ/ISGENQ requests, see the XCMCNFXT exit routine.

The processing in the XCMNAVXT exit routine takes place in the CA MII address space and is based on the job name or TSO user ID of the 'victim' in the failed conditional resource request and not the job name or TSO user holding the target resource.

The XCMNAVXT exit routine can influence the conflict processing for a specific resource requester by setting the appropriate flags in XCMNAVXP parameter list (field NAVX_FLG1). The XCMNAVXT exit routine should pass a return code value of 4 back to its caller when altering the conflict actions.

If the goal of the installation user exit is to customize conflict message processing, you may want to use the XCMMSGXT or the XCMNFYXT exit points instead.

Entry Environment

Supervisor state; Storage Protect Key Eight; Enabled; No locks held; Task Mode; AMODE 31; PASN = SASN = HASN (CA MII address space); Primary ASC mode.

Programming Restrictions

The XCMNAVXT exit routine cannot implicitly or explicitly cause the current task to be placed in a WAIT.

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list for the exit. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this parameter list. It has the following format:

+0

Fullword containing the address of the exit-specific parameter list for the XCMNAVXT exit routine. The exit-specific parameter list is mapped by the XCMNAVXP mapping macro that is contained in the CAI.CBTDMAC data set.

Note: This is not used for exit initialization calls.

+4

Fullword containing the communication word of the XCMNAVXT exit routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during any subsequent initialization calls, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the exit routine. The high order bit of the first byte is set on (X'80') for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of a standard 18-word register save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the XCMNAVXT exit routine.

All other register contents are undefined upon entry to the XCMNAVXT exit routine.

XCMNAVXT Exit-specific Parameter List

Offset +0 in the common exit parameter list (mapped by UXPARM) points to the XCMNAVXT exit routine-specific parameter list. See member XCMNAVXP in the CAI.CBTDMAC data set for a mapping macro for this parameter list. You can set various flags and values in the XCMNAVXP parameter list to influence conflict actions for the requester involved in the resource conflict. Return to the caller with a return code of 4 in R15 when requesting any specific actions in the XCMNAVXP parameter list.

Return Codes

R15

Contains one of the following return codes when control is returned to CA MIM:

0

Continue normal processing.

4

Continue processing this requester conflict using the actions specified in the XCMNAVXP parameter list.

8

Ignore this requester conflict.

XCMNFYXT Exit

The XCMNFYXT exit routine is called before notifying a TSO user that owns a resource that a given requester is in conflict (conditional or unconditional) for ownership of the target resource. This exit may be called on a different system from that were the resource requester is executing,

The processing in the XCMNFYXT exit routine takes place in the CA MII address space and is provided data on the conflicting resource name, the owning TSO user, and the resource requester that is in conflict.

The XCMNFYXT exit routine can influence the conflict processing for a specific resource owner by setting the appropriate flags in XCMNFYXP parameter list (field NFYX_FLG1). The XCMNFYXT exit routine should pass a return code value of 4 back to its caller when altering the conflict notification actions.

Entry Environment

Supervisor state; Storage Protect Key Eight; Enabled; No locks held; Task Mode; AMODE 31; PASN = SASN = HASN (CA MII address space); Primary ASC mode.

Programming Restrictions

The XCMNFYXT exit routine cannot implicitly or explicitly cause the current task to be placed in a WAIT.

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list for the exit. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this parameter list. It has the following format:

+0

Fullword containing the address of the exit-specific parameter list for the XCMNFYXT exit routine. The exit-specific parameter list is mapped by the XCMNFYXP mapping macro that is contained in the CAI.CBTDMAC data set.

Note: This is not used for exit initialization calls.

+4

Fullword containing the communication word of the XCMNFYXT exit routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during any subsequent initialization calls, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the exit routine. The high order bit of the first byte is set on (X'80') for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of a standard 18-word register save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the XCMNFYXT exit routine.

All other register contents are undefined upon entry to the XCMNFYXT exit routine.

XCMNFYXT Exit-specific Parameter List

Offset +0 in the common exit parameter list (mapped by UXPARM) points to the XCMNFYXT exit routine-specific parameter list. See member XCMNFYXP in the CAI.CBTDMAC data set for a mapping macro for this parameter list. You can set various flags and values in the XCMNFYXP parameter list to influence conflict actions for the requester involved in the resource conflict. Return to the caller with a return code of 4 in R15 when requesting any specific actions in the XCMNFYXP parameter list.

Return Codes

R15

Contains one of the following return codes when control is returned to CA MIM:

0

Continue normal processing.

4

Continue processing this requester conflict using the actions specified in the XCMNFYXP parameter list.

8

Suppress all conflict notification processing for this resource owner.

XCMPGMXT Exit

The XCMPGMXT exit routine is called when a resource requester has made a conditional request for resource ownership for which control cannot be immediately granted. These are conditional ENQ/ISGENQ requests for which resource control is not immediately granted and the requester is not queued for resource control. For information on processing conflicts generated by unconditional ENQ/ISGENQ requests, see the discussion of the XCMCNFXT exit (see page 99) routine.

The processing in the XCMPGMXT exit routine takes place in the victim's address space and is based on the job name or TSO user ID of the 'victim' in the failed conditional resource request and not the job name or TSO user holding the target resource. The XCMPGMXT exit routine can influence the conflict processing for a specific resource requester by setting the appropriate flags in XCMPGMXP parameter list (field PGMX_FLG1). The XCMPGMXT exit routine should pass a return code value of 4 back to its caller when altering the conflict actions. If the goal of the installation user exit is to customize conflict message processing, you may want to use the XCMMSGXT or the XCMNFYXT exit points instead.

Entry Environment

Supervisor state; Storage Protect Key Eight; Enabled; No locks held; Task Mode; AMODE 31.

Programming Restrictions

The XCMPGMXT exit routine cannot implicitly or explicitly cause the current task to be placed in a WAIT.

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list for the exit. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this parameter list. It has the following format:

+0

Fullword containing the address of the exit-specific parameter list for the XCMPGNMXT exit routine. The exit-specific parameter list is mapped by the XCMPGMXP mapping macro that is contained in the CAI.CBTDMAC data set.

Note: This is not used for exit initialization calls.

+4

Fullword containing the communication word of the XCMPGMXT exit routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, then the value set during the first initialization call is passed during any subsequent initialization calls, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the exit routine. The high order bit of the first byte is set on (X'80') for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of a standard 18-word register save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the XCMPGMXT exit routine.

All other register contents are undefined upon entry to the XCMNFYXT exit routine.

