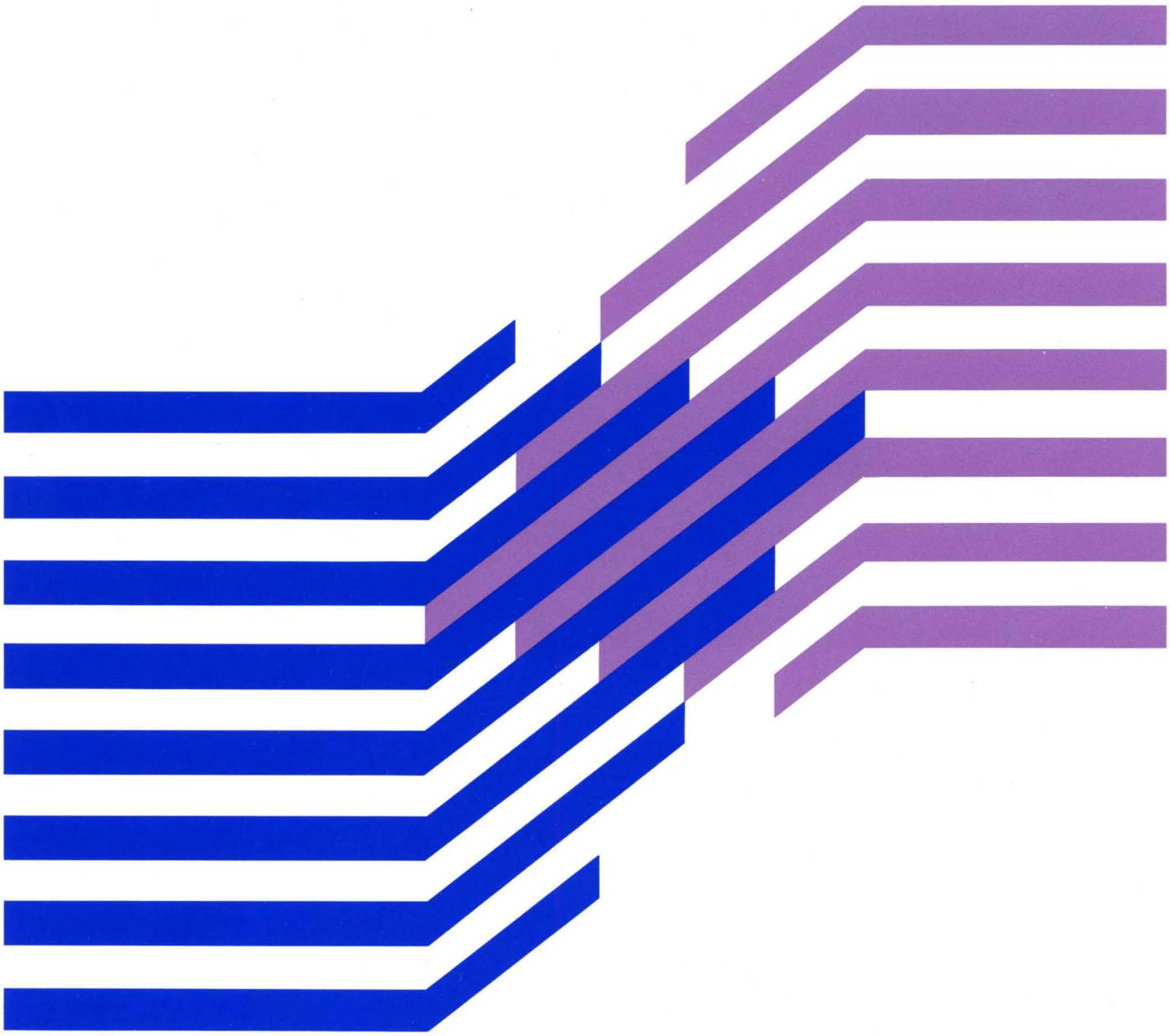
GC28-1818-2

MVS/ESA Planning: Global Resource Serialization

MVS/System Product:
JES2 Version 3
JES3 Version 3



GC28-1818-2

MVS/ESA
Planning: Global Resource Serialization

MVS/System Product:
JES2 Version 3
JES3 Version 3

Third Edition (June, 1991)

This is a major revision of, and obsoletes, GC28-1818-1. See the Summary of Changes for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Version 3 Release 1.3 of MVS/System Product 5685-001 or 5685-002 and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

PROGRAMMING INTERFACES

This book does NOT contain any programming interface information.

About This Book

This book describes how to plan for a global resource serialization complex. A global resource serialization complex, which consists of two or more systems connected by communication links, allows users on multiple systems to serialize access to resources, particularly data sets on shared DASD volumes.

Trademarks

The following are trademarks of International Business Machines Corporation.

- IBM™
- MVS/ESA™
- MVS/DFP™
- MVS/SP™
- MVS/XA™
- PR/SM™
- System/370™
- VM/XA™

Who Should Use This Book

This book is for the people at an installation who must decide whether or not to use global resource serialization to serialize access to data sets on shared DASD volumes and who, once that decision is made, must plan how to make the best use of global resource serialization.

This book applies to MVS/ESA systems (those systems that run MVS/SP Version 3 with MVS/DFP Version 3) but also includes, where necessary, information about earlier versions of MVS, such as MVS/XA. This book uses "MVS" to apply to all versions of MVS. When it must make distinctions, it uses "MVS/ESA" or "MVS/XA" or identifies specific releases. With minor differences, global resource serialization works in the same way on all versions of MVS.

How This Book Is Organized

This book assumes that you are very familiar with MVS and have a thorough knowledge of your installation's data processing goals, problems, and available hardware and software.

The book deals solely with the process of planning — the thinking and decision-making someone at your installation must do before you actually use global resource serialization to serialize access to global resources, such as data sets on shared DASD volumes. It does not describe such details as the complete syntax of the macros, parmlib members, and system commands that you use to actually carry out your plan. These details are described in other books. These required books are listed both in this preface and in the chapters of this book that describe the planning tasks.

The book has five chapters:

"Chapter 1: Introduction" presents an overview of what global resource serialization can do for you. It describes what a global resource serialization complex is and how it works. The information in Chapter 1 can help you to decide whether a global resource serialization complex can solve resource serialization problems that your installation might have.

"Chapter 2: Selecting the Data" describes how global resource serialization processes requests for resources. The information in Chapter 2 can help you to decide how to handle various resources at your particular installation.

"Chapter 3: Designing the Complex" describes the basic principles of designing a global resource serialization complex. The information in Chapter 3 can help you to design a complex that will meet your needs.

"Chapter 4: Operating the Complex" describes how to build and operate a global resource serialization complex. It presents extensive information on the procedures that operators need to run the complex. The information in Chapter 4 can help you to plan operations procedures that will work for your installation.

"Chapter 5: Installing and Tuning the Complex" describes one step-by-step approach to installing the global resource serialization complex. It identifies installation steps as well as planning tasks and long-term considerations. It also describes the storage requirements for global resource serialization and some actions you can take to tune the performance of the complex.

"Appendix: The RNL Syntax Checker" describes how to use the resource name list (RNL) syntax checker to verify the syntax of RNLDEF statements in SYS1.PARMLIB GRSRNL members. It also lists the messages that the RNL syntax checker issues when processing members.

How To Use This Book

Once you have decided to build a global resource serialization complex, there are three basic planning tasks:

- Select the data to be shared
- Design the complex
- Provide effective operator procedures

All of these tasks are closely related, and the design and operation of a global resource serialization complex are especially closely related. Therefore, a good way to use this book is to read it quickly all the way through to become familiar with all of the planning you need to do for a global resource serialization complex. Then, use the information and the planning aids in the book to develop your installation's plan for the use of global resource serialization.

Related Products

This book refers to the following licensed programs:

DFHSM: Data Facility Hierarchical Storage Manager, Version 2 (5665-329)

DFP (Data Facility Product): MVS/XA DFP Version 2 (5665-XA2) or MVS/DFP Version 3 (5665-XA3)

ISPF: Interactive System Productivity Facility (5665-319)

ISPF/PDF: Interactive System Productivity Facility/Program Development Facility (5665-317)

RACF: Resource Access Control Facility (Program Number 5740-XXH)

RMF: Resource Measurement Facility Version 4 (5685-029)

TSO/E: TSO Extensions Version 1 Release 4 (5665-285) or Version 2 (5685-025)

Related Information

To use this book effectively, you need to know about the information in the following books; they provide detailed information about how to implement your plan for a global resource serialization complex:

Short Title Used in This Book	**Title**	**Order Number**
Initialization and Tuning	MVS/ESA System Programming Library: Initialization and Tuning	GC28-1828
System Commands	MVS/ESA Operations: System Commands	GC28-1826
System Messages	MVS/ESA Message Library: System Messages, Volumes 1 - 2	GC28-1812 and GC28-1813

Related Books

While you are using this book, you might need the information in the following books; they contain information related to the process of planning for a global resource serialization complex:

Short Title Used in This Book	**Title**	**Order Number**
Application Development Guide	MVS/ESA Application Development Guide	GC28-1821
Application Development Macro Reference	MVS/ESA Application Development Macro Reference	GC28-1822
SPL: Application Development Guide	MVS/ESA System Programming Library: Application Development Guide	GC28-1852
SPL: Application Development Macro Reference	MVS/ESA System Programming Library: Application Development Macro Reference	GC28-1857
System Codes	MVS/ESA Message Library: System Codes	GC28-1815
Installation Exits	MVS/ESA System Programming Library: Installation Exits	GC28-1836
System Modifications	MVS/ESA System Programming Library: System Modifications	GC28-1831
Service Aids	MVS/ESA Service Aids	GC28-1844
Planning: Dump and Trace Services	MVS/ESA Planning: Dump and Trace Services	GC28-1838
Initialization and Tuning	MVS/ESA Initialization and Tuning	GC28-1828
RMF User's Guides	MVS/ESA Resource Measurement Facility (RMF) Version 4 Reference and User's Guide	LY28-1007 and LY28-1008
Data Administration: Macro Instruction Reference	MVS/ESA Data Administration: Macro Instruction Reference	SC26-4506

Other books that you might find useful are the books that describe the hardware used in your complex and the books that describe your job entry subsystem.

Contents

Figures

Summary of Changes

Summary of Changes
for GC28-1818-2
MVS/System Product Version 3 Release 1.3

This revision consists mainly of editorial changes.

The section describing "RNLs in LINKLIB" has been moved into an appendix.

Summary of Changes
for GC28-1818-1
MVS/System Product Version 3 Release 1.3

This major revision includes the following changes.

New Information

An appendix describes the RNL utility checker. The RNL utility checker verifies the syntax of GRSRNL members of SYS1.PARMLIB.

Changed Information

This book contains updates to system resources that are candidates for the resource name lists.

Summary of Changes
for GC28-1818-0
MVS/System Product Version 3 Release 1.0E

This book contains information previously presented in *OS/VS2 MVS Planning: Global Resource Serialization*, GC28-1062-1. The following summarizes the changes to that information.

New Information

This book includes information about using the IBM 3088 Multisystem Channel Control Unit (MCCU) to connect systems in a global resource serialization complex and describes using GRSRNLxx to define the resource name lists. It also provides information about using ring acceleration and various recovery/performance enhancements.

Moved Information

Detailed information about using the GRS RNL exit search routine has been moved to *MVS/ESA SPL: User Exits*, GC28-1836.

Deleted Information

This book no longer contains detailed descriptions of using integrated channel-to-channel (CTC) adapters to connect the systems in a global resource serialization complex.

Chapter 1. Introduction

At many data processing installations, users must share access to data. Sharing data, however, requires a way to control access to that data. Users who are updating data, for example, need exclusive access to that data; if several users tried to update the same data at the same time, the result would be a data integrity exposure (the possibility of incorrect or damaged data). In contrast, users who only read data can safely access the same data at the same time.

MVS enables users to protect the integrity of data, or other types of resources, by allowing users to serialize access to those resources. Users can serialize access to resources at three levels:

1. Within an address space
2. Within a single MVS system
3. Across multiple MVS systems

To serialize access to data within an address space, programs use the ENQ macro to obtain access to a resource and the DEQ macro to free the resource. A scope of STEP indicates that MVS is to protect the resource only within the address space of the job issuing the macro.

To serialize access to data within a single system, programs use the ENQ macro to obtain access to the resource and the DEQ macro to free the resource. A scope of SYSTEM indicates that MVS is to protect the resource within the system.

To serialize access to data across multiple systems, programs can use two methods:

1. Issue the RESERVE macro to obtain access to a resource and the DEQ macro to free the resource. A scope of SYSTEMS is required with RESERVE. This method serializes access to data sets that reside on shared DASD volumes.

2. Issue the ENQ macro to obtain access to the resource and the DEQ macro to free the resource, specifying a scope of SYSTEMS. A scope of SYSTEMS indicates that MVS is to protect the resource across multiple systems.

Using the first method, the RESERVE macro, protects only resources that reside on shared DASD. While it protects those resources, it does have drawbacks. See "RESERVE Macro" on page 1-2.

Using the second method, the ENQ macro with a scope of SYSTEMS, requires a multisystem approach to resource serialization. Your installation combines the systems that must serialize access to resources in a global resource serialization complex. See "Global Resource Serialization" on page 1-2.

With a global resource serialization complex, your installation can override RESERVE macros without changing existing programs. This technique overcomes the drawbacks of the RESERVE macro and can provide a path for installation growth by increasing the availability of the systems and the computing power available for applications.

Another macro, GQSCAN, enables users to obtain information about the status of each resource, including information about the tasks that have requested the resource. Information about using ENQ and DEQ appears in *Application Development Macro Reference*. Additional information about ENQ and DEQ, as well as descriptions of GQSCAN and RESERVE, appears in *SPL: Application Development Macro Reference*.

RESERVE Macro

The RESERVE macro serializes access to a resource (a data set on a shared DASD volume) by obtaining control of the volume on which the resource resides to prevent jobs on other systems from using any data set on the entire volume.

Serializing access to data sets on shared DASD volumes by means of a reserve generally protects the resource. However, it creates several critical problems:

- When a task on one system has issued a RESERVE macro to obtain control of a data set, no programs on other systems can access any other data set on that volume. Protecting the integrity of the resource means delay of jobs that need the volume but do not need that specific resource, as well as delay of jobs that need only read access to that specific resource. These jobs must wait until the system that issued the reserve releases the device.

  Because the reserve ties up an entire volume, it greatly increases the chance of an interlock (also called a deadlock) occurring between two tasks on different systems, as well as the chance that the interlock will expand to affect other tasks. (An *interlock* is unresolved contention for use of a resource.)
- A single system can monopolize a shared device when it encounters multiple reserves for the same device because the system does not release the device until it processes all of the reserves. No other system can use the device until the reserving system releases the device.
- A system reset while a reserve exists terminates the reserve. The loss of the reserve can leave the resource in a damaged state if, for example, a program had only partly updated the resource when the reserve ended. This type of potential damage to a resource is a data integrity exposure.

Because of these problems, many installations use job scheduling instead of a reserve to protect the integrity of data sets on shared DASD volumes. That is, jobs that need the same data set are assigned to the same job class and run at different times. Protecting resources by job scheduling works, but it complicates operations, might increase job turnaround time, and might reduce the additional workload your installation can handle.

Global Resource Serialization

Combining the systems that access the shared DASD volumes into a global resource serialization complex is another way to serialize access to resources across multiple systems, and it can solve the problems related to using the RESERVE macro.

A global resource serialization complex consists of two or more systems connected by dedicated communication links. The systems in the complex use global resource serialization to serialize access to data sets on shared DASD volumes. Global resource serialization uses the links to communicate information about global resources from one system in the complex to another. Each link can be either a data link in an IBM 3088 Multisystem Channel Communication Unit (MCCU) or an integrated channel-to-channel (CTC) adapter. In this book, "link" or "CTC link" means either an IBM 3088 MCCU data link or an individual CTC adapter.

Figure 1-1 shows the basic elements of a four-system global resource serialization complex.

Note: In your hardware configuration, all the links shown would physically run through the IBM 3088. Because ring processing is a logical concept, diagrams throughout this book, for simplicity and clarity, show the links as direct connections between systems.

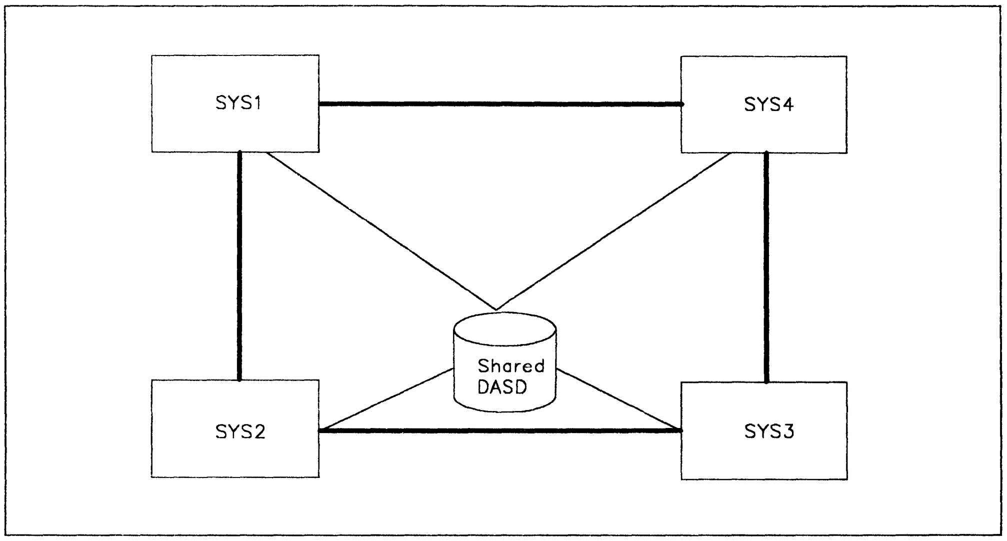


Figure 1-1. Global Resource Serialization Complex

In a global resource serialization complex, programs can serialize access to data sets on shared DASD volumes at the data set level rather than at the DASD volume level. To serialize access to data on shared DASD, programs use the ENQ macro to obtain access to the resource and the DEQ macro to free the resource, specifying a scope of SYSTEMS. SYSTEMS indicates that global resource serialization is to protect the integrity of the resource across multiple systems. A program on one system can access one data set on a shared volume while other programs on any system can access other data sets on the same volume.

A global resource serialization complex thus overcomes the major drawbacks of using the RESERVE macro. Because it enables jobs to serialize their use of resources at the data set level, it can reduce contention for these resources and minimize the chance of an interlock occurring between systems. It thus ends the need to protect resources by job scheduling. Because global resource serialization maintains information about global resources in system storage, it does away with the data integrity exposure that occurs when there is a system reset while a reserve exists.

How a Global Resource Serialization Complex Works

Understanding how a global resource serialization complex works is basic both to deciding whether to build a complex and deciding how you want the complex to process requests for specific resources.

This book describes how global resource serialization works in an MVS/ESA system (a system that includes MVS/SP Version 3). A mixed complex, a global resource serialization complex that includes MVS/ESA and MVS/XA systems, for example, is also possible. In general, the differences are minor and noted where they apply.

Local and Global Resources

The ENQ, DEQ, and RESERVE macros identify a resource by its symbolic name. The symbolic name has three parts: major name (qname), minor name (rname), and scope (which can be STEP, SYSTEM, or SYSTEMS).

Global resource serialization identifies a resource by its entire symbolic name. It recognizes a resource specified as A,B,SYSTEMS as a different resource from A,B,SYSTEM or A,B,STEP. It uses the scope on the ENQ, DEQ, or RESERVE macro to determine whether a resource is a local resource or a global resource.

A **local resource** is a resource requested with a scope of STEP or SYSTEM. It is serialized only within the system processing the request for the resource. If a system is not part of a global resource serialization complex, all resources, regardless of scope (STEP, SYSTEM, SYSTEMS), are local resources.

A **global resource** is a resource requested with a scope of SYSTEMS. It is serialized among all systems in the complex.

In general, global resource serialization identifies a resource with a scope of STEP or SYSTEM as a local resource, and a resource with a scope of SYSTEMS as a global resource. Because users of previous versions of MVS could serialize access to resources across multiple systems only through a reserve, whether the user specified SYSTEM or SYSTEMS on the ENQ macro did not affect resource serialization. Thus, your installation might have programs that specify ENQ with a scope of SYSTEM for resources that you would want global resource serialization to identify as global resources. You might also have programs that specify ENQ with a scope of SYSTEMS for resources that you would want global resource serialization to identify as local resources.

To ensure that resources are treated as you want them to be without changes to your applications, global resource serialization provides three resource name lists (RNLs):

- The **SYSTEM inclusion RNL** lists resources requested with a scope of SYSTEM that you want global resource serialization to treat as global resources.
- The **SYSTEMS exclusion RNL** lists resources requested with a scope of SYSTEMS that you want global resource serialization to treat as local resources.
- The **RESERVE conversion RNL** lists resources requested on RESERVE macro for which you want global resource serialization to suppress the reserve.

By placing the name of a resource in the appropriate RNL, you can cause global resource serialization to process it as you want. The RNLs enable you to build a global resource serialization complex without first having to change your existing programs, though you might at some time want to change these programs.

Deciding how to use the RNLs to define your resource processing needs is a major part of the planning for a global resource serialization complex; the process is described in detail in Chapter 2, "Selecting the Data" on page 2-1. To make most effective use of the RNLs, you also need to understand how a global resource serialization complex processes requests for global resources.

Ring Processing

As stated earlier, a global resource serialization complex consists of two or more systems connected by communication links. Global resource serialization uses the links to pass information about requests for global resources from one system in the complex to another.

Regardless of the physical configuration of systems and links, the global resource serialization complex consists of every system that indicates at IPL time that it is to be part of the complex. For various reasons, such as a system or link failure, not all of the systems in the complex might be actively using global resource serialization at any particular time. Those systems that are actively using global resource serialization to serialize access to global resources make up the **global resource serialization ring**.

Figure 1-2 shows a four-system global resource serialization complex. When all four systems in the complex are actively using global resource serialization, the complex and the ring are the same.