XCMPGMXT Exit-specific Parameter List

Offset +0 in the common exit parameter list (mapped by UXPARM) points to the XCMPGMXT exit routine-specific parameter list. See member XCMPGMXP in the CAI.CBTDMAC data set for a mapping macro for this parameter list. You can set various flags and values in the XCMPGMXP parameter list to influence conflict actions for the requester involved in the resource conflict. Return to the caller with a return code of 4 in R15 when requesting any specific actions in the XCMPGMXP parameter list.

Return Codes

R15

Contains one of the following return codes when control is returned to CA MIM:

0

Continue normal processing.

4

Continue processing this requester conflict using the actions specified in the XCMPGMXP parameter list.

8

Ignore this requester conflict.

XCMREQXT Exit

The XCMREQXT exit routine is called once when a job in initiation is in conflict for data set ownership and the job requeue feature is active.

The processing in the XCMREQXT exit routine takes place in the address space of the batch job that is in initiation.

The XCMREQXT exit routine can influence requeue processing for the job in conflict by cancelling the job, immediately requeuing the job, or bypassing requeue for the job, based on criteria such as job name, ASCB, TCB, and data set name. You can customize requeue actions by setting the appropriate flags in the XCMREQXP parameter list (field REQX_FLG1). The XCMREQXT exit routine should pass a return code value of 4 back to its caller when altering the requeue actions.

Entry Environment

Supervisor state; Storage Protect Key Zero; Enabled; No locks held; Task Mode; AMODE 31; PASN = SASN = HASN (batch job address space); Primary ASC mode.

Adhere to the restriction for user-written type-2 SVC routines.

Note: For more information, see the IBM documentation

Programming Restrictions

The XCMREQXT exit routine cannot implicitly or explicitly cause the current task to be placed in a WAIT.

Register Usage at Entry

Common Parameter List

R1

Contains the address of the parameter list for the exit. The UXPARM member of the CAI.CBTDMAC data set contains a DSECT mapping for this parameter list. It has the following format:

+0

Fullword containing the address of the exit-specific parameter list for the XCMREQXT exit routine. The exit-specific parameter list is mapped by the XCMREQXP mapping macro that is contained in the CAI.CBTDMAC data set.

Note: This is not used for exit initialization calls.

+4

Fullword containing the communication word of the XCMREQXT exit routine. The content of this word may be set by the routine during its initialization call and is passed unchanged for all subsequent calls. This word is set to zero when the routine gets control over the first initialization call. If CA MIM is restarted, the value set during the first initialization call is passed during any subsequent initialization calls, unless the z/OS system has been IPLed.

+8

Fullword containing flags used to pass status information to the exit routine. The high order bit of the first byte is set on (X'80') for the initialization call.

+C

Fullword containing the global communication word, set by the MIMINIXT exit routine.

R13

Contains the address of a standard 18-word register save area.

R14

Contains the return address for CA MIM.

R15

Contains the entry point address for the XCMREQXT exit routine.

All other register contents are undefined upon entry to the XCMREQXT exit routine.

XCMREQXT Exit-specific Parameter List

Offset +0 in the common exit parameter list (mapped by UXPARM) points to the XCMREQXT exit routine-specific parameter list. For a mapping macro for this parameter list, see member XCMREQXP in the CAI.CBTDMAC data set. You can set various flags and values in the XCMREQXP parameter list to influence requeue actions for the job. Return to the caller with a return code of 4 in R15 when requesting any specific actions in the XCMREQXP parameter list.

Return Codes

R15

Contains one of the following return codes when control is returned to CA MIM:

0

Continue normal processing.

4

Provide requeue processing for this batch job according to the actions selected by the exit routine in the XCMREQXP parameter list.

8

Do not do requeue processing for this job.

Chapter 6: Utilities and Other Interfaces

This section contains the following topics:

Generate Reports for CA MII

CA MIM reports provide you with an effective reporting tool to evaluate the performance of the systems in your complex. The reports use a sampling of statistical data to yield important information about the operating activities in your complex.

Note: This functionality requires the CA Easytrieve Interface. For more CA Common Services requirements information, see the appendix "CCS for z/OS Component Requirements" in the *Installation Guide*.

How Report Generation Works

CA MIM uses a single SMF record for the record collection process. Using record *subtypes*, you can identify the statistical records to be collected for the different reports available under the CA MIM facilities.

The steps involved in the report generation process are:

1. Specify the SMF record number on a MIMINIT RECORDTYPE statement in the initialization member. Once this is set, you should not need to change it.
2. Specify the record collection criteria using the SETOPTION command and begin the record collection process.

   Note: You must allow the record collection process to run for a while before dumping the SMF records to data sets.
3. Dump the SMF records to a usable data set through the IBM IFASMFDP utility.
4. Run a report or create a report output file.

Important! CA MIM provides a set of reports for general use. If, however, any one of these reports does not meet your specific requirements, then you can write your own customized reports in the language of your choice, using the SMF records supplied by CA MIM. For more information on this topic, see the MIMSTREC member in the CAI.CBTDMAC data set.

Specify SMF Record Number

Before any statistical records can be collected for use in the CA MIM report process, you need to specify an SMF record number. To do this, you place a MIMINIT RECORDTYPE initialization statement in the initialization member. For example, if you decide to use the SMF record number 189, which is the default, you would specify the following statement in the initialization member:

```
MIMINIT RECORDTYPE=189
```

Now you are ready to specify the record collection criteria for the report or reports you want to run.

Specify Record Collection Criteria

To collect statistical records for CA MIM reports, you need to specify the type of statistical records to be collected by issuing the command SETOPTION STATCOLLECT with the specific CA MIM facility.

Specify the SUBTYPE operand on the SETOPTION STATCOLLECT command to select specific record subtypes for a facility, or specify ALL to collect records for all record subtypes available under that facility. These record subtypes correspond directly to the reports available for each facility.

Note: For more information on each record subtype, see Sample Reports (see page 118).

The record subtypes for CA MII are:

Record Subtype	Facility	Description
CR	ECMF	The CR record subtype is used to produce the CA MII Job Requeue (CR) report. This report provides a list of batch job requeue times as well as the names of the data sets causing the conflicts.
DC	EDIF	The DC record subtype is used to produce the CA MII Enhanced Data Set Integrity Benefits (DC) report. This report summarizes the data set integrity exposures prevented by EDIF.
DD	EDIF	The DD record subtype is used to produce the CA MII Enhanced Data Set Integrity Data Set Violation (DD) report. This report provides more specific information about integrity exposures detected and prevented by EDIF.

Record Subtype	Facility	Description
EC	GDIF	The EC record subtype is used to produce the following reports: CA MII QNAME Enqueue/Reserve Count (EC) report. This report provides detailed information about enqueues and reserves processed by GDIF.

To generate records for your CA MII facilities

- Start the record collection process for each CA MIM facility for which you want to run a report by issuing the SETOPTION command. For example:

  ```
  SETOPTION ECMF STATCOLLECT(SUBTYPE=CR)
  ```

  This command tells CA MIM to collect statistics for the report subtype CR.

- Specify the record collection sampling cycle and recording interval for ECMF.

  Important! To activate the record collection process for all record subtypes, you must repeat this procedure for each subtype, specifying the SETOPTION STATCOLLECT command and the facility associated with that record subtype.

- Set the sampling cycle time. Issue the following command:

  ```
  SETOPTION STATCYCLE=nn
  ```

  nn

  Specifies how long CA MIM waits between each statistical data sampling. The initial value for the STATCYCLE operand is 60 seconds.

- Specify the statistical record recording interval. Issue the following command:

  ```
  SETOPTION STATINTERVAL=nnn
  ```

  nnn

  Indicates how long CA MIM waits until it records the collected data in the form of actual statistical records. The initial value for the STATINTERVAL operand is 15 minutes.

Example: Starting Record Collection

The following example shows how to begin the record collection process for the CA MII Job Requeue report. This example indicates data sampling every 30 seconds, and a statistical record recording interval of every hour:

```
SETOPTION ECMF STATCOLLECT(SUBTYPE=CR) STATCYCLE=30, STATINTERVAL=60
```

Note: The STATCYCLE and STATINTERVAL values apply to all record collection subtypes for a given CA MIM facility.

Once the statistical record collection process has begun, records are collected and written to the SMF data sets. You need to use your site-specific procedures for dumping the records to physical sequential data sets that can be used by CA MIM to produce reports.

Notes:

- For more information on the SETOPTION commands used in this section, see the *Statement and Command Reference Guide*.
- For detailed information on using the SMF report generator, see the chapter "Utilities" in the *CA MIM Programming Guide*.

Sample Reports

This section contains samples of the various reports that can be generated.

CA MII Job Requeue Report (CR)

```
(1)                (2)                                                                                    (3)
2010/07/13 14:06   CA MII Job Requeue Report                                                          PAGE      1
(4) System:    SYSA             (5) Jobname:  MIMGR        (6)Release:   11.9        (7) Service Level:  0000
(8) From:      2009/10/25 00:00  (9) Each:     RECORD
(10)To:        2010/03/12 10:52  (11)Shift:    00:00 24:00
   (12)    (13)       (14)              (15)       (16)    (17)  (18)    (19)   (20) (21)
 Jobname   Job ID    Requeue Time     Release Time   Hold Time REQ  Owner    System  OWN Dataset Name
 -------- -------- ---------------- ---------------- --------- --- -------- -------- --- -------------------------------------------
 BATCHJBA JOB00941 2010/03/08 11:19 2010/03/08 11:40  20m 32s  EXC BATCHJBB SYSB     SHR MIMGR.REQUEUE.DSN
```

Field	Description
(1)	Date and time the report was created.
(2)	Report title.
(3)	Report page number.
(4)	System name of the CA MII image as defined by the DEFSYS statement.
(5)	Name of the CA MII started task.
(6)	The CA MII release number.
(7)	The CA MII service level.
(8)	Date and timestamp of the earliest record in the reporting interval.

Field	Description
(9)	Summary level for the data line. In this example, each RECORD is displayed in the detail line. If each HOUR was specified, then the detail line would be a summary of the hour's activity.
(10)	Date and timestamp of the latest record in the recording interval.
(11)	Range of records included in the report.
Jobname (12)	Job name of the job that was canceled and re-queued by CA MII due to a conflict with another job or jobs for one or more data sets.
Job ID (13)	JES2 assigned JOBID of the job that was re-queued.
Requeue Time (14)	Date and time that the job was canceled and re-queued for execution.
Release Time (15)	Date and time that the job was released from hold status by CA MII and was eligible for selection for execution. Selection for execution depends on available initiators, the execution selection priority of the released job in relation to other jobs queued for execution, and so forth.
Hold Time (16)	The amount of time in the form *hh:mm:ss* (hours, minutes, and seconds, respectively) that the specified job was not available for execution selection due to data set conflicts.
REQ (17)	The scope of control that the re-queued job requires for the named data set. EXC is exclusive control while SHR is shared control.
Owner (18)	The *jobname* of the job that currently owns the requested data set.
System (19)	The system as defined by a DEFSYS statement, upon which the data set owner is actively executing.
OWN (20)	The scope of control that the data set owner currently maintains. EXC is exclusive control while SHR is shared control.
Data Set Name (21)	The name of the data set in conflict.

CA MII Enhanced Data Set Integrity Benefits Report (DC)

```
(1)                     (2)                                                                                    (3)
2008/09/02 14:06   CA MII Enhanced Dataset  Integrity Benefits Report                                        PAGE      1
(4) System:     SYSA              (5) Jobname:  MIMGR       (6) Release:   11.9         (7)Service Level:  0000
(8) From:       2008/09/02 00:00  (9) Each:     RECORD
(10)To:         2008/09/02 10:52  (11)Shift:    00:00 24:00
                   (13)    (14)   (15)  (16)    (17)   (18)  (19)   (20)  (21)  (22)    (23)   (24)  (25)  (26)   (27)  (28)  (29)
       (12)        Open    Close  DCBs   DCBs   ENQ    ENQ   DEQ    ATTR  ATTR   EXCL   EXCL   UTIL  UTIL  ACC    ACC   COPY  COPY
  Interval Start   Count   Count  Exam.  Proc.  Issue  Conf. Issue  Tests  Fail  Tests  Fail  Tests  Fail  Tests  Fail  Exam. Proc.
 ---------------- ------- ------- ------ ------ ------ ----- ------ ------ ----- ------ ----- ------ ----- ------ ----- ----- -----
 2008/09/02 09:36     239     328    177     69     22     1     19     42     4      2     2      2     2      2     2     0     0
```

Field	Description
(1)	Date and time the report was created.
(2)	Report title.
(3)	Report page number.
(4)	System name of the CA MII image as defined by the DEFSYS statement.
(5)	Name of the CA MII started task.
(6)	The CA MII release number.
(7)	The CA MII service level.
(8)	Date and time stamp of the earliest record in the reporting interval.
(9)	Summary level for the data line. In this example, each RECORD is displayed in the detail line. If each HOUR was specified, then the detail line would be a summary of the hour's activity.
(10)	Date and time stamp of the latest record in the recording interval.
(11)	Range of records included in the report.
Interval Start (12)	Start date and timestamp that the report line interval data covers.
Open Count (13)	The number of SVC OPEN requests intercepted during the reporting interval. This value is similar to the INTERCEPT OPEN line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
Close Count (14)	The number of SVC CLOSE requests intercepted during the reporting interval. This value is similar to the INTERCEPT CLOSE line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.