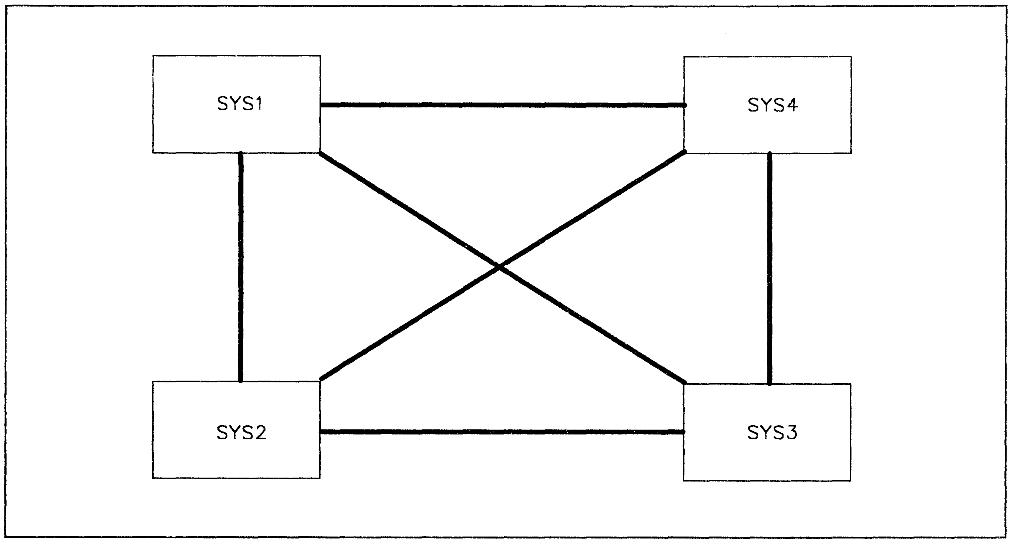


Figure 1-2. Fully-Connected Four-System Complex

The complex shown in Figure 1-2 has a CTC link between each system and every other system; such a complex is a **fully-connected complex**. A fully-connected complex is recommended because it allows the systems to build the ring in any order and allows any system to withdraw from the ring without affecting the other systems. It also offers more options for recovery if a failure disrupts ring processing.

If system SYS1 were to fail and end its active participation in serializing access to global resources, it would still be part of the complex, but it would not be part of the ring. Figure 1-3 shows the ring that would continue processing after system SYS1 stopped serializing access to global resources.

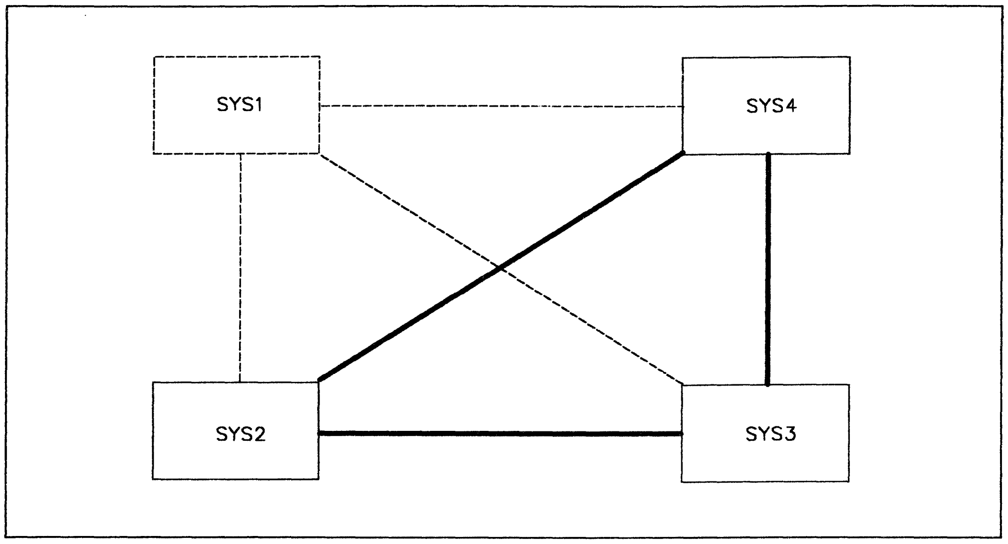


Figure 1-3. Three-System Ring — SYS1 Failed

Obviously, selecting the systems that are to be part of the complex and defining the links are important steps in the planning for a global resource serialization complex. Chapter 3, "Designing the Complex" on page 3-1 describes the process in detail. Chapter 4, "Operating the Complex" on page 4-1 describes how to plan operational procedures for the complex you design.

The concept of the global resource serialization ring is also important because, regardless of the physical configuration of systems and links that make up the complex, global resource serialization uses a ring processing protocol to communicate information from one system to another. Once the ring is active, the primary means of communication is the ring system authority message (RSA-message).

The RSA-Message

The RSA-message contains information about requests for global resources (as well as control information). It passes from one system in the ring to another. No system can grant a request for a global resource until other systems in the ring know about the request; your installation, however, can control how many systems must know about the request before a system can grant access to a resource. (See "Processing a Request for a Resource" on page 1-7.) The RSA-message contains the information each system needs to protect the integrity of resources; different systems cannot grant exclusive access to the same resource to different requestors at the same time.

When a system receives a request for a global resource, the system suspends the requestor and, when the RSA message arrives, places the request in the RSA-message. Systems in the global resource serialization ring batch their requests for global resources. For example, a system might receive seven requests for global resources while waiting for the incoming RSA-message — the message it receives from the preceding system. It adds all seven requests to the outgoing RSA-message — the message that it sends on to the next system in the ring.

Batching requests for resources minimizes the communication overhead for global resource serialization.

The order and direction of the RSA-message depend on the order in which systems join the ring and can change when systems enter or leave the ring. The amount of time that each system holds the RSA-message is called the **residency time**. Your installation sets the length of the residency time. While it holds the RSA-message, each system processes the requests in the incoming RSA-message and adds its new requests to the outgoing RSA-message.

The RSA-message is also a key factor in recovery from a failure that disrupts ring processing. Each RSA-message has a unique sequence number, and each system in the ring keeps a copy of the outgoing RSA-message. If a failure occurs, the systems in the ring use the sequence number to requeue any outstanding requests and synchronize their resource control information to resume processing correctly. Because global resource serialization can keep track of the RSA-message, it can recover from failures that disrupt the ring. It can resume processing requests for global resources without data integrity problems.

Processing a Request for a Resource

Global resource serialization processes requests for local resources and requests for global resources differently. When a task running on a system in the ring requests a local resource, that system handles the request on its own. Global resource serialization does not place a local resource request in the RSA-message.

When a task running on a system in the ring requests a global resource, however, the processing is very different because each system in the ring must keep track of all requests for global resources. How the ring processes a request for a global resource depends on whether or not your installation uses ring acceleration.

Request Processing without Ring Acceleration

Figure 1-4 on page 1-8 summarizes the processing of a global resource request without ring acceleration. The steps include:

1 The originating system suspends the requesting task — places it in a wait state.

2 When the incoming RSA-message arrives, the system places the request in the outgoing RSA-message, then passes the RSA-message on to the next system in the ring.

3 The request in the RSA-message makes a cycle around the ring. Each system in the ring records the request.

4 When the RSA-message completes its cycle, the originating system recognizes that all systems know about the request. The originating system then actually processes the request; it grants access to the resource according to normal ENQ processing. That is, if the resource is available, the system grants the suspended task access to the resource and marks the task as ready to execute. If the resource is not available, the task continues to wait until it becomes available.

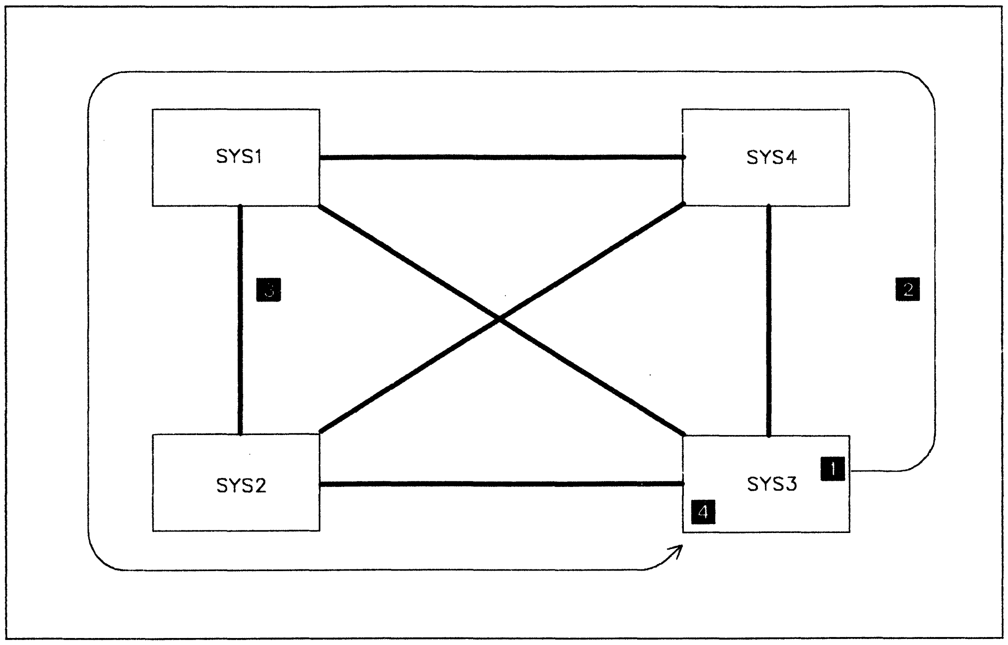


Figure 1-4. Global Resource Request Processing without Ring Acceleration

This processing ensures that each system in the ring has the same information about global resources at any particular time. The fact that each system has the same information ensures recovery from a failure that disrupts ring processing. Global resource serialization can maintain data integrity. Each system has accurate information about the state of all global resources granted at the time it last sent the RSA-message on to the next system in the ring. Thus, each system has the information it needs to:

1. Prevent different systems from allocating the same resource to different requestors
2. Allow the ring to continue processing without data integrity exposures even after multiple CTC link or system failures

This processing, however, means that every task that requests access to a global resource is suspended for at least the time it requires for the RSA-message to complete its cycle around the ring. For installations that cannot tolerate this delay, global resource serialization provides an alternative ring processing technique called ring acceleration.

Request Processing with Ring Acceleration

Ring acceleration is available only when all systems in the ring are MVS/ESA systems. It requires alternate links. Also, IBM recommends that the complex be a fully-connected complex.

Ring acceleration can significantly reduce the amount of time tasks spend waiting for global resources. It can, however, affect recovery. "Ring Acceleration (ACCELSYS)" on page 3-12 describes the factors involved in determining how your installation should use ring acceleration.

Using ring acceleration changes the processing of a global resource request. Figure 1-5 summarizes the processing of a global resource request with ring acceleration. The steps include:

1 The originating system suspends the requesting task — places it in a wait state.

2 When the incoming RSA-message arrives, the system places the request in the outgoing RSA-message, then passes the RSA-message on to the next system in the ring.

3 Your installation chooses the number of systems that must see the RSA-message before a system sends the shoulder-tap to the originating system. Assuming that the number of systems is two, the next system in the ring uses the alternate link to the originating system to send a "shoulder tap," an acknowledgement that it has received the RSA-message.

4 When the originating system receives the shoulder-tap, the ring acceleration signal, it grants access to the resource according to normal ENQ processing. That is, if the resource is available, the system grants the suspended task access to the resource and marks the task as ready to execute. If the resource is not available, the task continues to wait until it becomes available.

5 The RSA-message continues on its cycle around the ring so that each system in the ring knows about the request. The task that requested the resource, however, does not need to wait for the cycle to complete before obtaining access to the resource.

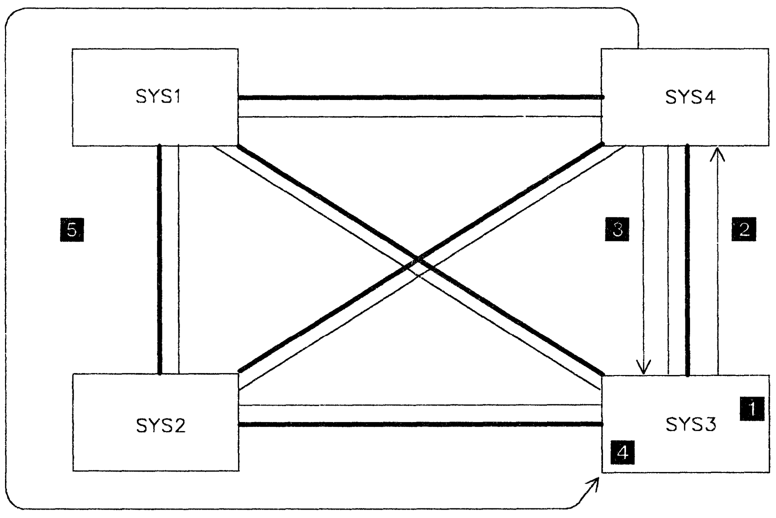


Figure 1-5. Global Resource Request Processing with Ring Acceleration

Chapter 2. Selecting the Data

The primary reason for a multisystem global resource serialization complex is to enable your installation to protect resources on shared DASD volumes as global resources rather than through reserves.

To ease migration to protecting these resources as global resources, global resource serialization provides three resource name lists (RNLs) — the SYSTEM inclusion RNL, the SYSTEMS exclusion RNL, and the RESERVE conversion RNL. The RNLs allow you to change resource scopes and convert reserves. IBM supplies default RNLs, shown in Figure 2-3 on page 2-9. You can modify the contents of the RNLs to define the resource serialization requirements of your installation. This chapter contains information you can use to select the resource names to place in each RNL.

Planning Aids

This chapter includes worksheets you can use to list the resources you want to include in each RNL. The format of the worksheets enables you to implement your plan easily.

Reference Book

Once you have completed your plan, see *Initialization and Tuning* for detailed information about how to define your RNLs in the GRSRNLxx parmlib member.

A major reason for carefully planning the contents of the RNLs is that the RNLs installed in all systems in the complex must be exactly the same. During its initialization, global resource serialization checks to make sure that the RNLs on each system are indeed identical. If they are different, global resource serialization does not allow the system to join the ring.

Also, changing any RNL requires a complex-wide IPL; all systems must shut down and then re-IPL. Plan your use of the RNLs carefully to avoid unnecessary IPLs.

Naming specific resources in the appropriate RNL enables you to build a global resource serialization complex without modifying any existing programs. Thus, your decisions about resources will probably focus on two questions:

1. What are my installation's short-term goals for the global resource serialization complex? Or, what must we do with the RNLs to get benefits from the complex as quickly as possible?
2. What are my installation's long-term goals for the global resource serialization complex? Or, what changes might we make to existing programs and procedures to make resource serialization more efficient?

Answering the first question requires you to analyze your use of data sets on shared DASD volumes to select the resources that are causing contention problems. As you do this analysis, focus on resources that are known to cause problems, such as resources that are frequently involved in interlocks. Also, remember that a shared DASD volume accessed by systems in the complex should not also be accessed by systems outside the complex. Answering the second question might require you to analyze the data set naming conventions at your installation.

You also must understand how global resource serialization uses the resource names in the RNLs and what suggestions and recommendations exist for various situations.

Data Set Naming Conventions

Using global resource serialization effectively requires an established installation-wide convention for naming data sets, especially those on shared DASD volumes. If your installation does not have a standard convention for naming data sets, you should probably define and implement a standard before trying to use a global resource serialization complex to serialize access to global resources.

Some data set naming conventions, such as one that relies heavily on system-dependent information like virtual storage addresses, do not work well with global resource serialization. If your installation has such a standard, you might want to redefine it as part of your long-term planning. A data set naming convention that works well with global resource serialization can avoid both very long RNLs and frequent changes to the RNLs.

In general, a data set naming convention that works well is one where the high-level qualifier or qualifiers have a meaning that proceeds from the general to the specific. For example, assume that an application named ACCOUNTS has three data sets: MASTER, TRANS, and ERRORS. If the data set names are ACCOUNTS.MASTER, ACCOUNTS.TRANS, and ACCOUNTS.ERRORS, you can use the high-level qualifier (ACCOUNTS) to cause all of the ACCOUNTS data sets to be either local resources or global resources.

Data set names that proceed from the general to the specific tend to work well with global resource serialization. Such names make it easy to identify and, if necessary, subdivide large groups of resources with a minimum number of entries in the RNLs. If your installation uses TSO/E, the format of the TSO/E userid is also important because it affects the names of user data sets.

RNL Processing

To evaluate how well your installation's data set naming conventions will work with global resource serialization, and to make decisions about specific resources, you need to understand how global resource serialization processes entries in the RNLs to determine if a specific resource is a local resource or a global resource.

Scanning the RNLs

Whenever global resource serialization encounters a request for a resource with a scope of SYSTEM or SYSTEMS, it invokes a single exit search routine (ISGGREX0) to scan the appropriate RNL. Your installation can replace the IBM-supplied exit search routine. See *Installation Exits* for more information.

To scan an RNL, global resource serialization compares the input search argument — the resource name — to the resource name entries in the RNL.

Entries in an RNL can be specific or generic. A **specific resource name entry** matches a search argument only when they are exactly the same. In contrast, a **generic resource name entry** is a portion of a resource name. A match occurs whenever the specified portion of the generic resource name entry matches the same portion of an input search argument. Thus, the exit search routine finds a match when:

- A specific resource name entry in the RNL matches the specific resource name in the search argument.

  For example, the specific entry APPL01,MASTER in the RNL matches APPL01,MASTER as a search argument.

  The length of the specific rname is important; for example, a specific rname of 'ABC' does not match a resource named 'ABC   '.

- A generic qname entry in the RNL matches the qname of the search argument.

  For example, a generic qname entry of APPL01 in the RNL matches any search argument with a qname of APPL01, such as APPL01,MASTER or APPL01,TRANS.

- A generic qname,rname entry in the RNL matches the corresponding portion of the resource name in the search argument.

  For example, a generic qname,rname entry of APPL01,MASTER in the RNL matches any search argument that begins with APPL01,MASTER, such as APPL01,MASTER or APPL01,MASTER2.

Each RNL entry indicates whether the name is generic or specific. Thus, a specific qname,rname entry of APPL01,MASTER would not match a search argument of APPL01,MASTER2.

Global resource serialization scans the RNL to determine if there is a match. The actions it takes, however, are different for each RNL.

SYSTEM Inclusion RNL

When global resource serialization encounters an ENQ or DEQ request for a resource with a scope of SYSTEM, it scans the SYSTEM inclusion RNL. If there is no match, global resource serialization processes the resource as a local resource.

If there is a match, global resource serialization changes the scope of the request from SYSTEM to SYSTEMS and processes the resource as a global resource. For example, if the resource a task requests is A,B,SYSTEM, and there is a match in the SYSTEM inclusion RNL, global resource serialization changes the resource name to A,B,SYSTEMS and processes A,B,SYSTEMS as a global resource. That is, global resource serialization then scans the SYSTEMS exclusion RNL for the resource name.

Thus, you can specify a generic name in the SYSTEM inclusion RNL, changing the scope of all resources with that generic name to SYSTEMS. You can then place specific resource names with that generic name in the SYSTEMS exclusion RNL, making those specific resources local resources. (The default RNLs use this technique to make certain system data sets local resources. See Figure 2-3 on page 2-9.)

SYSTEMS Exclusion RNL

When global resource serialization encounters a request for a resource with a scope of SYSTEMS, it scans the SYSTEMS exclusion RNL. Thus, it scans the SYSTEMS exclusion RNL when:

1. A task issues a RESERVE macro or specifies a scope of SYSTEMS on the ENQ or DEQ macro.
2. The resource name matched an entry in the SYSTEM inclusion RNL, which caused global resource serialization to change the scope to SYSTEMS.

If there is a match, global resource serialization changes the scope of the request from SYSTEMS to SYSTEM and processes the resource as a local resource. For example, if the resource name a program requests is A,B,SYSTEMS, and there is a match in the SYSTEMS exclusion RNL, global resource serialization changes the resource name to A,B,SYSTEM and processes A,B,SYSTEM as a local resource.

If there is a match and the task issued the RESERVE macro to request the resource, global resource serialization processes the resource as a local resource and does not scan the RESERVE conversion RNL; the system issues the reserve.

If there is no match, global resource serialization processes the request as a global resource. If the task issued the RESERVE macro to request the resource, global resource serialization then scans the RESERVE conversion RNL to determine if the system is to issue the reserve.

RESERVE Conversion RNL

Only when a task issues a RESERVE macro to request a resource and the resource name is not in the SYSTEMS exclusion RNL does global resource serialization scan the RESERVE conversion RNL.

If there is a match, global resource serialization suppresses the reserve for the global resource. It issues an ENQ with a scope of SYSTEMS to serialize access to the resource.

If there is no match, global resource serialization allows the system to issue the reserve. Thus, the RESERVE conversion RNL does not affect the scope of the resource but does determine whether or not the system is to issue the reserve.