Field	Description
DCBs Exam. (15)	The number of DCBs examined during the reporting interval. This value is similar to the TEST DCB line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
DCBs Proc. (16)	The number of DCBs that caused EDIF to take some action during the reporting interval. This value is similar to the PROCESS DCB line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
ENQ Issue (17)	The number of ENQs that EDIF raised during the reporting interval to protect access to data sets. This value is similar to the ENQ line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
ENQ Conf. (18)	The number of times during the reporting interval that two or more jobs tried to update the data set but were prevented from doing so by invocation of the EDIF ENQ. This value is similar to the UPDATE CONFLICT line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
DEQ Issue (19)	The number of DEQs that EDIF issued during the reporting interval for ENQs it had previously issued.
ATTR Tests (20)	The number of times during the reporting interval that EDIF checked the attributes of a data set against the attributes specified by a program or a user. This value is similar to the ATTRIBUTE TEST line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
ATTR Fail (21)	The number of times during the reporting interval that EDIF detected attribute violations. This value is similar to the ATTRIBUTE FAILURE line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
EXCL Tests (22)	The number of times during the reporting interval that EDIF checked the JCL of a job to determine whether a program would update a data set when DISP=SHR was specified. This value is similar to the CHECK EXCL TEST line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
EXCL Fail (23)	The number of times during the reporting interval that EDIF determined a job accessed a data set when DISP=SHR was specified. This value is similar to the CHECK EXCL FAILURE line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.

Field	Description
UTIL Tests (24)	The number of times during the reporting interval that EDIF checked to determine whether a program was authorized to update a data set. This value is similar to the UTILITY TEST line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
UTIL Fail (25)	The number of times during the reporting interval that EDIF detected updates by programs not authorized to update a data set. This value is similar to the UTILITY FAILURE line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
ACC Tests (26)	The number of times during the reporting interval that EDIF checked to determine whether a program was authorized to read a data set. This value is similar to the ACCESS CHECK TEST line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
ACC Fail (27)	The number of times during the reporting interval that EDIF detected read violations. This value is similar to the ACCESS CHECK FAILURE line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
Copy Exam (28)	The number of times during the reporting interval that EDIF checked a DCB used by the IEBCOPY utility to determine whether any action should be taken. This value is similar to the IEBCOPY TEST DCB line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.
Copy Exam (29)	The number of times during the reporting interval that EDIF determined that a DCB used by the IEBCOPY utility was a candidate for EDIF processing. This value is similar to the IEBCOPY PROCESS DCB line in the MIM4053 message generated by the DISPLAY EDIF STATISTICS command.

CA MII Enhanced Data Set Integrity Violation Report (DD)

```
(1)                                (2)                                                                                    (3)
2009/09/02 14:06   CA MII Enhanced Dataset  Integrity Benefits Report - ENQUEUE Conflicts                    PAGE      1
 (4) System:    SYSA             (5) Jobname:  MIMGR       (6) Release:   11.9         (7)Service Level:  0000
 (8) From:     2009/09/02 09:37  (9) Each:     RECORD
(10)To:        2009/09/02 09:41  (11)Shift:    00:00 24:00

      (12)         (13)     (14)                   (15)                       (16)    (17)    (18)
      Time       Job Name Program                Data Set Name                Volume Action User Exit
 ---------------- -------- -------- ------------------------------------------  ------ ------ --------
 2009/09/02 09:41 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                  TSO03A NONE


 2009/09/02 14:07   CA MII Enhanced Dataset Integrity Dataset Violation Report -  Authorized Utility Violations PAGE      1
System:    SYSA             Jobname:  MIMGR        Release:   11.9         Service Level:  0000
 From:      2009/09/02 09:37  Each:     RECORD
 To:        2009/09/02 09:41  Shift:    00:00 24:00
      Time       Job Name Program                Data Set Name                Volume Action User Exit
 ---------------- -------- -------- ------------------------------------------  ------ ------ --------
 2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBIN1                   TSO03A ABEND
 2009/09/02 09:39 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBIN1                   TSO03A ABEND


 2009/09/02 14:07   CA MII Enhanced Dataset Integrity Dataset Violation Report -  CheckExclusive Violations    PAGE      1
 System:    SYSA             Jobname:  MIMGR        Release:   11.9         Service Level:  0000
 From:      2009/09/02 09:37  Each:     RECORD
 To:        2009/09/02 09:41  Shift:    00:00 24:00

      Time       Job Name Program                Data Set Name                Volume Action User Exit
 ---------------- -------- -------- ------------------------------------------  ------ ------ --------
 2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                  TSO03A ABEND
 2009/09/02 09:39 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                  TSO03A ABEND

 2009/09/02 14:07   CA MII Enhanced Dataset Integrity Dastaset Violation Report - Read Access Violations       PAGE      1
 System:    SYSA             Jobname:  MIMGR        Release:   11.9         Service Level:  0000
 From:      2009/09/02 09:37  Each:     RECORD
 To:        2009/09/02 09:41  Shift:    00:00 24:00

      Time       Job Name Program                Data Set Name                Volume Action User Exit
 ---------------- -------- -------- ------------------------------------------  ------ ------ --------
 2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                  TSO03A ABEND
 2009/09/02 09:40 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                  TSO03A ABEND

 2009/09/02 14:07   CA MII Enhanced Dataset Integrity Dataset Violation Report - Attribute Violations          PAGE      1
 System:    SYSA             Jobname:  MIMGR        Release:   11.9         Service Level:  0000
 From:      2009/09/02 09:37  Each:     RECORD
 To:        2009/09/02 09:41  Shift:    00:00 24:00
                                                                                                    (19)   (20)   (21)  (22)
```

```
      Time       Job Name Program                Data Set Name                Volume Action User Exit  Atrib  Source DSCB  USER
---------------- -------- -------- ------------------------------------------ ------ ------ --------  -------  ------ -----  -----
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            BLKSIZE  JCL    03120  00019
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            LRECL    JCL    00080  00040
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            LRECL    JCL    00080  00040
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            RECFM    JCL    FB     F
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            DSORG    JCL    PO     PS
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            BLKSIZE  JCL    03120  00019
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            RECFM    JCL    FB     F
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            LRECL    JCL    00080  00040
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            DSORG    JCL    PO     PS
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            BLKSIZE  JCL    03120  00019
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            RECFM    JCL    FB     F
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            RECFM    JCL    FB     F
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            BLKSIZE  JCL    03120  00019
2009/09/02 09:37 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            LRECL    JCL    00080  00040
2009/09/02 09:38 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            BLKSIZE  JCL    03120  00019
2009/09/02 09:38 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            BLKSIZE  JCL    03120  00019
2009/09/02 09:38 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            RECFM    JCL    FB     F
2009/09/02 09:38 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            LRECL    JCL    00080  00040
2009/09/02 09:38 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            RECFM    JCL    36864  32768
2009/09/02 09:38 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            DSORG    JCL    PO     PS
2009/09/02 09:38 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            LRECL    JCL    00080  00040
2009/09/02 09:38 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            RECFM    JCL    FB     F
2009/09/02 09:38 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            DSORG    JCL    PO     PS
2009/09/02 09:39 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            BLKSIZE  JCL    03120  00019
2009/09/02 09:39 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            LRECL    JCL    00080  00040
2009/09/02 09:39 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            BLKSIZE  JCL    03120  00019
2009/09/02 09:39 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            LRECL    JCL    00080  00040
2009/09/02 09:39 OPSOSF   GTOPMVS  MIM.EDIFQA.EDITEST.DCBTEST                 TSO03A ABEND            RECFM    JCL    FB     F
```

Field	Description
(1)	Date and time the report was created.
(2)	Report title.
(3)	Report page number.
(4)	System name of the CA MII image as defined by the DEFSYS statement.
(5)	Name of the CA MII started task.
(6)	The CA MII release number.
(7)	The CA MII service level.
(8)	Date and timestamp of the earliest record in the reporting interval.
(9)	Summary level for the data line. In this example, each RECORD is displayed in the detail line. If each HOUR was specified, then the detail line would be a summary of the hour's activity.
(10)	Date and timestamp of the latest record in the recording interval.
(11)	Range of records included in the report.
Time (12)	Time that violation occurred.

Field	Description
Job Name (13)	Name of the job that created violation event.
Program (14)	Name of the program that was executing when the violation was detected.
Data Set Name (15)	Name of the data set involved in the violation.
Volume (16)	Volser upon which the data set resides.
Action (17)	Action that EDIF took as a result of the violation.
User Exit (18)	Up to eight bytes of information added to the SMF record by the installation EDIF exit, if coded to do so.
Attrib (19)	Attribute that caused the violation.
Source (20)	Source where failing attribute was detected.
DSCB (21)	Current data set DSCB value for the attribute in question. In the Attribute Violations report, the numbers for LRECL and BLKSIZE are displayed instead of DSCB information.
USER (22)	Value that the job specified for the attribute in question. In the Attribute Violations report, the numbers for LRECL and BLKSIZE are displayed instead of USER information.

CA MII QNAME Enqueue/Reserve Report (EC)

```
(1)                (2)                                                                                    (3)

2008/07/13 14:06   CA MII QNAME Enqueue/Reserve Report                                                    PAGE      1
(4) System:     SYSA              (5) Jobname:  MIMGR       (6) Release:   11.9         (7)Service Level:  0000
(8) From:       2003/10/25 00:00  (9) Each:     RECORD
(10)To:         2008/03/12 10:52  (11)Shift:    00:00 24:00
                  (13)     (14)     (15)      (16)      (17)       (18)      (19)       (20)      (21)    (22)   (23)
   (12)            Total     Average  Failures             Conflict     ENQs       ENQs     RESERVEs   RESERVES  Proc.  Proc.
  Interval Start   Requests  Serv Time Prevented   QNAME     Count     Received  Processed   Received  Converted   /Sec   /Cyc
 ---------------- ---------- --------- --------- --------- ---------- ---------- ---------- ---------- ---------- ------ ------
 2008/03/07 16:58      3,084   .007582       130 * OTHERS           0          0          0          0          0
                                                  $RXQUEUE          0          0          0          0          0
                                                  ACTLIB            0          0          0          0          0
                                                  ADRPRDCT          0          0          0          0          0
                                                  APTSYS            0          0          0          0          0
                                                  ARCBTAPE          0          0          0          0          0
                                                  ARCDSN            0          0          0          0          0
                                                  ARCENQG           8         20         20          0          0    .02    .01
                                                 *ARCGPA            0          0          0          5          0
                                                  ARCHBKLG          0          0          0          0          0
                                                  ARCLOG            0          0          0          0          0
                                                  ASM               0          0          0          0          0
                                                  ASM2.IXR          0          0          0          0          0
                                                  ASM2SYSA          0          0          0          0          0
                                                  ASM2SYSB          0          0          0          0          0
                                                  ASTEXMM           0          0          0          0          0
                                                  BUNDL             0          0          0          0          0
                                                  CA-ENF            0          0          0          0          0
                                                  CADB2             0          0          0          0          0
                                                  CADTCM01          0          0          0          0          0
                                                  CAIMSMF           0          0          0          0          0
                                                  CAJ$SCE1          0          0          0          0          0
```

Field	Description
(1)	Date and time the report was created.
(2)	Report title.
(3)	Report page number.
(4)	System name of the CA MII image as defined by the DEFSYS statement.
(5)	Name of the CA MII started task.
(6)	The CA MII release number.
(7)	The CA MII service level.
(8)	Date and timestamp of the earliest record in the reporting interval.
(9)	Summary level for the data line. In this example, each RECORD is displayed in the detail line. If each HOUR was specified, then the detail line would be a summary of the hour's activity.