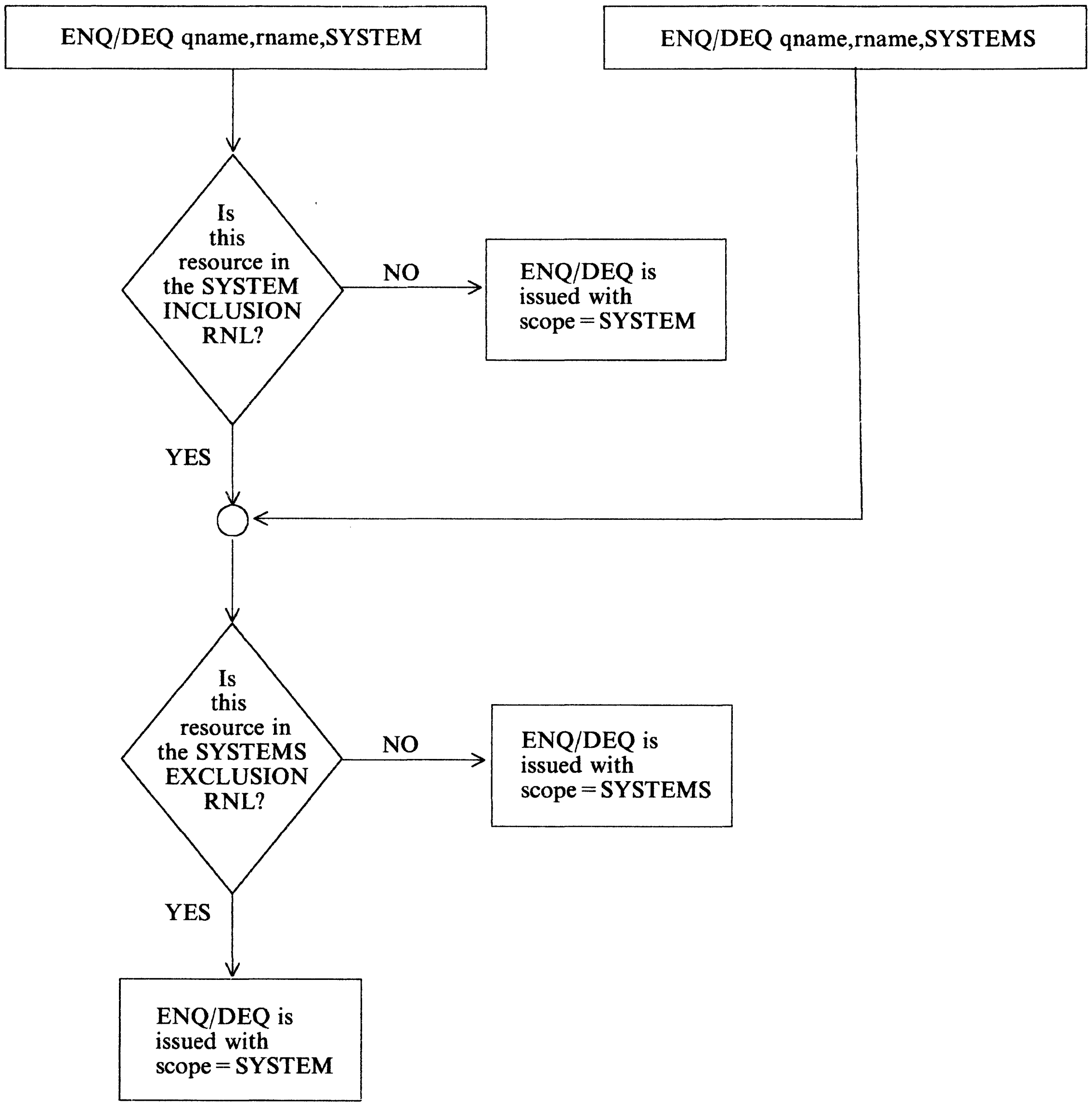
Choosing the reserves to be converted is a critical planning task; see "RESERVE Conversion" on page 2-7 for a full explanation of the considerations involved.

RNL Processing Sequence for ENQ and DEQ

Figure 2-1 summarizes the processing done for a resource requested by an ENQ or DEQ macro. The figure shows how global resource serialization uses the scope of the request and the RNLs to determine if the resource is a local or global resource. It also shows when global resource serialization internally changes the scope of the resource.

For an ENQ or DEQ request with a scope of SYSTEM, global resource serialization first scans the SYSTEM inclusion RNL. If it finds a match, it changes the scope to SYSTEMS and then scans the SYSTEMS exclusion RNL.

For an ENQ or DEQ request with a scope of SYSTEMS, global resource serialization scans only the SYSTEMS exclusion RNL.


ENQ/DEQ qname,rname,SYSTEM
ENQ/DEQ qname,rname,SYSTEMS
Is this resource in the SYSTEM INCLUSION RNL?
NO
ENQ/DEQ is issued with scope = SYSTEM
YES
Is this resource in the SYSTEMS EXCLUSION RNL?
NO
ENQ/DEQ is issued with scope = SYSTEMS
YES
ENQ/DEQ is issued with scope = SYSTEM

| *Figure 2-1. ENQ and DEQ Processing Summary*

RNL Processing Sequence for RESERVE

For a RESERVE request, global resource serialization first scans the SYSTEMS exclusion RNL to see if it should treat the resource as a local resource or as a global resource. If the resource is not named in the RNL and is thus a global resource, then global resource serialization scans the RESERVE conversion RNL to see if the global resource requires a reserve. If the resource is named in the SYSTEMS exclusion RNL, global resource serialization does not search the RESERVE conversion RNL; it treats the resource as a local resource and allows the system to issue the reserve.

Figure 2-2 summarizes the processing done for a resource requested by a RESERVE macro. The figure shows that global resource serialization first scans the SYSTEMS exclusion RNL to determine whether or not the resource is global. If there is no match, it scans the RESERVE conversion RNL to determine whether or not the system is to issue a reserve. also shows when global resource serialization internally changes the scope of a resource requested by a RESERVE macro.

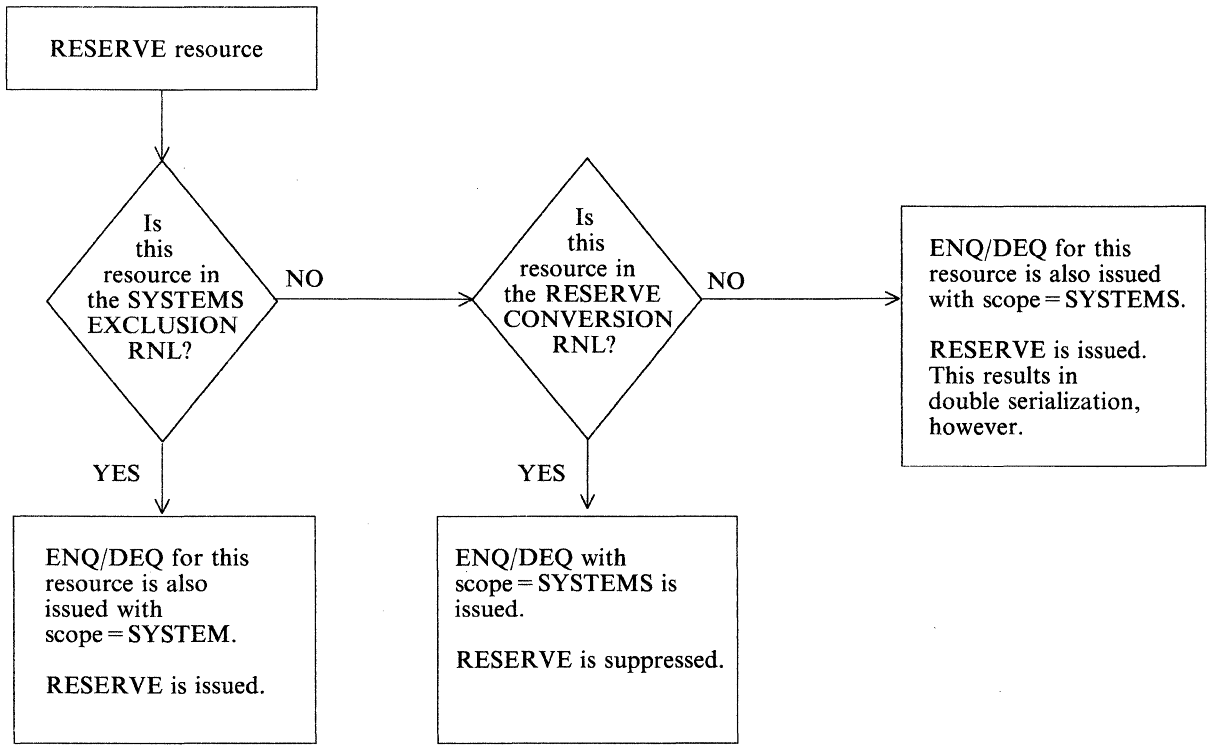

RESERVE resource
Is this resource in the SYSTEMS EXCLUSION RNL?
NO
Is this resource in the RESERVE CONVERSION RNL?
NO
ENQ/DEQ for this resource is also issued with scope=SYSTEMS.
RESERVE is issued. This results in double serialization, however.
YES
ENQ/DEQ for this resource is also issued with scope=SYSTEM.
RESERVE is issued.
YES
ENQ/DEQ with scope=SYSTEMS is issued.
RESERVE is suppressed.

Figure 2-2. RESERVE Processing Summary

RESERVE Conversion

One major purpose of global resource serialization is to eliminate the need to protect data sets on shared DASD volumes by issuing a RESERVE macro that causes a reserve of the entire volume. In general, you want to convert reserves to minimize problems your current methods of resource protection cause. These problems are:

- Interlocks
- Contention between jobs for the same volume
- The possibility that one system might monopolize a shared device
- The data integrity exposure that occurs as a result of a system reset while a reserve is in effect

By converting reserves, you make the systems and the resources more available. Not all reserves, however, should be converted. Thus, global resource serialization provides the following RESERVE conversion choices. You can:

1. Convert the RESERVE for a resource. Place the resource name in the RESERVE conversion RNL and do not place its name in the SYSTEMS exclusion RNL. Global resource serialization suppresses the reserve and treats the requested resource as a global resource.

2. Issue the reserve for the resource. Place the resource name in the SYSTEMS exclusion RNL. The system issues the reserve, and global resource serialization treats the requested resource as a local resource. It internally converts the scope of the request to SYSTEM to serialize access to the resource within the originating system. Global resource serialization then does not search the RESERVE conversion RNL, so the system issues the reserve to serialize access to the resource among multiple systems.

   This choice is a good way to handle a reserve that must be issued.

3. Do nothing. Do not place the resource name in either the RESERVE conversion RNL or the SYSTEMS exclusion RNL. The system issues the reserve, and global resource serialization treats the resource as a global resource.

   Because it causes both a reserve and a SYSTEMS ENQ, this choice is generally not a good way to handle any resource. Doing nothing, however, may lead to an interlock if jobs on different systems contend for multiple resources on the same shared DASD volume, as shown in the following example:

   1. Job A on system SYS1 issues a RESERVE macro for resource (A,B) on the DASD volume.
   2. Job B on system SYS2 issues a RESERVE macro for resource (C,D) on the DASD volume.
   3. Job A on system SYS1 issues a RESERVE macro for resource (C,D) on the DASD volume.

   Job B on system SYS2 owns the SYSTEMS ENQ portion of resource (C,D), but it is unable to reserve the DASD volume. Job A on system SYS1 holds the hardware reserve on the volume from its RESERVE of (A,B), but it is waiting for the SYSTEMS ENQ for (C,D). If resources (A,B) and (C,D) had been in either the SYSTEMS exclusion RNL or the RESERVE conversion RNL, this interlock would not have occurred.

In deciding which reserves your installation should convert, concentrate on reserves that are causing known problems. In addition, there are two general restrictions:

1. Do not convert a reserve for a resource on a volume if any system in your complex shares the volume with a system that is not part of the complex (or is part of another complex). In this case, you require the reserve to protect the data sets on the shared volume.
2. Do not convert the reserves for a resource when different systems in the complex use different names for the resource. This inconsistency can occur when the resource name includes system-dependent information, such as a control block address. Global resource serialization assumes that the different names represent different resources, and it cannot protect a single resource known by different names. If you modify existing programs to use the same name for the resource, you can then, of course, convert the reserves.

If an application issues a single reserve to serialize access to multiple resources on the same volume, you cannot convert the reserve unless you change the application.

Another general consideration is the use of the resource. You cannot convert some reserves because the application that uses the resource cannot tolerate the ring delay time required to process a request for a global resource. Your installation might have applications where dependencies on quick access to a resource outweigh the additional availability resulting from converting the reserve.

RNL Considerations

Because the resources to include in each RNL depend on the needs of your installation, it is not possible to supply default RNLs that will work well in every environment. IBM does supply a default for each RNL; Figure 2-3 shows these defaults.

The generic qname entry for SYSDSN in the SYSTEM inclusion RNL indicates that all data sets that go through MVS allocation (except for VIO and subsystem data sets, such as SYSIN, SYSOUT, and SUBSYS data sets) are to be global resources. The entries in the SYSTEMS exclusion RNL identify the system data sets with a qname of SYSDSN that specifically cannot be global resources.

Note: Only MVS/ESA and MVS/XA systems use SYS1.DAE. If your complex includes a system running an earlier version of MVS, the SYSTEMS exclusion RNL on that system, as well as the other systems, must include an entry for SYS1.DAE (because the RNLs on all systems in the complex must be identical.)

SYSTEM Inclusion RNL:

SYSDSN

SYSTEMS Exclusion RNL:

SYSDSN PASSWORD
SYSDSN SYS1.BRODCAST
SYSDSN SYS1.DAE (applies only to MVS/XA and MVS/ESA systems)
SYSDSN SYS1.DCMLIB
SYSDSN SYS1.DUMP (generic — all dump data sets)
SYSDSN SYS1.LOGREC
SYSDSN SYS1.MAN (generic — all SMF data sets)
SYSDSN SYS1.NUCLEUS
SYSDSN SYS1.PAGE (generic — all page data sets)
SYSDSN SYS1.STGINDEX
SYSDSN SYS1.SVCLIB
SYSDSN SYS1.UADS

RESERVE Conversion RNL:

The RESERVE conversion list is empty.

Figure 2-3. Contents of the Default Resource Name Lists (RNLs)

If all the systems in your complex include MVS/SP Version 3 or MVS/SP Version 2 (Release 1.2 or later), you can IPL the systems and build a ring using the default RNLs. An IPL with the default RNLs, however, might not reflect all the goals your installation has for the global resource serialization complex. For example, you might want to change the contents of the default RNLs to emphasize RESERVE conversion and avoid potential interlocks.

The DISPLAY GRS,CONTENTION command provides information about resources that are causing contention, and RMF reports can also help. The most useful RMF reports are:

- Monitor I enqueue activity report
- Monitor II system enqueue contention (SENQ) and system enqueue reserve (SENQR) reports
- Monitor III (workload delay monitor) reports on resource-oriented enqueue delays and resource-oriented device delays

Because a global resource serialization complex can consist of systems that include MVS/SP Version 1 (Release 2 or later), MVS/SP Version 2, or MVS/SP Version 3, and because each system control program usually works with multiple levels of other products, it is not possible to compile a complete and exhaustive list of recommended treatment for resources in all possible situations. There are, however, some general recommendations as well as some specific suggestions on known resources that are good candidates for the SYSTEM inclusion RNL, the SYSTEMS exclusion RNL, and the RESERVE conversion RNL. These recommendations and suggestions cover the following topics:

- CVOLs
- DFHSM
- ISPF or ISPF/PDF
- JES2
- JES3
- RACF

- Temporary data set
- TSO/E
- VSAM
- VSAM catalogs

The information is meant to give you an idea of the kind of decisions you might need to make, and it does not deal with the specifics of every level of every product.

Note: If you need more information about the macros mentioned in the explanation, see *SPL: Application Development Macro Reference.*

CVOLs

CVOLs (non-VSAM user catalogs) are protected by a reserve that includes the UCB address.

If you use CVOLs, you cannot convert the reserve because the UCB address is system-dependent.

DFHSM

If you are using DFHSM (or Release 3 of HSM) on all systems in the complex that run DFHSM, you do not need to take any action for user data set serialization as long as SYSDSN is named in the SYSTEM inclusion RNL. You can, however, convert some of the reserves that DFHSM issues. Information about these reserves, as well as the facts to consider when deciding whether or not to convert them, appears in *Data Facility Hierarchical Storage Manager Installation and Customization Guide*, SH35-0084.

ISPF or ISPF/PDF

To serialize access to resources with concurrent batch or TSO/E use of the resources, ISPF relies on MVS allocation (qname of SYSDSN).

To ensure the integrity of shared data, batch or TSO/E users who are updating a data set must allocate it with DISP=OLD.

Because MVS allocation does not satisfy an exclusive request and a shared request for the same resource at the same time, data set integrity is maintained between ISPF users and batch or TSO/E users.

To serialize access to partitioned data sets among multiple ISPF users, ISPF also issues its own ENQ, DEQ, and RESERVE macros.

To allow users to update a data set that has a record format of "U", ISPF serializes with the linkage editor to protect the entire partitioned data set.

Notes:

1. If both ISPF and SPF (meaning a pre-ISPF product, such as 5668-009 or 5740-XT8) are installed on the same system, there is a danger of destroying partitioned data sets that are being updated. This problem can occur when ISPF and SPF update the same data set at the same time.
2. A partitioned data set extended (PDSE) is a data set that like a PDS allows you to partition data into members. Unlike PDS members, members in a PDSE cannot contain executable programs; however, users can access PDSE members more easily and quickly than they can PDS members. For information about PDSE data sharing, see *MVS/DFP Managing Non-VSAM Data Sets*.

If you use ISPF on more than one system in the complex and do not use SPF on any system in the complex:

1. Place entries in the SYSTEMS exclusion RNL for the SPFEDIT data sets that cannot be global resources. For example, create SPFEDIT entries for the system data sets that cannot be shared. These data sets have the same rnames as those specified for SYSDSN in the default SYSTEMS exclusion RNL (shown earlier in Figure 2-3 on page 2-9).
2. Place entries in the RESERVE conversion RNL to convert the SPFEDIT reserves ISPF issues for data sets that you want global resource serialization to protect as global resources.

   Note: If your complex includes both a system running ISPF and a system running SPF, you cannot use global resource serialization to serialize access to global resources among ISPF and SPF users. You must not include entries for either SPFEDIT or SPFDSN in the RESERVE conversion RNL.
3. Resources that ISPF users on more than one system might share with batch users or TSO/E users must be global resources defined with both a qname of SYSDSN and a qname of SPFEDIT. That is, you must define, either explicitly or by default, an entry in the SYSTEM inclusion RNL for SYSDSN,dsname.

JES2

Global resource serialization can work well in a JES2 environment because it can replace job scheduling as a method of preserving the integrity of the data on shared DASD volumes. All of the considerations on temporary data sets and TSO/E, however, apply to a JES2 environment, and there is an additional consideration.

Handling the reserve for the checkpoint data set in a JES2 multi-access spool configuration is a particularly critical decision. It is generally not a good idea to convert the reserve for the checkpoint data set. JES2 cannot tolerate the processing delay required to protect the data set as a global resource rather than with a reserve. Also, if a failure disrupts ring processing, JES2 cannot access its checkpoint data set, and it comes to a complete stop.

The location of the checkpoint data set, however, is another factor that affects your decision. There are two basic situations:

1. The checkpoint data set is the only data set on a volume or resides on a volume that contains no data sets that are ever serialized by RESERVEs. In this case, do not convert the reserve; include the name of the checkpoint data set in the SYSTEMS exclusion RNL.
2. The checkpoint data set resides on a volume that contains other data sets that are serialized by RESERVEs. Such use of a volume contradicts recommendations for placement of the JES2 checkpoint data set. If your installation requires such use, then you must convert the reserves for all resources on the volume, including the checkpoint data set. In this case, include the names of all data sets in the RESERVE conversion RNL.

Note: If your installation also uses an alternate checkpoint data set, the reserve JES2 issues for the primary checkpoint data set also provides serialization for the alternate. Checkpoint duplexing thus does not require additional action.

JES3

The major advantage of global resource serialization in a JES3 environment lies in the conversion of reserves to avoid interlocks and reduce contention for data sets on shared DASD volumes. Global resource serialization also provides the only way to serialize access across multiple systems to new non-specific DASD data sets that the Storage Management Subsystem (SMS) does not manage.

JES3 cannot tolerate the processing delay required to protect its checkpoint data set as a global resource rather than with a reserve. The possible actions are the same as those described under "JES2" on page 2-11.

RACF

If you are experiencing contention problems related to the RACF data base, consider converting the reserves, as long as all systems that access the RACF data base are MVS systems that are part of the complex. (If a VM system is sharing access to the RACF data base, you cannot convert the reserves.)

To convert the reserves, place a generic entry for SYSZRACF in the reserve conversion RNL.

In any case, do not put an entry that begins with 'SYSZRACF' or 'SYSZRAC2' in the SYSTEM inclusion RNL.

Temporary Data Sets

If you take no action, global resource serialization treats non-VIO temporary data sets as global resources. One reason for letting temporary data sets be global resources is that your installation can then run scratch functions (such as SCRATCH VTOC,SYS) safely at any time against shared volumes.

If your installation, however, wants global resource serialization to treat temporary data sets as local resources, the data set names must appear in the SYSTEMS exclusion RNL. The format of a temporary data set name, however, is not compatible with the format of an RNL entry. There is no way to create a generic entry that will work all the time in every installation. The format of a temporary data set name is one of the following:

```
SYSyyddd.Thhmmss.RA000.jobname.R0000nnn
SYSyyddd.Thhmmss.RA000.jobname.ddname
```

TSO/E

When one or more of the systems in your global resource serialization complex runs TSO/E, you must decide whether or not to treat SYS1.UADS and SYS1.BRODCAST as global resources and how to handle user data sets. If your installation uses the RACF data base in place of SYS1.UADS, see "RACF."

If your installation runs ISPF, see the considerations identified under "ISPF or ISPF/PDF" on page 2-10. How you handle user data sets is especially critical in a JES2 environment.

SYS1.UADS and SYS1.BRODCAST

The default SYSTEMS exclusion RNL includes entries for SYS1.UADS and SYS1.BRODCAST, causing them to be local resources while you decide whether your installation wants them to be global resources. To make this decision, your installation must investigate and measure:

- Resource requirements (that is, the resources required to merge multiple versions of the two data sets into a single version of each and test the new versions)
- Performance implications (that is, the performance of one version of each data set accessed by all users in contrast to multiple versions of the same data sets each accessed by a subset of those users)

There are, however, significant advantages to treating SYS1.UADS and SYS1.BRODCAST as global resources:

- Your installation has only two data sets to maintain, rather than two data sets for each TSO/E system in the complex.
- A user can logon from any system in the complex, allowing a better workload balance.
- For foreground-initiated background jobs, a user who specifies NOTIFY will always receive the job-ended message regardless of which system in the complex processed the job.

To cause global resource serialization to treat SYS1.UADS and SYS1.BRODCAST as global resources, you must:

1. Merge all existing versions of SYS1.UADS and SYS1.BRODCAST into a single version of each data set.
2. Modify the default RNLs:
   a. Delete the entries for SYS1.UADS and SYS1.BRODCAST from the SYSTEMS exclusion RNL.
   b. Add SYSIKJUA as a generic qname entry in the SYSTEM inclusion RNL to make SYS1.UADS a global resource.
   c. Add SYSIKJBC as a generic qname entry in the SYSTEM inclusion RNL to make SYS1.BRODCAST a global resource.

Related information appears in *TSO/E Customization*. Some of the information is repeated here for your convenience.

TSO/E User Data Sets

The data sets that are used only by TSO/E users include:

- Private user data sets that are not shared by other users or by batch jobs
- Temporary user-related data sets, such as ISPF, log, or recovery data sets
- Shared data sets, such as program libraries

If you use the default RNLs, the SYSDSN entry in the SYSTEM inclusion RNL defines all of these data sets as global resources.