Field	Description
(10)	Date and timestamp of the latest record in the recording interval.
(11)	Range of records included in the report.
Interval Start (12)	Start date and timestamp that the report line interval data covers.
Total Requests (13)	The total number of managed ENQ/RESERVE requests that CA MII intercepted and processed during the reporting interval. This number is similar to the REQUEST column of the MIM1021 message generated by the DISPLAY GDIF SERVICE command. Note that it is possible and quite probable that the number in this column differs from the sum of the numbers in the ENQ/PROC column and the RESV/CONV column. This is due to the fact that the managed ENQ/RESERVE count is incremented once for every occurrence of an SVC 56 that is intercepted that contains at least one managed QNAME in its parameter list. The ENQ/PROC count and the RESV/CONV count is incremented for every respective managed QNAME encountered in the SVC 56 parameter list. Since the SVC 56 parameter list supports the occurrence of multiple QNAME/RNAME combinations in a single invocation of SVC 56 (the SYSDSN ENQ done by that initiator is a good example of this), it is not expected that these columns will be equal.
Average Serv Time(14)	The average time, in seconds, that CA MII took during the reporting interval to perform global processing for a given ENQ or RESERVE request. This number is similar to the TIME/REQUEST column of the MIM1021 message generated by the DISPLAY GDIF SERVICE command.
Failures Prevented (15)	The number of integrity failures prevented, from the viewpoint of the system being reported on, due to outstanding activity on external systems. This count represents the number of times that a requester in the local system issued a request for a QNAME/RNAME combination that was already owned on a remote system, and would have caused a potential integrity exposure were it not for global resource integrity protection. This number is similar to the first value displayed in the MIM1015 message generated by the DISPLAY GDI BENEFITS command.
QNAME (16)	ENQ or RESERVE QNAME encountered by CA MII during the reporting cycle. If the QNAME is preceded by an asterisk (*), then CA MII does not manage the QNAME. This number is similar to the RESOURCE column in the MIM1016 message generated by the DISPLAY GDIF COUNTS command.

Field	Description
Conflict Count (17)	The number of global conflicts encountered for the QNAME, from the standpoint of the local system, which occurred during the reporting interval. This number is similar to the GLOBAL CONFLICTS column in the MIM1016 message generated by the DISPLAY GDIF COUNTS command.
ENQs Received (18)	The number of times the QNAME was encountered in an ENQ request that was intercepted by CA MII. This number is similar to the ENQS ISSUED column in the MIM1016 message generated by the DISPLAY GDIF COUNTS command. The column total on the total line reflects the number of managed and non-managed ENQ requests.
ENQs Processed (19)	The number of times the QNAME was processed as a managed QNAME in an ENQ request that was intercepted by CA MII. This number is similar to the ENQS PROCESSED column in the MIM1016 message generated by the DISPLAY GDIF COUNTS command.
RESERVEs Received (20)	The number of times the QNAME was encountered in a RESERVE request that was intercepted by CA MII. This number is similar to the RESERVES ISSUED column in the MIM1016 message generated by the DISPLAY GDIF COUNTS command. The column total on the total line reflects the number of managed and non-managed RESERVE requests.
RESERVEs Converted (21)	The number of times the QNAME was processed (converted to a global ENQ request) as a managed QNAME in a RESERVE request that was intercepted by CA MII. This number is similar to the RESERVES CONVERTED column in the MIM1016 message generated by the DISPLAY GDIF COUNTS command.
Proc./Sec (22)	The average number of managed requests (or requests for the named QNAME) that CA MII received in one second. This number is similar to the RATE/SECOND column of the MIM1021 message generated by the DISPLAY GDIF SERVICE command. Note that this value is for managed requests only; non-managed requests display as zeros.
Proc./Cyc (23)	The average number of managed requests (or requests for the named QNAME) that CA MII processed in one control file cycle. This number is similar to the RATE/CYCLE column of the MIM1021 message generated by the DISPLAY GDIF SERVICE command. Note that this value is for managed requests only; non-managed requests display as zeros.

GDPS/PPRC HyperSwap Support

GDPS, or Geographically Dispersed Parallel Sysplex, is a family of service offerings from IBM that manages disk and tape mirroring and works to speed the recovery of applications on the z/OS environment. There are several versions of GDPS available, each of which is based upon a DASD mirroring technology.

PPRC, or Peer to Peer remote copy, from IBM is one of these DASD mirroring technologies. PPRC provides synchronous data mirroring between two separate DASD subsystems, primary and secondary, that are connected by a fiber link. WRITES to logical volumes within the primary DASD subsystem are replicated synchronously on the secondary DASD subsystem. Synchronously, in this instance, means that the WRITE IO to the primary subsystem does not complete until the secondary subsystem notifies the primary subsystem that the replication is complete. This process is transparent to the applications and ensures that the data on the primary and secondary subsystems is always consistent. Given this, in the event of a disaster, it should be possible to switch from the primary to the secondary DASD subsystem and continue processing without needing to restore any data. PPRC, like other DASD mirroring technologies, provides commands to perform this switch.

However, although PPRC provides data consistency and provides commands to switch between DASD subsystems, a switch is still a disruptive process that may take hours to complete.

HyperSwap, also a service offering from IBM, is a set of automation scripts built upon GDPS/PPRC that aims to quickly swap a large number of PPRC volume pairs and do it without restarting systems.

HyperSwap requires that all hardware RESERVES directed against PPRC enabled volumes be converted to global ENQs. Because of this requirement, HyperSwap has code that attempts to validate that the shared DASD complex is configured to convert all RESERVES by default. With GDPS/PPRC 3.1 APAR AG31C38, HyperSwap code calls theVPCEXIT4 REXX EXEC, which validates that all necessary RESERVES are converted to global ENQs and then returns the answer to the HyperSwap code.

CA MII uses this interface to provide the support necessary to run GDPS/PPRC HyperSwap with CA MII. There are two parts to this support supplied with CA MIM:

- VPCEXIT4 - a sample REXX EXEC
- MIMHYPS() - an external REXX function that VPCEXIT4 invokes

The VPCEXIT4 Sample REXX EXEC

The HyperSwap code periodically calls the VPCEXIT4 REXX EXEC if it exists in the NetView CLIST concatenation. VPCEXIT4 returns one of two possible return codes:

0

All RESERVES for PPRCed volumes are converted to global ENQs; GDPS remains enabled for HyperSwap.

4

RESERVES for PPRCed volumes may not be converted to global ENQs; GDPS temporarily disables HyperSwap.

VPCEXIT4 calls the MIMHYPS() external REXX function and returns one of the above return codes to its caller based upon the result of the MIMHYPS() REXX function. The sample EXEC resides in member VPCEXIT4 of the CAI.CBTDSAMP data set.

You may use the sample VPCEXIT4 REXX EXEC provided or you can write your own VPCEXIT4 REXX EXEC. Either way, you must make sure that a REXX EXEC named VPCEXIT4 exists in the NetView CLIST concatenation if you wish to run HyperSwap with CA MII.

MIMHYPS() REXX Function

The MIMHYPS() REXX function returns one of the following text strings to VPCEXIT4:

- **OK**, if the following is true:
  - CA MII is active and operational.
  - CA MII is running with GDIINIT PROCESS=ALLSYSTEMS.
- **NO**, followed by a string describing why HyperSwap should be disabled

Note: The list of criteria checked by the MIMHYPS() function is not complete, and, by itself, does not guarantee that all RESERVES are converted to global ENQs. You can ensure that all appropriate RESERVES are converted correctly by following the steps in the next section.