If your installation wants all of them or some of them to be local resources, you must exclude them from global serialization. You can, if the structure of your userid allows, place a generic entry in the SYSTEMS exclusion RNL to define all TSO/E user data sets as local resources. If, however, a user might logon to different systems at different times, that user's data sets must be global resources. Place a generic entry for the userid in the SYSTEM inclusion RNL, and do not place an entry for the user in the SYSTEMS exclusion RNL.

If the structure of your userid does not allow you to create a generic entry to define all TSO/E user data sets as local resources, you can place a generic entry in the SYSTEMS exclusion RNL for the userid of each user whose TSO/E data sets are to be local resources. Omit the entry for a user whose TSO/E data sets are to be global resources. This method, however, might cause a very long RNL that you would have to change frequently. If the problem is significant at your installation, you might want to modify the exit search routine (ISGGREX0) to recognize the TSO/E user data sets that are to be local resources and exclude them from global serialization. See *Installation Exits*.

VSAM

Whether or not you can serialize access to VSAM data sets and VSAM and ICF catalogs as global resources depends on the level of VSAM installed on all systems in the complex. Only the current level of VSAM is compatible with global resource serialization. The current level of VSAM is available in MVS/370 DFP Release 1.1 or later, in MVS/DFP Release 2.1 or later, and in MVS/DFP Version 3. (If you need more information about VSAM, see the VSAM information at your installation.)

Thus, the major consideration related to VSAM is that you must not treat VSAM data sets as global resources or convert any reserves for VSAM and ICF catalogs unless all systems that might access the resources under any possible conditions include the current level of VSAM. There are additional specific considerations for VSAM data sets and for VSAM and ICF catalogs.

VSAM Data Sets

As a general guideline, treat SYSVSAM the way you treat SYSDSN. If you take no action, SYSDSN and SYSVSAM both identify global resources. If you make SYSDSN resources local, then make SYSVSAM resources local; delete the entry for SYSDSN from the default SYSTEM inclusion RNL and add an entry for SYSVSAM to the SYSTEMS exclusion RNL.

The following information provides more details about the serialization of VSAM data sets. There are three basic serialization techniques: allocation serialization, OPEN serialization, and READ/WRITE serialization.

Allocation serialization: During allocation, the system serializes on the VSAM data set. Because of the SYSDSN entry in the default SYSTEM inclusion RNL, global resource serialization treats the data set as a global resource.

If your application programs specify each possible name for the VSAM data set on a DD statement or always use the same name for the data set, you can use the allocation serialization to serialize access to the data set as a global resource.

Allocation serialization depends on the existence of a generic qname entry for SYSDSN in the SYSTEM inclusion RNL as well as on the user's specifying the correct parameter (DISP=OLD or DISP=SHR) in the JCL. It is thus a less effective method of control than OPEN serialization, which depends on the VSAM share options.

OPEN serialization: If the DD statement for the data set specified DISP=SHR, VSAM OPEN processing enforces the VSAM cross-region share options 1 and 2. (OPEN processing does not affect share options 3 and 4. For share options 3 and 4, users must provide their own READ/WRITE operation serialization.)

Global resource serialization automatically treats the data set as a global resource. If you do not want global resource serialization to treat VSAM data sets as global resources, place a generic entry for SYSVSAM in the SYSTEMS exclusion RNL.

Using this method means that, once a user on any system in the complex has opened the VSAM data set, no other user in the complex can delete the data set.

READ/WRITE operation serialization: Cross-region share options 3 and 4, where serialization is the users' responsibility, require the user to provide serialization for READ/WRITE operations as well as specify DISP=SHR on the DD statement. The usual way to protect the integrity of the data set is to serialize its use by RESERVE and DEQ macros.

If your application programs issue RESERVE macros for a VSAM data set, you can convert the reserves under the following conditions:

1. No application programs that run on systems outside the complex ever access the data set.
2. Your application programs use consistent and repeatable names for the data set on every system.
3. Your application programs always acquire and release any resources — VSAM data sets as well as any implied system resources — in the same order.
4. Your installation also converts any reserves that system services issue to the same volume on behalf of your application.

Unless all of these conditions exist, you cannot convert the reserves.

If you are converting the reserves, place entries for the data sets in the RESERVE conversion RNL. If you are not converting the reserves, place entries for the data sets in the SYSTEMS exclusion RNL. The form of the entry (generic or specific) and the qname depend on the specific conventions at your installation, but be sure that the entry or entries cover each name the application programs use for each data set.

VSAM and ICF Catalogs

VSAM and ICF catalogs are VSAM data sets, but VSAM recognizes catalogs and manages access to them in a special way. If catalog reserve contention is a concern at your installation, replacing VSAM catalogs with ICF catalogs can help. Converting catalog reserves can also reduce contention caused by catalog activity. If your installation wants to convert catalog reserves, the following considerations apply:

1. If any system in the complex does not include the current level of VSAM, you cannot convert the reserves for any catalog. Place generic qname entries for SYSIGGV2 and SYSZVVDS in the SYSTEMS exclusion RNL.
2. If all systems in the complex include the current level of VSAM, you can convert the reserves for a catalog as long as all systems using the catalog are part of the complex. To convert all catalog reserves, place generic qname entries for SYSIGGV2 and SYSZVVDS in the RESERVE conversion RNL. To convert the reserves for a specific catalog, place a qname,rname entry for SYSIGGV2 and a corresponding entry for SYSZVVDS in the RESERVE conversion RNL. In either case, remember that, to IPL the system, you cannot convert the reserve for the system master catalog; you must place an entry for the system master catalog in the SYSTEMS exclusion RNL.

If all systems in the complex are MVS/ESA systems, use specific entries rather than a generic entry. During ring processing, global resource serialization compresses specific entries for catalogs.

RNL Candidates

Based on these general recommendations, there are certain resources that are good candidates for a particular RNL. Figure 2-4 shows suggested resources for the SYSTEM inclusion RNL, Figure 2-5 shows suggested resources for the RESERVE conversion RNL, and Figure 2-6 shows suggested resources for the SYSTEMS exclusion RNL.

For each resource shown, the figures include information on the resource name and a brief description of why you should consider placing the particular resource in the RNL.

Note: You must specify the parts of the resource name shown in upper case letters exactly as shown, and you must replace the parts of the resource name shown in lower case letters with your installation-specific information.

Figure 2-4. SYSTEM Inclusion RNL Recommendations

Qname	Rname	Notes
SYSDSN	dsname (optional)	Include a generic qname entry for SYSDSN to make data sets that go through MVS allocation global resources; use entries in the SYSTEMS exclusion RNL to make specific data sets local resources. The default RNL contains a generic entry for SYSDSN. A generic name is most useful, but the name can also be specific.
SYSIKJUA	none	You must include this entry if the TSO/E data set SYS1.UADS is to be a global resource. The name must be generic. Remember to delete the entry for SYSDSN,SYS1.UADS from the default SYSTEMS exclusion RNL.
SYSIKJBC	none	You must include this entry if the TSO/E data set SYS1.BRODCAST is to be a global resource. The name must be generic. Remember to delete the entry for SYSDSN.SYS1.BRODCAST from the default SYSTEMS exclusion RNL.

Figure 2-5. RESERVE Conversion RNL Recommendations

Qname	Rname	Notes
SYSIEWLP	dsname (optional and padded to 44 bytes)	Include an entry to convert this reserve if all systems that share the resource are part of the complex. The entry serializes access between ISPF and the linkage editor. The name can be specific or generic.
SPZAPLIB	dsname	Include an entry to convert this reserve if all systems that share the resource are part of the complex. The name can be specific or generic.
SYSIGGV2	catalog dsname (optional and padded to 20 or 44 bytes)	Include this entry to convert the reserves for VSAM and ICF catalogs. Including this entry also avoids catalog interlocks. If you use specific entries and the dsname is less than 20 bytes, pad it to 20 bytes; if the dsname is more than 20 bytes, pad it to 44 bytes. If you use specific entries, however, global resource serialization creates compressed RSA-message descriptions, which increases ring capacity. Using generic entries reduces the number of RNL entries that you need.
SYSZVVDS	volser (optional)	Include this entry when you are converting catalog reserves to identify the serial number of the volume on which the catalog resides. However, converting SYSZVVDS reserves is not generally recommended because reserves are of short duration and I/O intensive. If you use specific entries, global resource serialization creates compressed RSA-message descriptions, which increases ring capacity. Using generic entries reduces the number of RNL entries that you need. See SYSZVVDS in Figure 2-6.
SYSZRACF	none	Include this entry to convert the reserves for the RACF data base. The name can be generic or specific.
SYSVTOC	none	If you plan to run DFDSS DEFRAG programs with MVS SP2.2 or a later version in a multisystem environment, you can convert SYSVTOC reserves by including a generic entry. Otherwise, SYSVTOC is generally not a good candidate for reserve conversion because reserves are of short duration and I/O intensive. See SYSVTOC in Figure 2-6.
any	any	Include an entry for any reserve that you want to convert. Global resource serialization will treat the resource as a global resource, and the system will not issue the reserve. The name can be specific or generic.

Figure 2-6. SYSTEMS Exclusion RNL Recommendations

Qname	Rname	Notes
SYSZVVDS	none	Include this entry when you want to treat SYSZVVDS reserves as local reserves. The name is generic.
SYSZARC	ACTIVE	You must include this entry if (1) more than one system in the complex runs HSM and (2) at least one of these systems runs HSM Release 2. The name must be specific.
SYSVTOC	volser of SYS1.DCMLIB data set	For MVS releases earlier than SP2.2, include this entry to IPL with MVS/XA DFP. The name can be specific or generic.
		For MVS SP2.2 and later releases, you do not need to include this entry.
SYSVTOC	none	Include this entry when you want to treat SYSVTOC reserves as local reserves. The name is generic.
SYSZJES2		In a JES2 environment, include an entry for each checkpoint data set specified in the CKPTDEF initialization statement. The name can be specific or generic. The qname is always SYSZJES2
	volser SYS1.dsname	For MVS/SP Version 1.3.6 or MVS/SP Version 2 Release 1.5, use the volser and dsname specified for PRIMARY and DSNAME in CKPTDEF; for MVS/SP Version 2.2 or later releases, include a separate entry for each checkpoint data set and use the volser and dsname specified for CKPT1, CKPT2, NEWCKPT1, and NEWCKPT2 in CKPTDEF. See *JES2 Initialization and Tuning*.
SYSZIAT	CKPT	In a JES3 environment, include an entry for the checkpoint data set. The name must be specific.
SYSDSN	dsname	Include an entry for every system data set that is to be a local resource. The default RNLs contain several (see Figure 2-3 on page 2-9). The name can be specific or generic.
SYSCTLG	none	You must include an entry for SYSCTLG. The name must be generic.
SYSVSAM	dsname (optional)	It is not generally recommended to include entries for VSAM data set resources. However, to make specific VSAM data set local resources include an entry for every VSAM data set that is to be a local resource. To make all VSAM data sets local resources, use a generic qname entry for SYSVSAM. In any case, you must treat SYSVSAM and SYSDSN resources the same way at your installation.

Defining the RNLs

The RNLs on all systems in the complex must be the same; that is, each RNL on each system must contain the same resource name entries, and these resource name entries must appear in the same order.

You use the GRSRNLxx parmlib member to define the RNLs. During IPL, specifying the desired member(s) on the GRSRNL system parameter tells global resource serialization where it can find the RNLs.

Note: RNLs for systems that include MVS/SP Version 1 Release 5 or MVS/SP Version 2 Release 1.1 must be defined in SYS1.LINKLIB. See Appendix B, "RNLs in LINKLIB" on page B-1. In a single complex, you can use both methods; you must, however, ensure that the RNLs contain the same resources and that the resources appear in the same order.

IBM supplies a default, member GRSRNL00 of SYS1.PARMLIB, that defines the default RNLs listed in Figure 2-3 on page 2-9. You can use this member as a base for your modifications, but make at least the following change to the default member. To have the system load your RNLs from parmlib, remove the first RNLDEF statement:

```
RNLDEF LINKLIB(YES)
```

This statement, included to ease migration for installations that have existing RNLs in LINKLIB, tells global resource serialization that the RNLs appear in member ISGGRNL0 in SYS1.LINKLIB.

Once you have removed this statement, you can either use GRSRNL00 as is, modify it, or create additional GRSRNLxx members. You might find it useful to leave GRSRNL00 as is and use one or more separate members to define your own entries. Separating the IBM-supplied entries from your installation-dependent entries can make future migration easier. You might also find it useful to set up one member for generic entries and another member for specific entries.

Initialization and Tuning contains detailed information about specifying the entries in GRSRNLxx. The worksheet shown in Figure 2-7 lists the RNL entries IBM supplies in GRSRNL00 and includes blank slots for your own entries.

Each resource entry consists of:

- An RNL identifier, which is RNL(EXCL) for the SYSTEMS exclusion list, RNL(INCL) for the SYSTEM inclusion list, and RNL(CON) for the RESERVE conversion list
- An entry type indicator, which is TYPE(SPECIFIC) if the entry is a specific name or TYPE(GENERIC) if the entry is a generic name
- A qname, in the form QNAME(name)
- An optional rname, in the form RNAME(name)

Test GRSRNLxx carefully to ensure that it does not include syntax entries. In SYS1.SAMPLIB, IBM supplies an RNL syntax checker that can detect such errors. Member ISGRNLCK of SYS1.SAMPLIB contains information about using the RNL syntax checker, or see the appendix of this book for information about the RNL syntax checker and its messages. To minimize the testing, create a single GRSRNLxx member, test it, and then copy it to the parmlibs of all systems.

Figure 2-7. GRSRNLxx Worksheet

List Identifier	**Type Indicator**	**Qname**	**Rname**
RNLDEF RNL(EXCL)	TYPE(SPECIFIC)	QNAME(SYSDSN)	RNAME(PASSWORD)
RNLDEF RNL(EXCL)	TYPE(SPECIFIC)	QNAME(SYSDSN)	RNAME(SYS1.BRODCAST)
RNLDEF RNL(EXCL)	TYPE(SPECIFIC)	QNAME(SYSDSN)	RNAME(SYS1.DAE)
RNLDEF RNL(EXCL)	TYPE(SPECIFIC)	QNAME(SYSDSN)	RNAME(SYS1.DCMLIB)
RNLDEF RNL(EXCL)	TYPE(GENERIC)	QNAME(SYSDSN)	RNAME(SYS1.DUMP)
RNLDEF RNL(EXCL)	TYPE(SPECIFIC)	QNAME(SYSDSN)	RNAME(SYS1.LOGREC)
RNLDEF RNL(EXCL)	TYPE(GENERIC)	QNAME(SYSDSN)	RNAME(SYS1.MAN)
RNLDEF RNL(EXCL)	TYPE(SPECIFIC)	QNAME(SYSDSN)	RNAME(SYS1.NUCLEUS)
RNLDEF RNL(EXCL)	TYPE(GENERIC)	QNAME(SYSDSN)	RNAME(SYS1.PAGE)
RNLDEF RNL(EXCL)	TYPE(SPECIFIC)	QNAME(SYSDSN)	RNAME(SYS1.STGINDEX)
RNLDEF RNL(EXCL)	TYPE(SPECIFIC)	QNAME(SYSDSN)	RNAME(SYS1.SVCLIB)
RNLDEF RNL(EXCL)	TYPE(SPECIFIC)	QNAME(SYSDSN)	RNAME(SYS1.UADS)
RNLDEF RNL(INCL)	TYPE(GENERIC)	QNAME(SYSDSN)	

Chapter 3. Designing the Complex

In a sense, the decisions you make about how you want to process requests for various resources are the decisions that set your installation's goals for global resource serialization. The actual global resource serialization complex that you design is one of the tools you use to achieve these goals.

As stated earlier, a global resource serialization complex consists of all the systems that are able to share global resources. There are links between the systems that enable them to communicate, primarily by passing the RSA-message from one system to another. Each system in the complex indicates at IPL time that it is to be part of the complex.

Designing the complex involves answering three basic questions:

1. What resources does your installation want to share?
2. Which systems use these resources?
3. What communication links are available?

The resources your systems need to share determine the systems in the complex. The most likely candidates, of course, are those systems that are already serializing access to resources on shared DASD volumes and, especially, those systems where interlocks, contention, or data protection by job scheduling are causing significant problems.

You then need to decide how many of these systems to combine into one complex. The design of global resource serialization allows up to 32 systems per complex. However, the practical limit is much lower, and the actual number of systems your particular installation can reasonably configure into a complex depends on a number of factors. You should, for example, consider the operations and performance implications of a very large complex.

It is, of course, possible for a single installation to have two or more global resource serialization complexes, each operating independently. However, the independent complexes should not share resources. Also, there should be no common links made available to global resource serialization on any two complexes.

To avoid a data integrity exposure, ensure that no system outside the complex can access the same shared DASD as any system in the complex.

Once you have selected the systems that are to be part of the complex, you must then define the communication links that connect the systems. Use the information in "Choosing the Configuration" on page 3-2 to help you with this process.

Note: "Systems" refers to the number of MVS operating systems, not the number of processors. For example:

- An IBM 3090 Processor Complex Model 400E could be partitioned into two MVS systems.
- In a PR/SM environment, as many as four or eight MVS systems can run in the same processor complex.
- Under VM/XA, multiple MVS systems can run as second-level systems in the same processor complex.

In any of these environments, each MVS system can be an independent system in a global resource serialization complex.

Choosing the configuration is one major part of designing the complex; the other is defining the processing options that your complex needs. For example, you must decide:

- The length of the residency time value, that is, the length of time a system is to hold the RSA-message before sending it on to the next system in the ring
- The length of the tolerance interval, that is, the length of time that global resource serialization is to wait for the incoming RSA-message before it signals a disruption
- Whether or not the system can automatically rebuild a disrupted global resource serialization ring
- Whether or not the system can automatically rejoin the ring after it has temporarily stopped

Use the information in "Processing Options" on page 3-8 to make these decisions.

Note: This book describes how global resource serialization works on an MVS/ESA system (a system that includes MVS/SP Version 3). A mixed complex, a global resource serialization complex that includes MVS/XA systems, for example, is also possible. In general, the differences are minor and noted where they apply.

Planning Aids

"Defining the Complex to MVS" on page 3-16 includes a blank configuration diagram and a worksheet for recording the design of your complex. The diagram and worksheet can help you implement your plan.

Reference Book

Once you have completed your plan, see *Initialization and Tuning* for information about how to specify the global resource serialization parmlib member and system parameters.

Choosing the Configuration

Each communication link can be either a data link in an IBM 3088 Multisystem Channel Communication Unit (MCCU) or an integrated channel-to-channel adapter (CTC adapter). In this book, "link" or "CTC link" means either an IBM 3088 data link or an individual CTC adapter.

If your hardware configuration includes an IBM 3088 Multisystem Channel Communication Unit, the number of available communication links is probably not a problem:

- Model A1 can connect two systems through up to 63 data links.
- Model 1 can connect up to four systems through up to 126 data links.
- Model 2 can connect up to eight systems through up to 252 data links.

Each data link is separately addressable and performs exactly the same function as a CTC adapter.

In contrast to the integrated CTC adapter, the IBM 3088 offers many advantages, summarized in the following table.

IBM 3088 MCCU	CTC Adapter
The 3088 allows as many data links as you need.	There can be only one CTC adapter per processor.
The 3088 can share its channel with other control units. Early channel disconnect allows multiple paired links on the same channel.	IBM recommends that a CTC adapter be the only device on its channel.
Channel cables can be up to 400 feet in length. Also, the channel connecting a processor to a 3088 can be extended with fiber optics (IBM 3044 Fiber Optic Channel Extender Link).	Channel cables can be up to 200 feet in length.
The 3088 is powered independently of the processors.	The CTC gets its power from the processor.
The 3088 has a data transfer rate of up to 4.5MB/second.	The CTC has a data transfer rate of 1.5MB/second.

IBM thus recommends that you use the IBM 3088 MCCU to provide channel-to-channel communication among the systems in a global resource serialization complex. Throughout this book, any configuration examples assume that a 3088 provides the communication links.

Link Placement: Global resource serialization requires fast communication between systems. Without channel interference, this communication normally requires only a few milliseconds. To achieve the optimum communication speed, place the 3088 on a 3-megabyte or faster data streaming block multiplexor channel.

Other data links used by other MVS subsystems can share the channel, but do not connect data links on the same channel as any devices that keep the channel busy for a long period of time. For example, IBM 3880 DASD control units might be compatible while tape and terminal control units are not. The 3088, however, might monopolize the channel, causing delays in I/O operations to DASD.

Note: If one or more of the systems in the complex are running as VM guest systems, provide real CTC links (not virtual CTCs) and dedicate the CTC links to global resource serialization.

Other configuration recommendations are:

- Design a fully-connected complex — one where every system has a communication link to every other system. See "Level of Connectivity" on page 3-4 for more information.
- Provide at least one alternate (a second connection) for each communication link. Alternate links are required if you want to use ring acceleration to speed up the processing of global resource requests. See "Alternate Links" on page 3-6 for more information.
- Provide a backup IBM 3088. See "Backup Considerations" on page 3-7 for more information.

Following these recommendations ensures a complex that provides the best possible performance and availability. Configuration decisions have a significant effect on recovery planning.

Recovery

Any failure that disrupts ring processing requires recovery. Global resource serialization can both detect the failure and respond to it. Depending on options your installation selects, global resource serialization can automatically rebuild a disrupted ring. It can also automatically issue VARY commands to enable and disable communication links. These actions speed up recovery from a failure and reduce operator intervention in the recovery process.

The primary causes of a failure that requires recovery are:

- A system fails because of a software problem, which can be either related to ring processing or independent of ring processing.
- A system fails because of a hardware problem.
- A CTC link fails. The failure can occur in the link itself or in any hardware or software component required in the communication path.
- Global resource serialization detects either a temporary problem on a system in the ring (such as when an operator stops a system) as a system failure or a temporary delay in communication as a communication link failure.

Whatever the cause of a break in ring processing, the result is a ring disruption; global resource serialization suspends the processing of requests for global resources until it recovers from the failure.

In designing your complex, plan a complex that can recover from a system or a CTC link failure, regardless of whether the failure is temporary or not.

After a system failure, there should be enough CTC links available to reconfigure a subset ring of **n-1** systems, where **n** is the number of systems in the original ring. That is, you want a complex where a system failure means that only the failed system must withdraw from the ring.

After a CTC link failure, there should be enough CTC links available to reconfigure the original ring. That is, you want a complex where the failure of a single link does not force any system to withdraw from the ring.

To design a complex that can recover effectively from failures, consider both the level of connectivity and the need for alternate CTC links. Alternate links are CTC links that global resource serialization can use for ring acceleration, which improves global resource request response time. Alternate links are available for use when there is a failure on a link that global resource serialization is using to pass the RSA-message around the ring, which improves recovery. Providing an additional 3088 for backup also has significant advantages.

Level of Connectivity

The number of CTC links available to the complex determines, to a great extent, the availability, ease of operation, and performance of the complex.

A **fully-connected complex** exists when every system in the complex has a communication link to every other system in the complex. Each system in a fully-connected n-system complex has **n-1** communication links, where **n** is the number of systems in the complex. For example, each system in a fully-connected four-system complex requires three link connections.

In contrast, a **partially-connected complex** is one where not every system has a communication link to every other system. That is, some systems in an **n**-system complex have less than **n-1** communication links. For example, some systems in a partially-connected four-system complex have less than three communication links.

The level of connectivity of the complex affects the operation of the complex in two basic ways. One is the order in which one system starts the complex and other systems join the complex. A partially-connected complex of four or more systems requires coordination of IPLs so that global resource serialization can build a valid ring. The second effect is on recovery from a system or CTC link failure that cannot be repaired immediately. A partially-connected complex limits the reconfiguration options available to recover from the failure.

Figure 3-1 illustrates the problem of a system failure on a partially-connected complex. It shows an active four-system ring with five communication link connections.

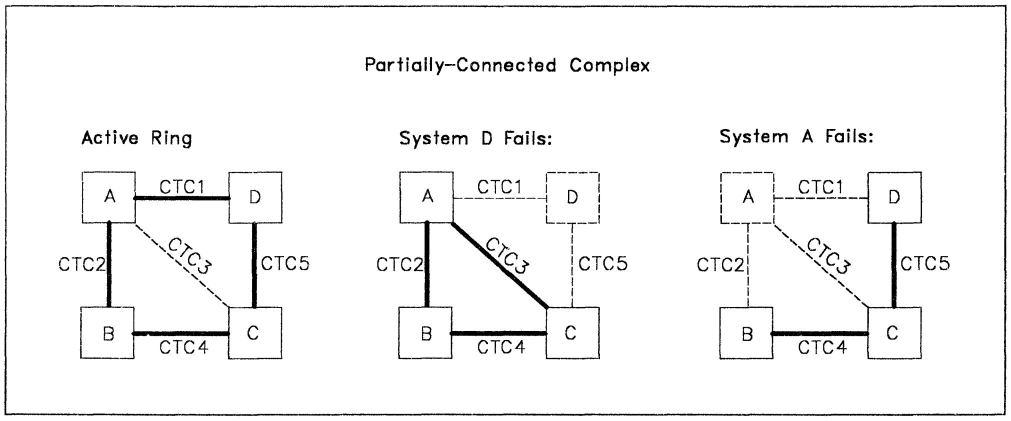


Figure 3-1. Recovery Problems with a Partially-Connected Complex

If system D fails, global resource serialization can automatically rebuild systems A, B, and C into a three-system ring using CTC2, CTC3, and CTC4, as shown in the figure. The three-system ring can resume processing requests for global resources. A three-system ring can recover just as quickly from a failure on system B.

If system A fails, however, the problem is entirely different because the complex is not fully connected. Global resource serialization cannot rebuild a three-system ring. It can automatically rebuild a two-system ring consisting of systems D and C (using CTC5) or a two-system ring consisting of systems B and C (using CTC4), and there is no way to predict which ring it would build. The same problem occurs if system C fails.

If the complex shown in the figure were fully connected, it would include a CTC link between system B and system D. With this fully-connected complex, global resource serialization can respond to a failure on any system by rebuilding a three-system ring that can quickly resume processing requests for global resources.

The distinction between a fully-connected complex and a partially-connected complex does not exist for a two-system complex or a three-system complex. For these complexes, the number of communication link connections required for full connectivity and the minimum number of communication link connections required are the same.

IBM recommends that you design a fully-connected global resource serialization complex. Even a fully-connected complex, however, might not meet the level of reliability or performance your installation requires. Designing a complex that includes alternate links offers significant advantages for both reliability and performance.

Alternate Links

Global resource serialization uses one link, the primary link, to send the RSA-message from one system in the ring to another. Alternate links provide additional connections. At IPL time, global resource serialization uses one link as the primary connection and marks any other links as alternates. Alternate links provide two very important benefits.

If alternate links exist, you can use ring acceleration during normal ring processing. Ring acceleration speeds up the process of granting requestors access to global resources. See "Ring Acceleration (ACCELSYS)" on page 3-12 for more information.

Alternate links also provide improved recovery. If the primary link fails during ring processing, a ring disruption occurs. If an alternate link is available, however, global resource serialization automatically selects an alternate link to replace the primary link, and ring processing can resume almost immediately. If global resource serialization was using the alternate link to send the ring acceleration signal, and no other link is available, the ring acceleration signal can no longer be sent between the two systems. This loss might affect performance, but the ring continues processing.

Figure 3-2 shows the problem of a link failure in a three-system complex. If CTC1 fails, system A and system B cannot communicate. Global resource serialization can rebuild a two-system ring, but the two-system ring omits either system A or system B. One of the two systems cannot continue to serialize access to global resources.

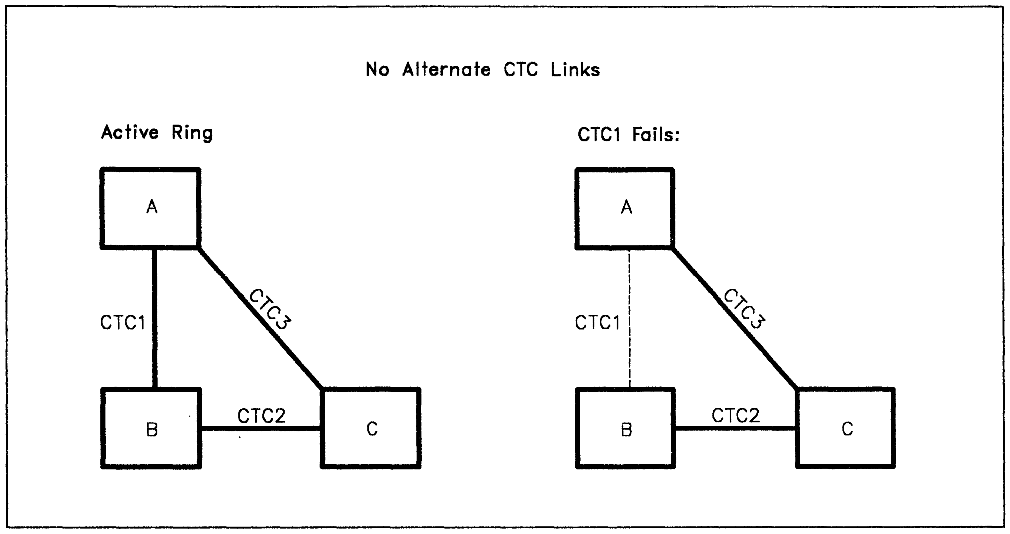


Figure 3-2. CTC Link Failure — No Alternate Available

Figure 3-3 shows how alternate links can solve this problem. If there is an alternate link between system A and system B, global resource serialization can use the alternate to rebuild the original three-system ring. The alternate link means that the two systems can continue to serialize access to global resources even when the primary link fails; global resource serialization uses the alternate in place of the primary, and ring processing continues.

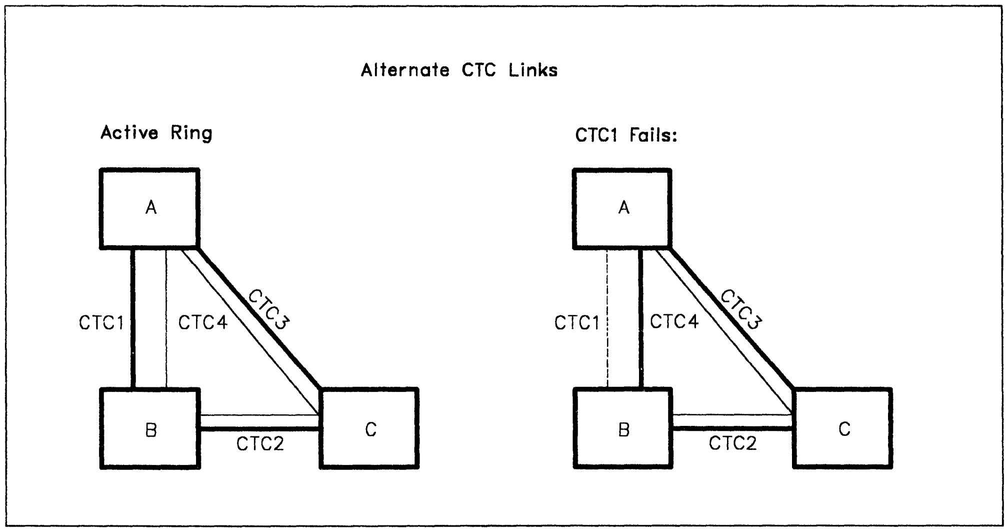


Figure 3-3. CTC Link Failure — Alternate Available

Because of the recovery and performance benefits, IBM recommends that you provide alternate links for the global resource serialization complex.

Backup Considerations

A fully-connected complex with alternate links provides the option of using ring acceleration as well as a very high level of reliability. It does not, however, remove the IBM 3088 as a potential single point of failure. Thus, IBM recommends that you provide an alternate 3088 to act as a backup. (Integrated CTC adapters can also act as backup connections.)

In such a configuration, dedicate two data links from the primary 3088 and two data links from the alternate 3088 for each connection in your fully-connected complex. If a system runs on a processor that, like the 3090 Model 400E, can be partitioned, connect links from both 3088 units to both sides of the processor complex.

Processing Options

Processing options provide the information that global resource serialization needs about the systems in the complex. Some of this information comes from the system parameters specified at IPL time. These parameters are:

- GRS=START or JOIN — indicates whether the system is to start or join the complex.
- SYSNAME=name — identifies the name the system has in the global resource serialization complex.
- GRSCNF=xx — identifies the GRSCNFxx parmlib member that defines the complex.
- GRSRNL=xx — identifies the GRSRNLxx parmlib member that holds the RNLs the system is to use.

Chapter 2, "Selecting the Data" on page 2-1 described how to use GRSRNLxx to define the RNLs. This section describes how to use GRSCNFxx to define the processing options for the complex. You must create GRSCNF. Because there is no way of predicting the configuration of a particular installation's complex, IBM does not supply a default GRSCNFxx member.

The information that you specify in GRSCNFxx for each system is:

- The name of the system. See "System Name (MATCHSYS)" on page 3-9.
- The specific device numbers of all CTC links attached to the system and available to global resource serialization. See "CTC Link Device Numbers (CTC)" on page 3-9.
- The length of the RSA-message residency time. See "Residency Time Value (RESMIL)" on page 3-9.
- The length of the tolerance interval — the length of time that global resource serialization is to wait for an overdue RSA-message before it signals a disruption. See "Tolerance Interval (TOLINT)" on page 3-11.
- Whether or not the complex is to use ring acceleration and, if so, how many systems must see the RSA-message before a system sends the shoulder-tap acknowledgement. See "Ring Acceleration (ACCELSYS)" on page 3-12.
- Whether or not the system can automatically rebuild a disrupted ring. See "Automatically Rebuilding a Disrupted Ring (RESTART)" on page 3-15.
- Whether or not the system can automatically rejoin a ring after the system has been temporarily stopped. See "Automatically Rejoining the Ring (REJOIN)" on page 3-15.

Initialization and Tuning contains complete syntactical information about how to create GRSCNFxx and how to specify the system parameters. Before you can create GRSCNFxx, however, you need to understand the options and what effect each option has on the complex.

System Name (MATCHSYS)

Global resource serialization matches the name specified on the SYSNAME system parameter at IPL time to a name specified on MATCHSYS to locate the information in GRSCNFxx for a particular system. Each name must be unique within the complex. It is a good practice to use the same four-character name for MATCHSYS that you use to identify the system to SMF, that is, the same value you specify for the SMF SID parameter. Consistent use of the same system name makes identifying the system in various records and messages easier and provides a consistent identifier for the operators.

The syntax of the MATCHSYS parameter in GRSCNFxx allows you to specify MATCHSYS(*). Global resource serialization considers MATCHSYS(*) to be a match to any SYSNAME value. If an operator enters an erroneous value for SYSNAME, global resource serialization sees MATCHSYS(*) as a match and might build the complex incorrectly. To avoid this problem, identify each system explicitly by name in GRSCNFxx and specify SYSNAME in IEASYSxx.

CTC Link Device Numbers (CTC)

Use the CTC option in GRSCNFxx to specify the device numbers of all CTC links attached to the system and available to global resource serialization. Specifying CTC links that you might have included at system generation but not yet installed causes global resource serialization to issue an error message (ISG046E) during system initialization.

When you have more than one CTC link for the same connection, global resource serialization uses the last device number specified as the primary link. Thus, the order in which you specify the device numbers affects how global resource serialization chooses the primary and the alternate link. If two links are online and available when a system joins the ring, global resource serialization chooses the second CTC link specified as the primary link and marks the first as an alternate. For example, if GRSCNFxx specifies CTC(860), followed by CTC(780), global resource serialization selects 780 as the primary and 860 as the alternate.

Residency Time Value (RESMIL)

Use the RESMIL option in GRSCNFxx to specify the residency time value. The residency time value is the minimum length of time the RSA-message spends in each system. If processing the RSA-message takes longer than the RESMIL value, the system holds the RSA-message until processing is complete. If processing takes less time, the system holds the RSA-message for the amount of time specified for RESMIL.

The RESMIL value you specify can significantly affect the performance of your complex. The default value is 35 milliseconds. In general, a low value, which means that the RSA-message passes around the ring more frequently, improves ring performance. That is, a low RESMIL value:

- Increases ring capacity — the number of global resource requests the ring can process
- Decreases response time — the amount of time the ring requires to process a specific request for a global resource.

Thus, setting the RESMIL value is very important to tuning your complex for efficient operation. Setting a low RESMIL value increases ring capacity and decreases response time, though it can slightly increase processor utilization. The RESMIL value, however is not the only factor that affects ring performance; the configuration and workload are also important.

Configuration Considerations

To set an optimal value for RESMIL, you must also consider the configuration:

- The transfer rate of the communication device
- The number of systems in the ring

To see how these factors relate, assume that you reduce by half the value of RESMIL on each system in a four-system ring. This action cuts the RSA-message cycle time roughly in half, thus approximately doubling ring capacity and cutting response time in half.

You can obtain a similar result by replacing a CTC adapter connection with an IBM 3088 link connection. The transmission rate of a 3088 data link is about three times as fast as the transmission rate of a CTC adapter. The faster communication speed reduces the RSA-message cycle time, thus increasing ring capacity and decreasing response time.

Adding a system to the ring, however, has the opposite effect; it decreases ring capacity and increases response time. The additional system adds time to the RSA-message cycle. If you are adding a system to the ring, decreasing the RESMIL value or switching to a faster communication device can maintain the existing ring capacity and response time.

Workload Considerations

The rate of global resource requests is the number of requests a system generates in a given time period. The rate depends on the workload. Thus, the workload on the systems in the ring also affects ring performance. TSO/E or a batch workload normally creates more global resource requests than a workload that is mostly CICS, IMS, or DB2.

The workload can also determine how important ring performance is. Ring performance is especially important to installations that:

- Run time-sensitive jobs that use global resources or jobs (like HSM back-out) that must complete in a predetermined amount of time
- Are experiencing resource contention problems
- Are planning to use the ring to serialize catalog access

Such installations need the lowest possible RESMIL value. See "Tuning the Complex" on page 5-3 for more information on performance.

RESMIL Recommendations

You do not have to specify the same RESMIL value on all systems, but there is no reason to set different values. The only effect of specifying a large value for one system is to increase the RSA-message cycle time, which decreases ring capacity and increases response time. It is simpler to use the same RESMIL value for all systems.

For a complex that consists only of small systems (such as systems running on an IBM 4381) or only of large systems (such as systems running on an IBM 3090 Model 400E), set RESMIL to the lowest value that works for your complex.

For a complex that includes systems running on both large and small processors, there is a basic problem: the total percent of a small processor's power spent on global resource serialization might be very high because it must process global resource requests generated on all systems.

Setting the RESMIL value differently according to the processor's power does not affect the time (or processor utilization) a system spends processing global resource requests; a system, regardless of the RESMIL value, processes all requests in the incoming RSA-message before it sends the outgoing RSA-message. The RESMIL value affects only the time (or processor utilization) a system spends on ring processing. There is no benefit in trying to use the RESMIL value to adjust for differences in processor power; only the average value is significant.

Thus, use the workload on the large processor to select the RESMIL value and set the same value on all systems. If the workload on the large processor generates many requests for global resources, set RESMIL to a lower value on all systems. Otherwise, make no adjustment for processor power.

The same recommendation applies to a ring where each system runs a distinct workload. Use the workload on the system that generates the largest number of global resource requests as a base for setting RESMIL on all systems.

See "Tuning the Complex" on page 5-3 for a description of some techniques you can use to determine the best value for your complex.

Tolerance Interval (TOLINT)

Use the TOLINT option in GRSCNFxx to specify the tolerance interval. The tolerance interval is the maximum amount of time that global resource serialization waits before it detects an overdue RSA-message. During normal processing, the RSA-message makes from tens to hundreds of trips around the ring each second. A system that fails or is stopped temporarily, or a link that fails or temporarily slows down communication, can cause a long delay of the RSA-message. If the delay is longer than the tolerance interval, global resource serialization detects an overdue RSA-message. An overdue RSA-message causes a ring disruption.

During a ring disruption, all tasks that request or free global resources are suspended because the RSA-message is halted and there is no communication between systems in the ring. As the ring disruption continues, more and more tasks are suspended, slowing the throughput of each system in the ring.

A ring disruption requires recovery. Global resource serialization can recover automatically from most ring disruptions when you specify automatic restart and automatic rejoin. Specifying RESTART(YES) and REJOIN(YES) allows recovery without operator intervention; global resource serialization issues messages but does not usually require operator action. See "Automatically Rebuilding a Disrupted Ring (RESTART)" on page 3-15 and "Automatically Rejoining the Ring (REJOIN)" on page 3-15.

The value you set for TOLINT affects how rapidly global resource serialization detects an overdue RSA-message, and setting the value properly requires a basic trade-off:

- To detect a system failure or a link failure, the best TOLINT value is one that recognizes the condition almost immediately.
- To deal with a temporary delay, the best TOLINT value is one that does not detect the condition. There are many reasons for a system entering a temporary stop, such as a spin loop or taking an SDUMP to capture the contents of common storage. For a temporarily stopped system or a temporary link delay, the best TOLINT value is one that is large enough to allow normal RSA-message processing to resume without causing a ring disruption.

Thus, the best TOLINT value is one that allows global resource serialization to detect a system or link failure promptly but does not cause it to continuously detect temporary delays.. If you specify RESTART(YES) and REJOIN(YES), setting a low TOLINT value has minimal effect because ring recovery is automatic. The default for TOLINT is three minutes. It is a good idea to change the default.

When all systems in the ring are MVS/ESA systems, a good value is between 20 and 30 seconds. In a mixed complex (all systems are not MVS/ESA systems), or when MVS is running in a PR/SM environment, a good value is between 40 and 60 seconds. If MVS is running as a guest under VM, set the TOLINT value even higher.

If your installation chooses not to use automatic restart and automatic rejoin, set a higher value. The higher value avoids unnecessary ring disruptions that require operator intervention.

The TOLINT value does not have to be the same for all systems, but it is a good idea to specify it consistently. If you set it to different values, the system with the smallest value is the first system to detect the disruption and initiate recovery.

Ring Acceleration (ACCELSYS)

Use the ACCELSYS option in GRSCNFxx to specify ring acceleration and the number of systems that must see a resource request before it is granted.

Without ring acceleration, every system in the ring must see each request for a global resource. While the RSA-message makes a complete cycle around the ring, the task that requested the global resource is suspended. This processing guarantees the integrity of resources; no global resource request is granted until all systems know about it. It does, however, mean that every task that requests a global resource must wait for at least one RSA-message cycle.

Ring acceleration offers an alternative technique, which protects the integrity of resources while potentially providing a significant reduction in global resource request response time (the time a task is suspended while waiting for ring processing).

Ring acceleration requires that all systems in the complex include MVS/SP Version 3. It also requires alternate links, used to send the ring acceleration signal from one system to another. In addition, IBM recommends that the complex be a fully-connected complex. Using ring acceleration can significantly improve ring performance in large complexes; in two-system complexes, it provides minimal benefits.

Figure 3-4 shows an example of a complex that could use ring acceleration. It is a fully-connected four-system complex with primary links, shown as heavy lines, and alternate links, shown as light lines.

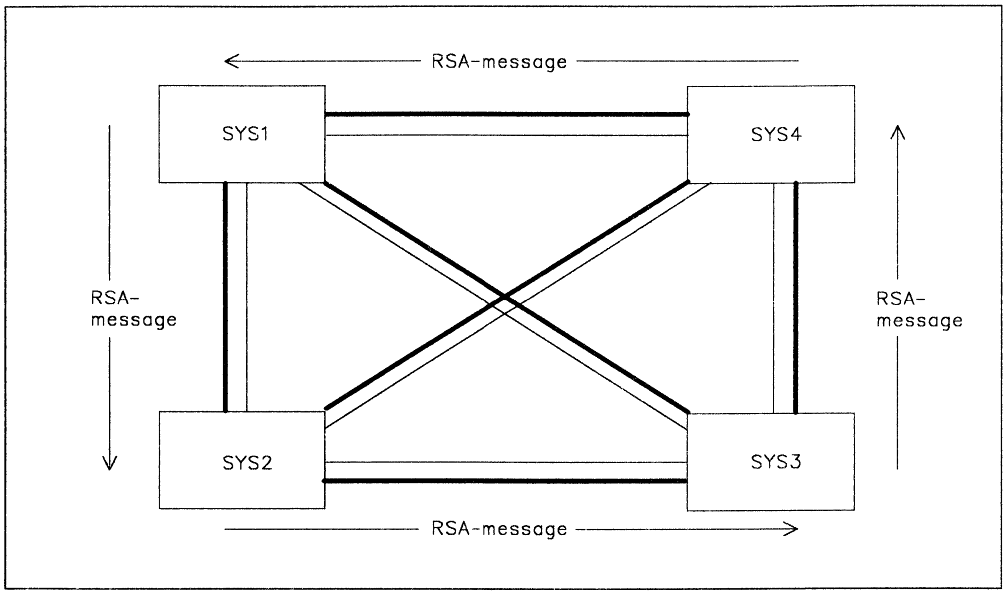


Figure 3-4. Ring Acceleration Configuration

Request Processing

Using Figure 3-4, assume that a task on SYS1 requested access to a resource and that the RSA-message is moving as shown. Without ring acceleration, the task on SYS1 would wait until the RSA-message made a complete cycle around the ring. Only when SYS2, SYS3, and SYS4 all know about the request can SYS1 grant the resource to the requestor.