Verify that CA MII Converts All Reserves

Verify that all RESERVES targeted at PPRCed DASD volumes are converted to global ENQs.

Follow these steps:

1. Verify CA MII is synchronized before all other address spaces that issue hardware reserves. Otherwise, some reserves are not converted. We recommend starting GDIF with the Early Start Mechanism which is detailed in the C*A MIM Installation Guide*.
2. CA MII cannot use DASD(3390) control files: CA MII uses a hardware reserve to serialize its control file, and it cannot convert its own MIMGLOBL reserve. We recommend using COMMUNICATION=CTCONLY or XCF or DASDONLY(with XES list structures). To verify your current communication method, issue the following command:

   ```
   DISPLAY MIM IO
   ```

3. Run GDIINIT PROCESS=ALLSYSTEMS and RESERVES=CONVERT. Issue the following command to determine your current operating values:

   ```
   DISPLAY GDIF INIT
   ```

4. Do not specify RESERVES=KEEP or GDIF=NO on any QNAME statement. Verify that the specification does not allow any hardware RESERVES to be issued against any PPRC-enabled DASD volumes. To review your current QNAME list, issue the following command:

   ```
   DISPLAY MIM QNAMES
   ```

5. Run CA MII with SETOPTION GDIF EXEMPTRESERVES=NO. Or verify that none of your exempt list definitions result in a hardware RESERVE being issued against a PPRC-enabled DASD volume. To review your setting for this option, issuing the following command:

   ```
   DISPLAY GDIF OPTIONS
   ```

   To review your exempt list, issue the following command:

   ```
   DISPLAY GDIF EXEMPT
   ```

Appendix A: CA Product EDIF Statements

The following are EDIF recommendations for various CA products. These statements should be coded in the EDIPARMS member in the CA MIM parameter library.

CA 1

We do not recommend that you have EDIF attribute-checking in effect for the CA 1 audit data set. For example:

```
DATASET NAME=ca1.audit OPTION=(NOATTR)
```

CA 11

ENQ deadlocks are possible when EDIF ENQUEUE processing is specified for both CMT and JEHF data sets used by CA 11. Because these data sets are properly serialized without EDIF assistance, specify appropriate DATASET statements to suppress EDIF ENQUEUE activity. For example:

```
DATASET NAME=cmt.dataset OPTION=(NOENQ,NOWAIT)
DATASET NAME=jehf.dataset OPTION=(NOENQ,NOWAIT)
```

CA ACF2

For r8 and above, the program ACF8AMA0 should be exempted from EDIF attribute checking for the CA ACF2 sequential backup data sets. For example:

```
DATASET NAME=acf2.backup.one OPTION=(ATTR,ABEND) EXEMPT=(PROGRAM=ACF8AMA0)
DATASET NAME=acf2.backup.two OPTION=(ATTR,ABEND) EXEMPT=(PROGRAM=ACF8AMA0)
DATASET NAME=acf2.backup.three OPTION=(ATTR.ABEND) EXEMPT=(PROGRAM=ACF8AMA0)
```

CA Intertest

EDIF ENQ processing should not be in effect for the “profile” data sets. For example:

```
DATASET NAME=intertest.profile.dataset OPTION=(NOENQ)
```

CA PDSMAN

ENQ deadlocks involving QNAMEs EDIDSN, SPFEDIT, and SYSIEWLP are possible with the CA PDSMAN product. The deadlock occurs when CA PDSMAN tries to update the directory information as the result of the REF option on the $ACCESS statement. You can avoid this situation by specifying REF=N in all $ACCESS statements (CA PDSMAN), or by assuring that OPTION=NOENQ (EDIF) is in effect for all data sets managed by CA PDSMAN.

We have worked closely with the CA PDSMAN developers to produce the following solution:

Current releases of CA PDSMAN include an EDI=Y option on the $MISC statement to activate special code in CA PDSMAN that permits proper handling by both CA PDSMAN and EDIF without the possibility of a deadlock.

CA Jobtrac

We do not recommend that you have EDIF ENQ protection in effect for the CA Jobtrac checkpoint data set. For example:

```
DATASET NAME=checkpoint.dataset OPTION=(NOENQ)
```

CAIENF CAS9DB

EDIF ENQ processing should not be in effect for the ENFDB data sets because lockouts may occur when running the data base rebuild utility (CAS9DB). To turn EDIF ENQ processing off for ENFDB data sets, code the following in the EDIPARMS member:

```
DATASET,NAME=CAIP.ENFDB,OPTION=NOENQ
```

Appendix B: How You Convert RNL Specifications to CA MII Statements

IBM and some ISV products specify their cross-system integrity requirements in the form of Resource Name List (RNL) specifications. This chapter helps you understand which CA MII statements are similar to the RNL statements used by the IBM GRS.

This section contains the following topics:

Map RNL Specifications to CA MII QNAME Statements

In GRS, resource selection is accomplished with the use of three RNLs:

- SYSTEM inclusion RNL. This list is used to promote ENQs issued with a scope of SYSTEM to a scope of SYSTEMS; that is, force propagation.
- SYSTEMS exclusion RNL. This list is used to demote ENQs (or RESERVEs) issued with a scope of SYSTEMS to a scope of SYSTEM, that is, do not propagate.
- RESERVE conversion RNL. This list is used to suppress hardware RESERVE requests and allow the scope of SYSTEMS to remain.

In GRS RNL entries, the resources can be specific or generic.

- Specific means that the QNAME and RNAME of the request must exactly match the RNL specification or the RNL will not take effect.
- Generic means that the RNL takes effect if the QNAME and RNAME of the request match the RNL specification up to the length specified. In other words, generic entries are resource prefixes.

In CA MII, all resource selection is accomplished using statements specified in the MIMQNAME list. Exceptions, if required, are specified in the GDIEXMPT list.

Examples:

1. The following GRS RNL specifies that all ENQ requests issued with QNAME SYSDSN and scope SYSTEM will be promoted to SYSTEMS, and therefore will be propagated:

   ```
   RNLDEF RNL (INC) TYPE (GENERIC) QNAME (SYSDSN)
   ```

   The following CA MII QNAME statement produces similar results:

   ```
   QNAME SYSDSN SCOPE=SYSTEM GDIF=YES
   ```

2. The following GRS RNL specifies that ENQ requests issued with QNAME SYSDSN and RNAME 'SYS1.LOGREC' and whose scopes are (or are INCLuded, or promoted to) SYSTEMS will be downgraded to SYSTEM and will not be propagated:

   ```
   RNLDEF RNL (EXCL) TYPE (SPECIFIC) QNAME (SYSDSN) RNAME (SYS1.LOGREC)
   ```

   The following CA MII EXEMPT statement produces similar results:

   ```
   LOCAL QNAME=SYSDSN RNAME=SYS1.LOGREC
   ```

A Comprehensive Example

Note: This example is intended solely for the purpose of illustrating how you can convert RNL specifications to CA MII QNAME statements. It is not intended to be a recommendation for implementing DFSMS serialization in CA MII. As with any product, contact the vendor of that product to obtain the most current QNAME or RNL recommendations.