In contrast, assume that GRSCNFxx contains ACCELSYS(2) to request ring acceleration. With ring acceleration, SYS1 still suspends the task that requested the resource, puts the request in the RSA-message, and sends the RSA-message on to the next system in the ring — SYS2 in this example.

SYS2, when it receives the RSA-message, uses the alternate link to send a shoulder-tap acknowledgement, the ring acceleration signal, to SYS1. SYS2 sends the signal because it is the second system to see the request, and ACCELSYS(2) means that two systems must see the request before it can be granted. After sending the shoulder-tap, SYS2 then processes the RSA-message. The RSA-message continues its cycle around the ring. All systems see and process the request, which preserves the integrity of the resource.

SYS1, as soon as it receives the ring acceleration signal, can grant the request. The task that requested the resource does not have to wait for the RSA-message to make a complete cycle around the ring. If the requested resource is available, the task can resume execution almost immediately.

Using ring acceleration can significantly reduce the amount of time that tasks must wait for access to global resources. On ACCELSYS, you specify the number of consecutive systems that must see the RSA-message before one of the systems sends the shoulder-tap to the originating system. If the complex shown earlier in Figure 3-4 used a value of ACCELSYS(3), a resource requested on SYS1 would be granted once SYS1 received a shoulder-tap from SYS3.

Recovery

Ring acceleration provides significant performance improvement, but it does introduce recovery considerations. The ACCELSYS value, as stated earlier, specifies the number of systems that must see the RSA-message before the originating system can grant a request. It also, in effect, specifies the number of consecutive systems that can fail before ring acceleration introduces a possible data integrity exposure.

For example, look again at the complex shown in Figure 3-4 and assume that ACCELSYS(2) is in effect. If any single system fails or is stopped temporarily, global resource serialization can safely rebuild the ring because at least one of the remaining systems has current resource information. If two non-consecutive systems, like SYS1 and SYS3, fail or are stopped temporarily, rebuilding the ring safely is still possible because either SYS2 or SYS4 has current resource information.

If, however, two consecutive systems, like SYS1 and SYS2, fail, and one of the failed systems held the RSA-message, then rebuilding the ring safely is not possible. There is a potential data integrity exposure because the failed systems are the systems that have current resource information. For example, assume that SYS2 held the RSA-message and sent the shoulder-tap to SYS1. SYS1, after receiving the shoulder-tap, granted access to one or more resources. If both systems then fail or are stopped temporarily, the active systems (SYS3 and SYS4) do not know about the resources granted on SYS1 and, if the ring were rebuilt, might grant other tasks access to these same resources.

In this situation, global resource serialization does not automatically rebuild the ring. Instead, it issues a unique operator message. The operator must follow normal installation procedures to resolve any data integrity exposure and then issue a restart command to rebuild the ring.

ACCELSYS Recommendations

To minimize the chance of such a problem, avoid temporarily stopping more than one system at a time. If your operators regularly issue VARY GRS(sysname),QUIESCE for a system before stopping it, you can avoid any recovery problems related to temporarily stopped systems. See Chapter 4, "Operating the Complex" on page 4-1 for additional information about operating a global resource serialization complex.

To use ring acceleration, specify ACCELSYS on all systems. ACCELSYS(2) provides the maximum performance benefits. Global resource serialization rejects an ACCELSYS value that is less than 2 or greater than 99. The operator must re-IPL with a valid value, or specify GRS=NONE to continue with the IPL. Specifying a value greater than the number of systems in the complex turns ring acceleration off. The default is ACCELSYS(99), which turns ring acceleration off.

Automatically Rebuilding a Disrupted Ring (RESTART)

Use the RESTART option in GRSCNFxx to indicate whether or not the system can automatically rebuild the ring when it detects an error that disrupts global resource serialization processing. Having a system rather than the operator initiate recovery has several advantages:

- It can speed up the recovery process significantly.
- It reduces operator intervention in recovery.
- It avoids the possibility of split rings, which can occur when more than one operator tries to restart the ring at the same time, causing the ring to split into multiple independent rings. See "Split Rings" on page 4-14.

Thus, specify RESTART(YES) whenever possible:

- In two-system complexes, specify RESTART(YES) on one system and RESTART(NO) on the other. Global resource serialization can automatically rebuild a one-system ring if the link fails or if the RESTART(NO) system fails. Specifying RESTART(YES)on both systems introduces the possibility of split rings if the link fails and there is no alternate available.

  Specifying RESTART in this way means that an operator must rebuild a disrupted ring only when:

  - The primary link fails and there is no alternate available
  - The RESTART(YES) system itself fails.

  If one system is more critical than the other, specify RESTART(YES) on the critical system and RESTART(NO) on the on the other; otherwise, arbitrarily choose one system as the RESTART(YES) system.

- In complexes of three or more systems, specify RESTART(YES) for all systems.

Note: An installation that specifies both RESTART(YES) and REJOIN(YES) can reduce the need for operator intervention in recovery from a ring disruption.

Automatically Rejoining the Ring (REJOIN)

Use the REJOIN option in GRSCNFxx to indicate whether or not a system that has been temporarily stopped can automatically rejoin the ring when the system resumes processing. A system that stops temporarily causes a ring disruption; the ring is rebuilt without the stopped system. REJOIN(YES) allows the system to automatically rejoin the ring when it resumes processing; no operator intervention is required. REJOIN(NO) means that the operator must bring the system back into the ring when the system resumes processing.

IBM recommends that you always specify REJOIN(YES) to avoid the problems of operator intervention and operator delay during recovery. REJOIN(YES) does not cause any potential ring processing problems or data integrity exposures. When the automatic rejoin does not succeed, global resource serialization issues messages, and the operator can then intervene to bring the system back into the ring. In general, the automatic rejoin succeeds when there is at least one active system in the ring and an available link between the active system and the rejoining system.

Specify REJOIN(NO) only when your installation needs absolute control over the process of rebuilding a disrupted ring.

Note: An installation that specifies both RESTART(YES) and REJOIN(YES) can reduce the need for operator intervention in recovery from a ring disruption.

Defining the Complex to MVS

A diagram consisting of system blocks and CTC link connection lines is a useful way to represent the configuration of a global resource serialization complex. To make the block diagram also useful as input to the process of defining your complex to MVS, expand the basic diagram to include the information that global resource serialization needs about the systems in the complex.

Figure 3-5 shows one way to expand the basic block diagram to combine the information MVS needs about your complex with the block diagram design of your complex. The complex shown in the figure consists of 4 systems and 24 CTC links. It is a fully-connected complex with alternate links. The configuration includes a primary IBM 3088 MCCU (3088-P) and a backup 3088 (3088-B). Figure 3-6 on page 3-18 is a blank configuration diagram for your use.

To define your complex to MVS, you use the GRSCNFxx parmlib member. You can create a single member that defines all systems in the complex and copy it to the parmlibs of all systems, or you can create a parmlib member for each system that defines only that system. For more efficient testing and control, it is better to define the entire complex in a single member, then copy that member to the parmlib of each system.

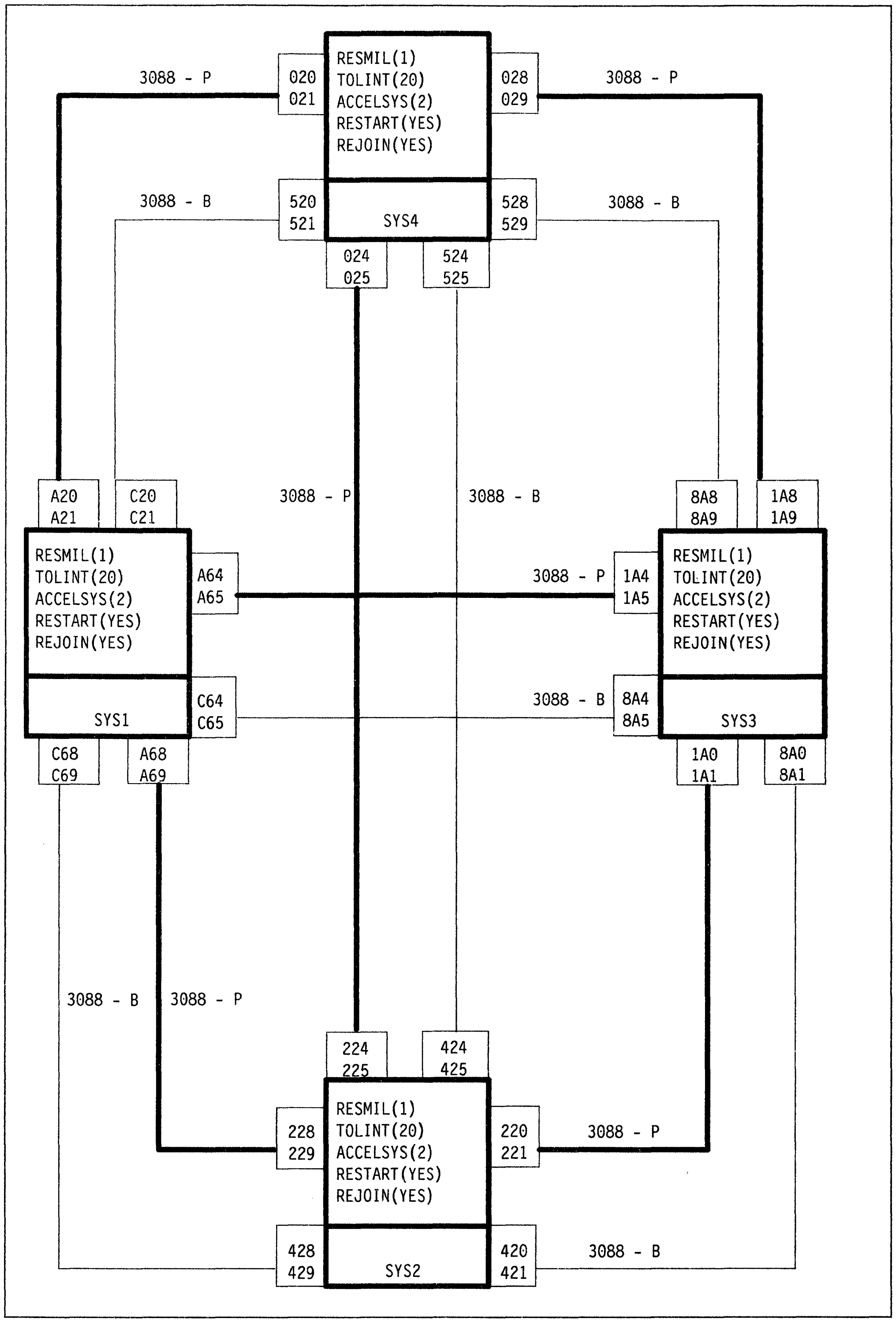


Figure 3-5. Sample Complex Design and Definition Diagram

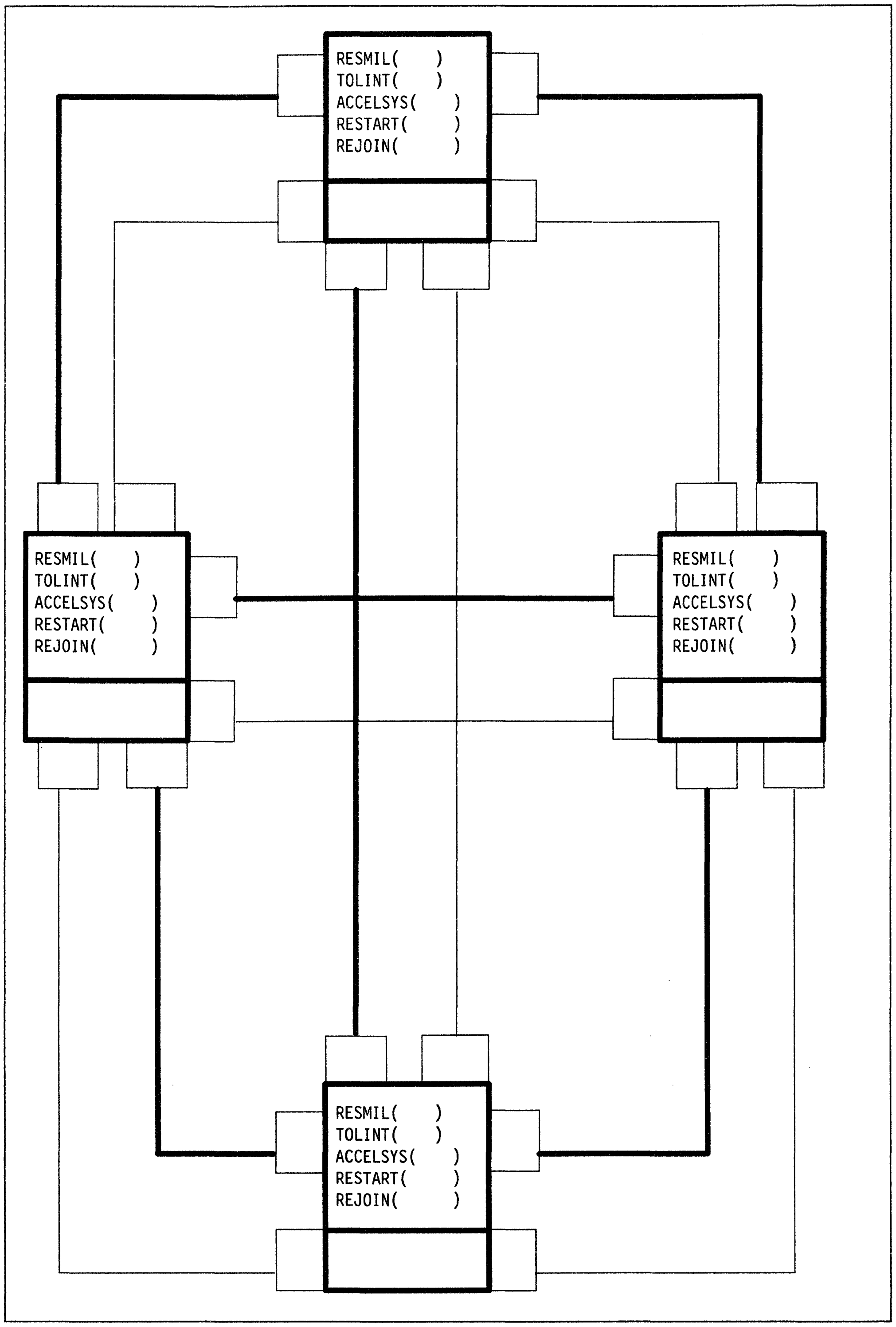


Figure 3-6. Design and Definition Diagram

GRSCNFxx Worksheet

One way to describe the design of your complex in a format you can use to actually create GRSCNFxx is to use the GRSCNFxx worksheet. Figure 3-7 shows a completed worksheet for the complex shown in Figure 3-5. Figure 3-8 is a blank worksheet for your use.

Figure 3-7 (Page 1 of 2). Sample Complex Definition Plan

Statement	**Parameter**	**Comments**
GRSDEF	MATCHSYS(SYS1)	Matches SYSNAME=SYS1 system parameter
	RESMIL(1)	RSA-message residency = 1 millisecond
	TOLINT(20)	Tolerance interval = 20 seconds
	ACCELSYS(2)	Ring acceleration definition
	CTC(A68)	3088-P link to SYS2
	CTC(A69)	3088-P link to SYS2
	CTC(A64)	3088-P link to SYS3
	CTC(A65)	3088-P link to SYS3
	CTC(A20)	3088-P link to SYS4
	CTC(A21)	3088-P link to SYS4
	CTC(C68)	3088-B link to SYS2
	CTC(C69)	3088-B link to SYS2
	CTC(C64)	3088-B link to SYS3
	CTC(C65)	3088-B link to SYS3
	CTC(C20)	3088-B link to SYS4
	CTC(C21)	3088-B link to SYS4
	RESTART(YES)	Can rebuild disrupted ring automatically
	REJOIN(YES)	Can rejoin ring after temporary stop
GRSDEF	MATCHSYS(SYS2)	Matches SYSNAME=SYS2 system parameter
	RESMIL(1)	RSA-message residency = 1 millisecond
	TOLINT(20)	Tolerance interval = 20 seconds
	ACCELSYS(2)	Ring acceleration definition
	CTC(228)	3088-P link to SYS1
	CTC(229)	3088-P link to SYS1
	CTC(220)	3088-P link to SYS3
	CTC(221)	3088-P link to SYS3
	CTC(224)	3088-P link to SYS4
	CTC(225)	3088-P link to SYS4
	CTC(428)	3088-B link to SYS1
	CTC(429)	3088-B link to SYS1
	CTC(420)	3088-B link to SYS3
	CTC(421)	3088-B link to SYS3
	CTC(424)	3088-B link to SYS4
	CTC(425)	3088-B link to SYS4
	RESTART(YES)	Can rebuild disrupted ring automatically
	REJOIN(YES)	Can rejoin ring after temporary stop

Figure 3-7 (Page 2 of 2). Sample Complex Definition Plan

Statement	Parameter	Comments
GRSDEF	MATCHSYS(SYS3)	Matches SYSNAME=SYS3 system parameter
	RESMIL(1)	RSA-message residency = 1 millisecond
	TOLINT(20)	Tolerance interval = 20 seconds
	ACCELSYS(2)	Ring acceleration definition
	CTC(1A4)	3088-P link to SYS1
	CTC(1A5)	3088-P link to SYS1
	CTC(1A0)	3088-P link to SYS2
	CTC(1A1)	3088-P link to SYS2
	CTC(1A8)	3088-P link to SYS4
	CTC(1A9)	3088-P link to SYS4
	CTC(8A4)	3088-B link to SYS1
	CTC(8A5)	3088-B link to SYS1
	CTC(8A0)	3088-B link to SYS2
	CTC(8A1)	3088-B link to SYS2
	CTC(8A8)	3088-B link to SYS4
	CTC(8A9)	3088-B link to SYS4
	RESTART(YES)	Can rebuild disrupted ring automatically
	REJOIN(YES)	Can rejoin ring after temporary stop
GRSDEF	MATCHSYS(SYS4)	Matches SYSNAME=SYS4 system parameter
	RESMIL(1)	RSA-message residency = 1 millisecond
	TOLINT(20)	Tolerance interval = 20 seconds
	ACCELSYS(2)	Ring acceleration definition
	CTC(020)	3088-P link to SYS1
	CTC(021)	3088-P link to SYS1
	CTC(024)	3088-P link to SYS2
	CTC(025)	3088-P link to SYS2
	CTC(028)	3088-P link to SYS3
	CTC(029)	3088-P link to SYS3
	CTC(520)	3088-B link to SYS1
	CTC(521)	3088-B link to SYS1
	CTC(524)	3088-B link to SYS2
	CTC(525)	3088-B link to SYS2
	CTC(528)	3088-B link to SYS3
	CTC(529)	3088-B link to SYS3
	RESTART(YES)	Can rebuild disrupted ring automatically
	REJOIN(YES)	Can rejoin ring after temporary stop

Figure 3-8 (Page 1 of 2). GRSCNF____ Definition

Statement	Parameter	Comments
GRSDEF	MATCHSYS(________)	
	RESMIL(______)	
	TOLINT(______)	
	ACCELSYS(_____)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	RESTART(YES|NO)	
	REJOIN(YES|NO)	
GRSDEF	MATCHSYS(________)	
	RESMIL(______)	
	TOLINT(______)	
	ACCELSYS(_____)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	RESTART(YES|NO)	
	REJOIN(YES|NO)	

Figure 3-8 (Page 2 of 2). GRSCNF____ Definition

Statement	Parameter	Comments
GRSDEF	MATCHSYS(________)	
	RESMIL(______)	
	TOLINT(______)	
	ACCELSYS(_____)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	CTC(______)	
	RESTART(YES|NO)	
	REJOIN(YES|NO)	
GRSDEF	MATCHSYS(__________)	
	RESMIL(________)	
	TOLINT(________)	
	ACCELSYS(_______)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	CTC(________)	
	RESTART(YES|NO)	
	REJOIN(YES|NO)	

Chapter 4. Operating the Complex

As shown in Chapter 3, "Designing the Complex" on page 3-1, the design of the global resource serialization complex is a critical factor in the successful operation of the complex. However, the procedures you provide for the operators of the systems in the complex are equally important.

You must educate the operators and prepare precise operator procedures for both normal operation of the global resource serialization complex and responses to various potential problems. Global resource serialization provides automatic recovery options (RESTART AND REJOIN) that allow you to minimize operator intervention in recovery. IBM recommends that you use automatic recovery whenever possible.

Reference Books

As you read this chapter and as you develop your operator procedures, consult *System Messages* for the exact text and responses to global resource serialization messages. Also, *System Commands* contains a complete description of how to use the VARY and DISPLAY commands with global resource serialization. It also includes a brief overview of global resource serialization from the operator's point of view.

The amount of education you provide for the operators depends, naturally, on the level of knowledge and skills your operators now have. Certainly you might consider an in-house workshop or seminar on what a global resource serialization complex is, what problems you expect it to solve at your installation, and what changes it will make in operations. In particular, the operators must understand that the systems in the complex are not isolated; an action taken on one system can affect all the other systems as well.

However, any general information on global resource serialization is just a starting point; in addition, your operators need specific information tailored to your particular installation. These procedures, often called a runbook or an operations workbook, describe precisely how your operators build the complex, keep an eye on its normal operation, and respond to the problems that automatic recovery does not handle.

Global resource serialization issues messages to the operator that describe key processing points, such as the fact that a system has started or joined the ring. Global resource serialization also issues messages that describe problems it encounters, ranging from invalid syntax for a system parameter to a system or CTC link failure that disrupts global resource serialization processing.

As part of your operations planning, read the ISG section of *System Messages* carefully. It describes all of the messages that global resource serialization issues. Use *System Messages* along with the general guidance and planning suggestions here to determine how you want operators to respond to the specific conditions that the messages describe.

In general, the messages, and thus your operational planning, focus on three areas: building the complex, normal operations, and recovery operations. The design of your complex affects all three areas. Chapter 3, "Designing the Complex" on page 3-1 describes the design considerations in detail, but the following list summarizes the most important recommendations:

- Use IBM 3088 data links, rather than CTC adapters, to connect the systems.
- Design a fully-connected complex, one where each system has at least one link to every other system.

  Note: You can, of course, operate a complex that is not fully connected. However, a minimal configuration can create availability, operations, and performance problems.
- Provide alternate links for each connection. Alternate links make recovery from a ring disruption easier. In addition, MVS/ESA systems can use the alternate link to send the ring acceleration signal.
- Provide a backup IBM 3088 to increase availability and eliminate the 3088 as a single point of failure.
- In GRSCNFxx, specify RESTART(YES) and REJOIN(YES) to minimize operator intervention during recovery.
- Tune the TOLINT value to meet your installation's needs. The TOLINT value determines the length of time required to detect a ring disruption.
- Tune the RESMIL value to meet your installation's needs. The RESMIL value determines the minimum length of time the RSA-message spends in each system.

Figure 4-1 shows a sample configuration diagram of a four-system complex. The system is fully-connected, and there are alternate links for all connections. Examples throughout this chapter use this configuration.

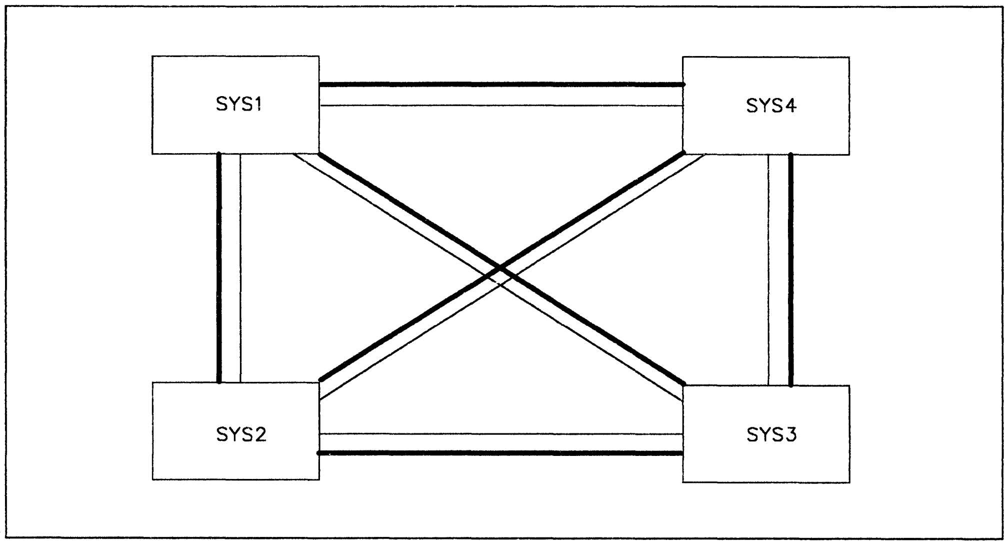


Figure 4-1. Sample Configuration

Building the Complex

The process of building a global resource serialization complex can have two phases: a configuration check and the IPL of the systems.

Configuration Check

Before an operator actually IPLs a system that is to start or join a global resource serialization complex, ensure that the operator verifies that the CTC link connections and the shared DASD connections are correct.

If the shared DASD connections are incorrect, a serious data integrity exposure could occur. This exposure occurs when systems in the complex are serializing access to a global resource by means of an ENQ macro with a scope of SYSTEMS. For example, if the RESERVE conversion RNL for the complex contains an entry for a resource, this entry causes global resource serialization on each system in the complex to suppress the reserve for that resource. If a system outside the complex can use a reserve to access the same resource at the same time, the resource could be damaged.

IPL

There are four system parameters (GRS, SYSNAME, GRSCNF, and GRSRNL) that indicate to MVS at IPL time that a system is to be part of a global resource serialization complex. These parameters remain in effect for the duration of the IPL; the only way to change a value is to IPL the system again.

As with all system parameters, there are several ways you can specify the global resource serialization parameters. To minimize operator intervention during IPL, it is generally best to specify SYSNAME, GRSRNL, and GRSCNF in IEASYSxx.

However, the best way to specify GRS is a less clear-cut choice:

1. If the same system always starts the complex, you can place GRS=START in IEASYSxx for that system and place GRS=JOIN explicitly or by default in IEASYSxx for the other systems.
2. You can tell the operators to enter GRS=START or GRS=JOIN at the console during IPL.
3. You can place GRS=JOIN either explicitly or by default in IEASYSxx for all of the systems. When you make this choice, the system that actually is to start the complex IPLs with GRS=JOIN. Global resource serialization issues a message stating that there is no active complex, followed by a prompting message. The operator can then respond START to the prompting message, but the system-operator interaction requires extra time.

It is both neater and more efficient to have the operator of the system that is to start the complex explicitly override the default by entering GRS=START in response to the SPECIFY SYSTEM PARAMETERS message. This procedure, which IBM recommends, is especially useful when different systems might start the complex at different times. One operator explicitly specifies GRS=START; the others use the default GRS=JOIN in IEASYSxx.

Whatever way you choose to specify GRS=START, the effect is the same. The system that starts the complex builds a one-system ring and issues a message stating that a complex is active. As each additional system IPLs with GRS=JOIN, an active system processes the request to join the complex. When a system IPLs with GRS=JOIN, global resource serialization issues messages on the joining system that identify the system that is assisting the joining system. Global resource serialization also issues messages on all systems in the complex to indicate that a new system is joining the ring. These messages indicate that the IPLs are proceeding normally and that the global resource serialization ring can process requests for global resources.

One major reason for having the systems IPL explicitly with GRS=START for the starting system and GRS=JOIN for any joining system is to make sure that two potentially critical messages appear only in abnormal situations. These messages are:

`ISG005I GRS START OPTION INVALID - SYSTEM sysname EXISTS IN A GRS COMPLEX`

This message, followed by a prompting message, can occur when the system IPLs with GRS=START after another system has already started the complex. It thus might be a "normal" or expected message, and the operator can respond JOIN to the prompting message. However, it can also occur when there is a serious problem, such as a combination of errors having created multiple independent complexes using incorrectly-connected CTC links or duplications of the same system name.

`ISG006I GRS JOIN OPTION INVALID - NO ACTIVE GRS SYSTEM`

This message, also followed by a prompting message, can occur when the system IPLs with GRS=JOIN before another system has started the complex. It thus might be a "normal" or expected message, and the operator can respond START to the prompting message. However, this message can also occur when there are some very serious problems, such as a combination of errors having created either multiple independent complexes using incorrectly-connected CTC links.

If systems IPL explicitly with GRS=START or GRS=JOIN, the operator can treat each occurrence of either of these messages as a potentially serious error rather than something that is usually normal. There is no chance the operator can think the serious error is a normal condition.

See "IPL Problems" on page 4-17 for additional information about planning the IPL process to minimize errors.

IPL Messages

Global resource serialization messages related to the IPL process include ISG001D, ISG002D, ISG003I, ISG004I, ISG005I, ISG006I, ISG007I, ISG008E, ISG009D, ISG010E, ISG011I, ISG014I, ISG041I, ISG042I, ISG043I, ISG044I, ISG045I, ISG061I, ISG062I, ISG063I, ISG064I, ISG065D, ISG066I, ISG067I, ISG068I, and ISG086E.

Normal Operations

Once the complex is built, it requires little, if any, operator intervention unless a problem disrupts ring processing or the ring requires reconfiguration for some other reason. If a problem occurs that is not related to ring processing, either global resource serialization or some other system component will detect the problem and issue messages that describe it before the operator could notice it.

For example, some of the error messages that global resource serialization issues indicate damage to resources or to the resource control blocks, rather than a problem with ring processing. These messages are ISG031E, ISG032E, ISG033E, ISG034E and ISG035E. The problem that causes any of these messages can also cause the job requesting the resource to terminate abnormally. If the damage is extensive, the problem can cause multiple jobs to terminate abnormally, requiring the system to IPL again to restore the control blocks. This problem is, of course, only one example of a problem not directly related to ring processing that can force a system in the ring to IPL again.

During normal processing, as well as during recovery from a ring disruption, operators use system commands to monitor and control global resource serialization. The system commands related to ring processing are:

1. DISPLAY GRS, which displays the status of each system in the ring. D GRS is the operator's primary way of checking ring processing and determining the source of problems. See "Displaying Ring Status" on page 4-6.
2. VARY GRS with the QUIESCE operand, which allows the operator to remove a system from the ring. See "Quiescing a System" on page 4-9.
3. VARY GRS with the PURGE operand, which allows the operator to remove a system from the complex. See "Purging a System" on page 4-10.
4. VARY GRS with the RESTART operand, which allows the operator to restart a quiesced system or an inactive system. See "Restarting a System" on page 4-13.
5. VARY *devnum*, which allows the operator to bring a link online or take a link offline. See "Controlling CTC Links" on page 4-15.

It is a good practice, whenever possible, to issue all VARY commands for the ring from the same system; this practice simplifies operations procedures, especially during recovery. See"Recovery Operations" on page 4-16.

Displaying Ring Status

The DISPLAY GRS (D GRS) command shows the state of each system in the ring and the status of the links that connect the systems. Note that D GRS shows system status only as it relates to the global resource serialization ring. D GRS does not reflect how well a system is running generally; for example, MVS on a system shown as QUIESCED or INACTIVE in the global resource serialization ring might run successfully for quite a while.

You can also use D GRS to display the local and global resources requested by the systems in the ring, contention information, or the contents of the RNLs. These uses are described in *System Commands*.

You can issue D GRS from any system in the ring and at any time after the ring has been started. The D GRS display shows the status of the ring from that system's point of view; thus, the displays issued from different systems might show different results. Figure 4-2 shows an example of the information D GRS produces and explains the values that can appear in each field.

D GRS is most useful, however, when a ring failure has occurred. The information displayed can help the operator to make informed decisions about the cause of an error and the correct response to the problem. Note that D GRS does not diagnose a problem; it simply reports status. Figure 4-3 on page 4-8 shows how an operator can use D GRS to determine the cause of a problem with ring processing.

```
18.40.07  ISG020I  18:40:06  GRS STATUS 340
SYSTEM    STATE       COMM    SYSTEM    STATE      COMM
 SYS2      ACTIVE              SYS1      ACTIVE     YES
 SYS3      QUIESCED    YES     SYS4      QUIESCED   NO

LINK      STATUS      TARGET  LINK      STATUS     TARGET
 220       ALTERNATE   SYS3    420      ALTERNATE   SYS3
 221       ALTERNATE   SYS3    421      ALTERNATE   SYS3
 224       QUIET       SYS4    424      QUIET       SYS4
 225       QUIET       SYS4    425      QUIET       SYS4
 228       ALTERNATE   SYS1    428      ALTERNATE   SYS1
 229       IN-USE      SYS1    429      ALTERNATE   SYS1
```

SYSTEM The name of the system. The first system shown is the system on which the command was entered.

STATE The state of the system at the time when the command was issued. There are seven possible states:

- **ACTIVE** The system is part of the ring and is actively participating in global resource serialization. ACTIVE is the normal condition. The system accepts all commands related to ring processing.
- **QUIESCED** The system is temporarily suspended from the ring, in response to either a ring disruption or operator command. The system does not have current information about global resources and is not currently processing global resource requests. Users of global resources retain ownership, but any users who try to obtain or free a global resource are suspended. The system remains quiesced, and the users remain suspended, until the system is restarted.
- **INACTIVE** The system is not part of the ring. INACTIVE appears when a ring disruption has occurred. The system has current information about global resources but is not currently processing global resource requests. Users of global resources retain ownership, but any users who try to obtain or release a global resource are suspended. Multiple systems can be INACTIVE, and an inactive system can restart the ring. An inactive system remains inactive until any system in the complex is restarted. **Note:** Access to local resources is not affected, but an attempt to cancel a job might not succeed if a global resource is involved.
- **JOINING** The system is joining the ring as part of its IPL process.
- **RESTARTING** The system is re-entering the ring as a result of a RESTART command.
- **ACTIVE+VARY** The system is executing a VARY GRS command.
- **ACTIVE+WAIT** A VARY GRS command was issued, but it is waiting because another VARY GRS command is now executing. When ACTIVE+WAIT appears, another system normally shows ACTIVE+VARY.

COMM An indication of whether or not the system has responded to a request for status. YES indicates that the system shown can communicate with the system issuing D GRS. NO indicates that there is no communication link, the system is temporarily stopped, or the system has failed. If NO appears, the state shown for the system might not be accurate. The field is blank for the system that issued D GRS.

LINK The address of each CTC data link defined for global resource serialization on the system.

STATUS The status of the link. There are four possible states:

- **IN-USE** The link is a primary link now being used to send the RSA-message from one system to another. IN-USE appears only for a link that connects active systems.
- **ALTERNATE** The link is an alternate. If a primary link fails, an alternate link can automatically replace it. An alternate link might be used to send the ring acceleration signal, which D GRS does not report; if an alternate link used for ring acceleration replaces a failed primary link, it can no longer send the ring acceleration signal.
- **DISABLED** The link is not physically connected or was taken offline because of an error.
- **QUIET** The link does not have any apparent problems, but the system it connects to did not respond to the request for status.

TARGET The name of the system that last responded from the other side of the link. The field is blank when the link has been disabled since the IPL of the system or when the system did not respond to the request for status.

Figure 4-2. D GRS Explanation

```
18.40.07  ISG020I  18:40:06  GRS STATUS 340
SYSTEM    STATE       COMM    SYSTEM    STATE       COMM
 SYS2      ACTIVE              SYS1      ACTIVE      YES
 SYS3      QUIESCED    YES     SYS4      QUIESCED    NO

LINK      STATUS      TARGET  LINK      STATUS      TARGET
 220       ALTERNATE   SYS3    420       ALTERNATE   SYS3
 221       ALTERNATE   SYS3    421       ALTERNATE   SYS3
 224       QUIET       SYS4    424       QUIET       SYS4
 225       QUIET       SYS4    425       QUIET       SYS4
 228       ALTERNATE   SYS1    428       ALTERNATE   SYS1
 229       IN-USE      SYS1    429       ALTERNATE   SYS1
```

The COMM field for system SYS2 is blank; the D GRS command was issued on system SYS2. The display shows the following:

1. System SYS2 and system SYS1 are active; they are processing global resource requests.
2. System SYS2 and system SYS1 are using link 229 to send the RSA-message. Link 229 is a primary link; its status is IN-USE.
3. All other links between system SYS2 and system SYS1 (228, 428, 429) are shown as ALTERNATE. One of these links might be sending the ring acceleration shoulder-tap.
4. The status of system SYS3 is QUIESCED. It is not part of the ring and is not processing global resource requests. YES appears in the COMM field for system SYS3, indicating that system SYS3 responded to the request for status. MVS is still active on system SYS3.
5. Because system SYS3 is quiesced, all of its links to system SYS2 (220, 221, 420, 421) are marked as ALTERNATE.
6. The status of system SYS4 is QUIESCED. Like system SYS3, it is not part of the ring and is not processing global resource requests. NO appears in the COMM field for system SYS4, indicating that system SYS4 (unlike system SYS3) did not respond to the request for status. Also, all links between system SYS4 and system SYS2 (224, 225, 424, 425) are marked as QUIET. System SYS4 is the source of the problem.

   D GRS can report a problem but it cannot diagnose the reason for the problem. Possible reasons for the problem shown in this example are:

   a. System SYS4 is temporarily stopped. Perhaps the system has stopped to take a dump, or MVS might be in a spin loop.
   b. System SYS4 has failed.
   c. All links have failed on the system SYS4 side. This possibility is unlikely; an I/O error on a link is normally detected by both systems (SYS2 and SYS4 in this case), and the status of a failed link normally appears as DISABLED.

Figure 4-3. Using D GRS to Analyze a Problem

Quiescing a System

In the context of ring processing, quiescing a system means removing it from the ring. The system can continue to run, but it cannot access or free global resources. Quiescing a system is normally done as the first step in removing a system from the complex.

To quiesce a system, the operator on the system to be quiesced (or on any active system that can communicate with the system to be quiesced) can issue the VARY GRS(sysname),QUIESCE command. A quiesced system is no longer part of the ring; it is, however, still known to the other systems in the ring.

Quiescing a system can slow down performance because no global resources are released:

- On the quiesced system, any task that controls any global resources retains control of those resources, and any task that is waiting for a global resource continues to wait.

  Programs on the quiesced system do continue to process, but only until they need to access or free a global resource. For example, a program that had exclusive control of a global resource can finish with the resource; however, the quiesced system cannot completely process the DEQ for the resource or tell the active systems that the resource is now available.

- On the active systems, any task that needs a global resource held by a task on the quiesced system continues to wait. This condition continues until the quiesced system either rejoins the ring or is purged from the ring.

Thus, quiescing a system is an action that you should take very seldom and for as short a time as possible. It might, for example, be part of the process of physical reconfiguration. When it is necessary to quiesce a system, the operator must first bring work on the system to an orderly shutdown. This procedure minimizes ring performance problems and data integrity exposures if the system is to be purged from the ring.

Global resource serialization, in response to the VARY GRS(sysname),QUIESCE command, places the target system in a quiesced state and forms a new ring without the quiesced system. The operators can use the global resource serialization messages (ISG011I and ISG013I) and, if necessary, D GRS, to verify that the system is now quiesced.

Global resource serialization can build a new ring without the quiesced system only when the links needed for the new ring are available. When the complex is fully-connected and alternate links are available, rebuilding the ring without the quiesced system is not a problem. If a link that it needs to build a new ring without the quiesced system is missing, global resource serialization rejects the VARY GRS(sysname),QUIESCE command. After enabling the required links, the operator can enter the command again. Once the system is quiesced, it can either rejoin the ring, which does not require a re-IPL, or be purged from the ring.

Quiesce Messages

The global resource serialization messages related to quiescing a system include ISG011I, ISG012I, ISG013I, ISG014I, and ISG015I.

Example: Quiescing a System

Using the four-system complex shown earlier in Figure 4-1 on page 4-2, assume that it is necessary to stop SYS4 temporarily.

On SYS4, the operator would take the following steps:

1. Bring the work to an orderly shutdown by taking such actions as stopping the subsystems and terminating jobs.
2. Issue VARY GRS(*),QUIESCE or VARY GRS(SYS4),QUIESCE. (An operator on any active system could also issue the second command.)

   On SYS4, the following message appears:

   ```
   ISG012I QUIESCE REQUEST PASSED TO SYSTEM SYS2
   ```

   This message indicates that global resource serialization has accepted the command and that SYS2 will assist in the quiesce process. SYS2 is the assisting system, and SYS4 is the target system. (This message appears only when the operator on the target system issues the VARY command.)

   On SYS2 and SYS4, the following message indicates that global resource serialization has started to quiesce the target system (SYS4):

   ```
   ISG011I SYSTEM SYS4 - QUIESCING GLOBAL RESOURCE SERIALIZATION
   ```

   On all systems, the following message appears after the system has been successfully quiesced.

   ```
   ISG013I SYSTEM SYS4 - QUIESCED GLOBAL RESOURCE SERIALIZATION
   ```

3. Once the quiesce is successful, the operator on SYS4 can stop the system.

Issuing D GRS would show the state of SYS4 as QUIESCED. The COMM field would show YES, indicating that communication still existed, and all links to SYS4 would appear as ALTERNATE.

Purging a System

To purge a system, the operator on any active system issues the VARY GRS(sysname),PURGE command. Purging a system is normally needed when:

- The operator must remove the system from the ring for a long period of time (perhaps for preventive maintenance).
- The system is no longer needed in the ring (perhaps because of a configuration change).
- The system has failed and must re-IPL.

In response to the purge command, each active system in the ring deletes all information related to the target system, including its requests for global resources, its control of global resources, and any appearance of its system name. In short, global resource serialization removes all indications that the purged system was ever a part of the complex.

If users on the purged system held global resources, these resources are freed. Message ISG018I, issued to SYSLOG, describes these resources. Your installation must plan in advance to investigate the state of resources as part of the process of removing a system from the complex.

The purged system must re-IPL with GRS=JOIN to rejoin the ring. Also, a system that has been part of an active ring cannot re-IPL with GRS=JOIN unless it has first been purged. Until the system is purged, global resource serialization knows about it and rejects its attempt to rejoin the ring because it detects a duplicate system name.

Because purging the system does not stop MVS, stop the system after purging it so that the system cannot continue to access shared resources. This procedure prevents a potential data integrity exposure. For example, assume that a job on a purged system, SYS1, was updating a resource and did not complete before SYS1 was purged. Purging SYS1 frees the resource and makes it available to other requestors, but, unless SYS1 is stopped, the job on SYS1 can continue to update the resource.

With MVS/SP Version 3, the target system — the system to be purged — can be an active system or a quiesced system. (On earlier levels of MVS, the target system must be a quiesced system.) When the target system is an active system, global resource serialization issues a message to remind the operator that the system is active; that is, the operator must bring the work to an orderly shutdown before proceeding. See "Example: Purging an Active System" for an example of purging an active system. See "Example: Purging a Quiesced System" on page 4-12 for an example of purging a quiesced system.

Purge Messages

The global resource serialization messages related to purging a system include ISG011I, ISG013I, ISG014I, ISG015I, ISG016I, ISG017D, ISG018I, ISG100E, ISG101D, ISG102I, ISG103I, ISG104I, and ISG106I.

Example: Purging an Active System

Using the four-system complex shown earlier in Figure 4-1 on page 4-2, assume that it is necessary to purge SYS1 from the ring. SYS1 is an active system. The operator on SYS2 issues the purge command; thus, SYS2 is the assisting system. The required steps are:

On SYS1, the operator, to avoid potential data integrity exposures, must bring the workload to an orderly shutdown by taking such actions as stopping the subsystems and terminating jobs.