In the *z/OS V1R12.0 DFSMShsm Implementation and* Customization guide, IBM gives an example of the RNL definitions required when running DFSMShsm with CDSR=YES and DFHSMDATASETSERIALIZATION options, and when the RESERVE for the JOURNAL file is not converted. The RNL definitions for this configuration are as follows:

```
RNLDEF RNL (EXCL) TYPE (GENERIC) QNAME (ARCAUDIT)
RNLDEF RNL (EXCL) TYPE (GENERIC) QNAME (ARCBACK)
RNLDEF RNL (EXCL) TYPE (GENERIC) QNAME (ARCGPA)
RNLDEF RNL (EXCL) TYPE (GENERIC) QNAME (ARCGPAL)
RNLDEF RNL (EXCL) TYPE (GENERIC) QNAME (ARCUPDT)
RNLDEF RNL (CON)  TYPE (GENERIC) QNAME (ARCBACV)
RNLDEF RNL (CON)  TYPE (GENERIC) QNAME (ARCMIGV)
RNLDEF RNL (INC)  TYPE (GENERIC) QNAME (SYSDSN)
RNLDEF RNL (EXCL) TYPE (GENERIC) QNAME (SYSVTOC)
```

In CA MII, code the following statements in your MIMQNAME list:

```
ARCAUDIT GDIF=NO  SCOPE=RESERVES EXEMPT=NO  RESERVES=KEEP
ARCBACK  GDIF=NO  SCOPE=RESERVES EXEMPT=NO  RESERVES=KEEP
ARCGPA   GDIF=NO  SCOPE=RESERVES EXEMPT=NO  RESERVES=KEEP
ARCGPAL  GDIF=NO  SCOPE=RESERVES EXEMPT=NO  RESERVES=KEEP
ARCUPDT  GDIF=NO  SCOPE=RESERVES EXEMPT=NO  RESERVES=KEEP
ARCBACV  GDIF=YES SCOPE=RESERVES EXEMPT=NO  RESERVES=CONVERT
ARCMIGV  GDIF=YES SCOPE=RESERVES EXEMPT=NO  RESERVES=CONVERT
ARCENQG  GDIF=YES SCOPE=SYSTEMS  EXEMPT=NO
SYSDSN   GDIF=YES SCOPE=SYSTEM   EXEMPT=NO
SYSVTOC  GDIF=NO  SCOPE=RESERVES EXEMPT=NO  RESERVES=KEEP
```

Note that there is no RNL entry for the ARCENQG QNAME, but there is an ARCENQG QNAME statement in the CA MII QNAME list. The reason for this is that the ARCENQG resource is always issued as an ENQ with a scope of SYSTEMS. In GRS, if there is no RNL entry to change the scope, then the ENQ is always propagated. In CA MII, if you are running with GDIINIT PROCESS=SELECT, then the ARCENQG QNAME statement is required for those requests to be propagated. If you are running PROCESS=ALLSYSTEMS, then the entry is not required. In either case, coding the above entry causes the ENQ to be propagated, which is the desired result.

Similarly, the CA MII QNAME entries for ARCAUDIT, ARCBACK, ARCGPA, ARCGPAL, and ARCUPDT are not required if you were running CA MII with GDIINIT PROCESS=SELECT. PROCESS=SELECT indicates that you want to propagate/convert only those resources that are explicitly specified in MIMQNAME. If you are running PROCESS=ALLSYSTEMS with GDIINIT RESERVES=CONVERT, then the above entries are required to *not* convert the reserves and propagate global ENQs. Again, in either case, coding the above entries causes the RESERVES *not* to be converted, which is the desired result.

In addition, note that the entries for SYSDSN and SYSVTOC in the RNL definitions were not explicitly given as RNL specifications in the IBM guide. However, the given DFSMS RNL specifications implied the co-requisite handling of SYSDSN and SYSVTOC in verbal notes. Therefore, we added the GRS RNL specifications and the corresponding CA MII QNAME statements in this example.

Note: For a complete understanding of how CA MII processes an ENQ request, see the CA MII flowchart in the chapter "Advanced Topics (see page 27)".

CA MII Parameters that Affect QNAME and EXEMPT Processing

CA MII has two processing modes that you can specify on the GDIINIT PROCESS statement in your MIMINIT parameters:

- PROCESS=SELECT specifies that only the QNAMES explicitly coded in MIMQNAME with GDIF=YES will be propagated to other systems.
- PROCESS=ALLSYSTEMS specifies that all ENQ requests issued with SCOPE=SYSTEMS will be propagated to other systems, unless otherwise directed in MIMQNAME/GDIEXMPT. PROCESS=ALLSYSTEMS operates similarly to GRS, in that no entries are required for ENQs to be propagated, if they are issued with a scope of SYSTEMS. The CA MII processing mode has an effect on what entries you need to code in your MIMQNAME list-it does not change the way individual statements behave.

The DEFAULT statement in the GDIEXMPT member has an effect on EXEMPT processing and what statements need to be coded in the EXEMPT member. For example, DEFAULT RNAME=GLOBAL JOB=GLOBAL coded in your exempt list causes ENQ requests for QNAMES that have EXEMPT=YES coded on the QNAME statement to be treated as global resources unless otherwise directed by LOCAL statements.

The GDIINIT RESERVES setting sets the default for whether RESERVES are converted for a QNAME that is managed by GDIF. This value can be overridden on a QNAME statement, a GLOBAL statement, or both.

The GDIINIT TEMPORARYDSN parameter, if set to NO, causes ENQ requests for temporary data set names to not be propagated. You can think of this as implied LOCAL statements for SYSDSN requests that fit the format of a temporary data set name.

The SETOPTION GDIF EXEMPTRESERVES parameter indicates whether LOCAL statements should apply to RESERVE requests. The default is NO, which indicates that LOCAL statements are ignored. If the request is a RESERVE request, then the RESERVE is converted and the ENQ is propagated. If YES is specified, then LOCAL statements cause RESERVEs to be kept (not converted) for those resources.

Index

U

X