On SYS2, the operator then issues VARY GRS(SYS1),PURGE. The following messages appear:

```
ISG100E SYSTEM SYS1 IS STILL AN ACTIVE GRS SYSTEM
ISG101D CONFIRM PURGE FOR ACTIVE SYSTEM SYS1 - REPLY YES OR NO
```

In this example, the operator can safely reply YES. These messages remind the operator that SYS1 is still active; purging it from the ring might create a data integrity exposure.

On SYS2, the operator replies YES to message ISG101D.

On all active systems, the following messages appear:

```
ISG011I SYSTEM SYS1 - QUIESCING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS1 - QUIESCED GLOBAL RESOURCE SERIALIZATION
ISG011I SYSTEM SYS1 - BEING PURGED FROM GRS COMPLEX
ISG013I SYSTEM SYS1 - PURGED FROM GRS COMPLEX
```

On SYS1, the operator must stop the system. At this point, SYS1 is no longer known to global resource serialization. The ring consists of SYS2, SYS3, and SYS4. To rejoin the ring, SYS1 must re-IPL with GRS=JOIN.

While purging SYS1, global resource serialization might detect a potential data integrity exposure. After purging an active system, the operator should follow the installation's procedures for resolving a data integrity exposure, such as contacting the system programmer responsible for investigating the state of resources. More information about the problem normally appears in SYSLOG, where the following message appears to describe any resources that might have been damaged by purging the system:

```
ISG018I REQUESTORS FROM SYSTEM SYS1 HAVE BEEN PURGED FROM RESOURCES
        NAMED xxxx,yyyy
```

Example: Purging a Quiesced System

Using the four-system complex shown earlier in Figure 4-1 on page 4-2, assume that it is necessary to purge SYS1 from the ring. SYS1 is a quiesced system. The operator on SYS2 issues the purge command; thus, SYS2 is the assisting system. The required steps are:

On SYS2, the operator issues VARY GRS(SYS1), PURGE. The following messages appear:

```
ISG016I SYSTEM SYS1 OWNS OR IS WAITING FOR GLOBAL RESOURCES
ISG017D CONFIRM PURGE REQUEST FOR SYSTEM SYS1 - REPLY YES OR NO
```

These messages indicate that users on SYS1 still own, or are waiting for, global resources. The operator on SYS2 should reply NO unless the operator knows the work on SYS1 has been shut down. The operator on SYS1 must shut the work down. Replying YES to message ISG017D might create a data integrity exposure. When the work on SYS1 has been shut down, the operator on SYS2 can reissue the VARY GRS(SYS1), PURGE command.

On SYS2, the following message appears:

```
ISG011I SYSTEM SYS1 - BEING PURGED FROM GRS COMPLEX
```

On all active systems, the following message appears:

```
ISG013I SYSTEM SYS1 - PURGED FROM GRS COMPLEX
```

On SYS1, the operator must stop the system. At this point, SYS1 is no longer known to global resource serialization. The ring consists of SYS2, SYS3, and SYS4. To rejoin the ring, SYS1 must re-IPL with GRS=JOIN.

While purging SYS1, global resource serialization detected a potential data integrity exposure, indicated by message ISG016I. When this message appears, the operator should follow the installation's procedures for resolving the problem, such as contacting the system programmer responsible for investigating the state of resources. More information about the problem normally appears in SYSLOG, where the following message appears to describe any resources that might have been damaged by purging the system:

```
ISG018I REQUESTORS FROM SYSTEM SYS1 HAVE BEEN PURGED FROM RESOURCES
        NAMED xxxx,yyyy
```

Restarting a System

Specifying the automatic recovery options, RESTART and REJOIN, means that operators seldom need to intervene to restart a system. When necessary, the operator on any system in the complex can issue the VARY GRS(sysname),RESTART command to bring the target system back into the ring. The target system can be either an inactive system or a quiesced system.

Restarting a system is the opposite of quiescing a system. VARY GRS(sysname),QUIESCE suspends global resource serialization and removes the target system from the ring. VARY GRS(sysname),RESTART resumes global resource serialization and brings the system back into the ring. The restarted system, because it is now part of the ring, can release any resources already freed by users and resume processing requests for global resources.

The VARY GRS(sysname),RESTART command can bring back into the ring a system that the operator had quiesced or a system that had become inactive or quiesced as a result of a ring disruption. After issuing the command, the operator can use D GRS to verify that the target system is now part of the ring.

There are three ways to issue the command:

1. VARY GRS(sysname),RESTART — issued from any active system to bring the named system back into the ring.
2. VARY GRS(*),RESTART — issued on the system to be restarted to bring that system back into the ring.

Issuing the first or second form of the command when all systems are inactive makes the specified system active, while all others become quiesced. The active system can then bring the other systems back into the ring.

3. VARY GRS(ALL),RESTART — issued on any inactive system to restart the ring. Ensure that operators:

   - Allow automatic recovery processing to complete before issuing this command.
   - Issue this command only when there is no active system in the ring and at least one inactive system.
   - Issue this command only once during the process of recovery from a ring disruption.

   The command changes the state of all inactive systems from inactive to active. The command can bring back into the ring all systems quiesced by a ring disruption but does not bring back a system that the operator quiesced specifically.

Issuing any form of the command when all systems are quiesced invokes the reactivate function, which is designed for very unusual recovery situations. "Reactivating a System" on page 4-29 describes the process of reactivating a quiesced system when there is no active or inactive system in the ring.

To avoid the possibility of split rings, ensure that the operators issue VARY GRS(ALL),RESTART with extreme care.

Restart Messages

Global resource serialization messages related to restarting a system include ISG011I, ISG012I, ISG013I, ISG014I, ISG015I, ISG022E, ISG024I, ISG025E, ISG026I, ISG027D, ISG080E, ISG082D, ISG119I, ISG120I, ISG121I, ISG122I, and ISG123I.

Split Rings

Split rings can occur when more than one operator tries to restart the ring at the same time, causing the ring to split into multiple independent rings, each able to grant access to global resources at the same time. Split rings create a severe data integrity exposure. Actions you can take to avoid split rings include:

- Ensure that an operator issues VARY GRS(ALL),RESTART only from an inactive system.
- Ensure that only one operator issues VARY GRS(ALL), RESTART
- Provide alternate links.
- Specify RESTART(YES) whenever possible, which allows automatic restart and reduces operator intervention in recovery from a ring disruption.

An operator trying to restart the ring should always issue VARY GRS(ALL),RESTART to restart all of the systems rather than VARY GRS(sysname),RESTART or VARY GRS(*),RESTART to restart a specific system. When the operator issues VARY GRS(ALL),RESTART, global resource serialization issues the following messages:

```
ISG026I SYSTEM SYS2 MAY CREATE A SPLIT RING IF ANY OTHER GRS SYSTEM
        IS ACTIVE. VERIFY THAT NO GRS SYSTEM IS ACTIVE BEFORE
        CONFIRMING RESTART
ISG027D CONFIRM RESTART RING FOR SYSTEM SYS2 - REPLY NO OR YES
```

Before replying to message ISG027D, the operator must issue D GRS and/or check with the other operators to verify that there are no active systems. If all other systems are inactive, the operator can safely reply YES to continue the restart. If any system is active, the operator must reply NO to avoid split rings.

For example, consider the two-system complex shown in Figure 4-4. SYS1 is active, SYS2 is quiesced, and the communication link has failed. If the operator issues a restart command on SYS2, message ISG026I appears to warn the operator that split rings might occur, followed by a prompting message. If the operator replies YES to the prompt, SYS2 will create a ring of one system. Because SYS1 is also active and there is no communication, there are two one-system rings. Both rings can grant access to the same global resources, and neither system can rejoin the ring created by the other without an IPL. Note that global resource serialization does not force the re-IPL; it is, however, required to resolve the data integrity exposure.

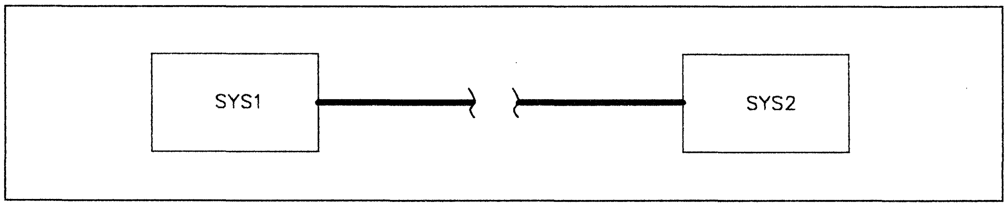


Figure 4-4. Two-System Ring with Link Failure

There are several ways to avoid split rings in this situation:

1. If the configuration includes an alternate link, and the alternate link has not failed, the problem does not occur; global resource serialization could use the alternate link and resume processing almost immediately.
2. If the operator issues the restart command on the active system, split rings do not occur. Instead, the following message appears:

   ```
   ISG014I VARY GRS RESTART REQUEST FOR SYSTEM SYS1 REJECTED -
           SYSTEM NOT RESPONDING
   ```
3. If the operator replies NO to the prompt following message ISG027D, split rings do not occur.

In the last two cases, message ISG026I or message ISG014I alert the operator to the actual problem; SYS1 and SYS2 cannot communicate. The operator could then respond correctly — fix the link problem, then reissue the VARY GRS(SYS2),RESTART command for the quiesced system.

Example: Restarting a System

Using the four-system complex shown in Figure 4-1 on page 4-2, assume that SYS4 has been quiesced but is now ready to rejoin the ring. An operator on any active system could restart SYS4, but assume that the operator on SYS2 is to handle the restart.

On SYS2, the operator issues VARY GRS(SYS4),RESTART.

On SYS2 and SYS4, the following message appears to indicate that the process of restarting SYS4 has begun:

```
ISG011I SYSTEM SYS4 - RESTARTING GLOBAL RESOURCE SERIALIZATION
```

On all active systems, the following message appears to indicate that SYS4 is now part of the active global resource serialization ring:

```
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

If the operator on SYS4 issued the restart command, global resource serialization would pass the command to an active system for processing. If the system selected was SYS2, the following message would appear on SYS4:

```
ISG012I RESTART REQUEST PASSED TO SYSTEM SYS2
```

Controlling CTC Links

To bring a CTC link online or take a CTC link offline, use the VARY command. All links that you want global resource serialization to use must be defined in the GRSCNFxx member.

To bring a defined link online, issue VARY devnum,ONLINE. In response to this command, the link comes online, and global resource serialization changes its status from DISABLED to ALTERNATE. It is then available to send the ring acceleration signal or to act as a backup for a primary link.

To take a defined link offline, use the following procedure:

1. Issue VARY devnum,OFFLINE.
2. Issue S DEALLOC or run an installation-supplied deallocation procedure to take the link completely offline. If you omit this step, the link remains as "pending offline" and global resource serialization continues to try to use it.

As a result of this procedure, the link goes offline, and global resource serialization changes its status from ALTERNATE to DISABLED. The specified link must be an alternate link; if an operator tries to take the primary link offline, global resource serialization rejects the command.

Link Messages

The global resource serialization messages related to controlling links include ISG022E, ISG046E, ISG047I, ISG048I, and ISG083E.

Recovery Operations

During normal operations, the RSA-message passes from one system in the ring to another. Problems that prevent the RSA-message from passing around the ring cause a ring disruption. During a ring disruption, the processing of global resource requests is suspended on all of the systems. MVS might still be running on all of the systems, but no system can process a global resource request.

Either a link failure or a system failure can disrupt the ring. Global resource serialization detects a link failure almost immediately. How quickly it detects a system failure depends, to a large extent, on the TOLINT value set in GRSCNFxx, as described in "Tolerance Interval (TOLINT)" on page 3-11. Recovery from a system failure might require the failed system to re-IPL.

In planning your specific recovery procedures for the operators of the systems in a global resource serialization ring, you must thus consider the following types of failures:

- IPL problems. IPL problems related to ring processing show up as the operators IPL the systems that are to form the ring or as the operator of a system that has failed tries to re-IPL and rejoin the ring. See "IPL Problems" on page 4-17.
- Primary link problems. A primary link problem occurs when a CTC link fails while global resource serialization is using the link to send the RSA-message. See "Primary Link Failure" on page 4-20.
- Alternate link problems. An alternate link problem occurs when a CTC link fails while global resource serialization is using the link to send the ring acceleration signal. An alternate link failure does not always disrupt ring processing. See "Alternate Link Failure" on page 4-22.
- System failures. When a system fails because of either a hardware or a software error, global resource serialization tries to rebuild the ring without the failed system. See "System Failure" on page 4-22.

How successfully the ring can recover from any failure that disrupts ring processing depends, first of all, on the design of the complex. Successful recovery from ring disruptions means that global resource serialization can rebuild a valid ring without the failed element. Successful recovery from a system failure is a ring that consists of all systems except the failed system. Successful recovery from a link failure is a ring that consists of all systems; successful recovery thus requires a fully-connected complex with alternate links. A backup 3088 provides another level of safety and eliminates the 3088 as a single point of failure.

You must also decide how much operator intervention you want in the ring recovery process. As a general rule, minimize operator intervention as much as possible. See "Automatically Rebuilding a Disrupted Ring (RESTART)" on page 3-15 and "Automatically Rejoining the Ring (REJOIN)" on page 3-15 for details. Specifying RESTART(YES) and REJOIN(YES) whenever possible allows the ring to recover automatically from most disruptions and minimizes operator intervention in the recovery process.

IPL Problems

Careful planning and testing can minimize or eliminate many potential IPL errors. For example:

- Ensure that, if necessary, the operators check the shared DASD connections and the link connections before building the complex.
- IPL each system explicitly with GRS=START or GRS=JOIN. Specify the SYSNAME system parameter in IEASYSxx.
- Test GRSCNFxx carefully to ensure that it does not contain syntax errors or incorrect CTC link device numbers. To minimize the testing, create a single GRSCNFxx member that describes the entire complex, test it, then copy it to the parmlibs of all systems. Do not use the MATCHSYS(*) syntax in GRSCNFxx. Make sure that GRSCNFxx exists in any backup parmlib that your installation maintains in case the system cannot read all or part of the parmlib.
- Ensure that the RNLs are correct and identical on all systems. Test GRSRNLxx carefully to ensure that it does not include syntax errors. In SYS1.SAMPLIB, IBM supplies an RNL syntax checker utility. Member ISGRNLCK of SYS1.SAMPLIB contains information about using the RNL syntax checker utility.

  To minimize the testing, create a single GRSRNLxx member, test it, then copy it to the parmlibs of all systems. Make sure that GRSRNLxx exists in any backup parmlib that your installation maintains in case the system cannot read all or part of the parmlib.
- Ensure that your operators understand that a failed system — a system that was part of the ring but has failed and requires a re-IPL — must be purged from the ring before a re-IPL can complete. See "Example 1: IPL Failure — System Not Purged" on page 4-18.

It is, of course, not possible to eliminate all potential IPL-time errors. For example, if a system tries to join an existing complex, but the existing complex is recovering from a ring disruption, or the system controlling join processing fails, the operator of the joining system must wait until the existing complex is active before the system can IPL with GRS=JOIN. See "Example 2: IPL Failure — Disruption in Progress" on page 4-19. You need to plan the operator action in such a situation. You must decide in advance whether, for example, the operator is to hold up the IPL until the problem is solved or IPL with GRS=NONE and join the complex at a later IPL with GRS=JOIN.

Note that using GRS=NONE might look like an effective short-term solution to problems with building the complex, but use it only with great care. Using GRS=NONE means that the system issues all reserves. If a system that is normally part of the complex IPLs with GRS=NONE while it has connections to shared DASD volumes in common with systems in the complex, a data integrity exposure can occur. If the complex becomes active and starts to suppress reserves for resources on those shared volumes, the integrity of those resources might be endangered.

The following examples, based on the sample configuration shown in Figure 4-1 on page 4-2, show problems that can occur during IPL and ways the operator can resolve them.

Example 1: IPL Failure — System Not Purged

SYS4 was part of an active ring but has failed and requires a re-IPL. Global resource serialization has automatically recovered from the system failure by rebuilding a three-system ring. Figure 4-5 shows the current situation. The operator on SYS4 started to re-IPL but did not first purge SYS4 from the ring.

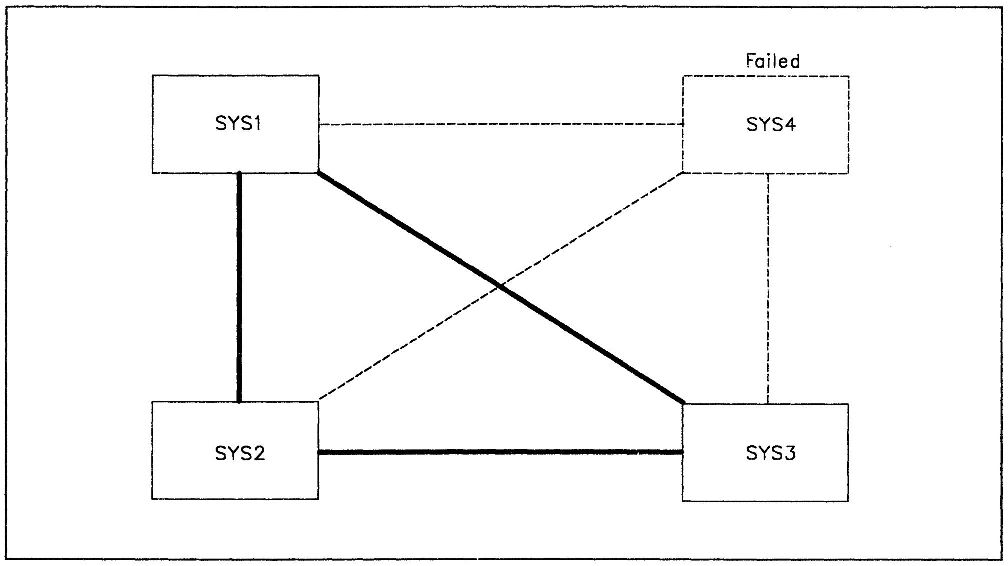


Figure 4-5. IPL Error — Example 1

On SYS4, the following messages appear:

```
ISG006I GRS JOIN OPTION INVALID - SYSTEM SYS4 ALREADY A GRS SYSTEM
```

This message indicates that the system name (SYS4) is already known in the global resource serialization complex. It generally means that the operator did not purge a failed system from the ring before starting to reload the system; there appear to be two systems in the ring with the same name. (This message also appears when the operator types the wrong system name during IPL or specifies the wrong IEASYSxx parmlib member).

```
ISG007I fc-rc ERROR PROCESSING GRS JOIN
ISG009D RELOAD THE SYSTEM OR REPLY JOIN OR NONE
```

For an obvious typing error, the operator should restart the IPL and specify the system name correctly.

In all other cases, recovery requires the following steps:

1. On an active system, issue D GRS to check the state of the systems in the ring. In this case, the display shows that SYS4 is QUIESCED and that all its links are QUIET. The following text describes the condition more specifically:

   ```
   SYSTEM SYS4 IS ATTEMPTING TO JOIN, BUT ALREADY IN COMPLEX
   ```

2. On an active system, issue VARY GRS(SYS4),PURGE to remove SYS4 from the ring.
3. On SYS4, continue with the IPL after the purge command completes successfully. Reply JOIN to message ISG009D.

The IPL continues. Message ISG011I appears on all systems to indicate that SYS4 has successfully joined the complex.

To avoid this problem, ensure that recovery procedures for a problem that involves a re-IPL require an operator to purge the system from the ring before reloading the system.

Example 2: IPL Failure — Disruption in Progress

SYS4 has failed and been purged from the ring. It requires a re-IPL to resume processing and rejoin the ring. The operator specified GRS=JOIN, but, while SYS4 was joining the ring, the active ring (systems SYS1, SYS2, and SYS3) has experienced a ring disruption.

In this example, assume that SYS2 is the system selected to initiate ring recovery procedures. To show the importance of using automatic recovery, this example also assumes that no system is allowed to restart the ring automatically; that is, RESTART(NO) is in effect on all systems. Figure 4-6 shows the current situation.

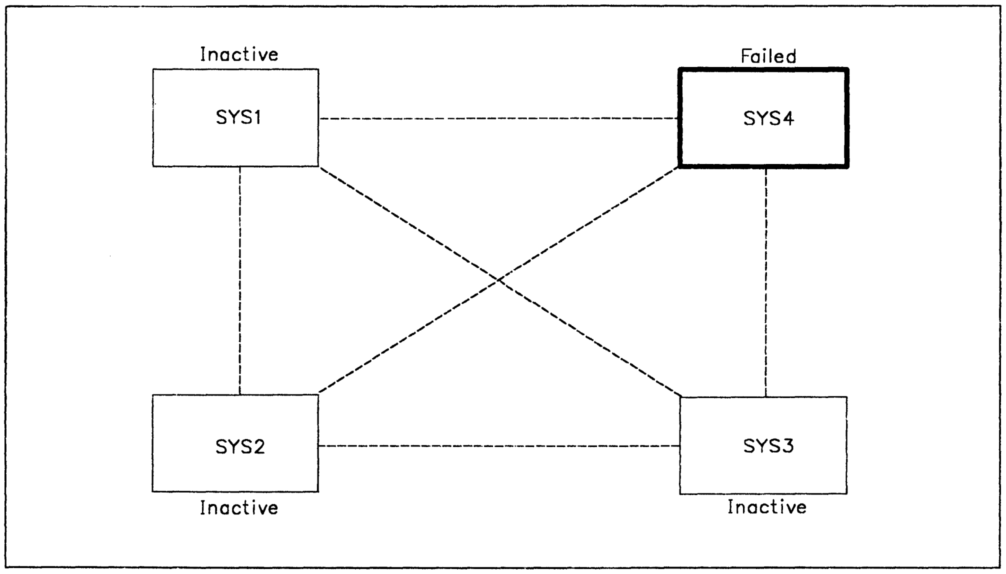


Figure 4-6. IPL Error — Example 2

On SYS4, the following messages appear:

```
ISG006I GRS JOIN OPTION INVALID - GRS DISRUPTION MAY BE IN PROGRESS
```

SYS4 tried to join the ring, but all systems in the ring are inactive. SYS4 cannot be brought into the ring.

```
ISG007I fc-rc ERROR PROCESSING GRS JOIN
ISG009D RELOAD THE SYSTEM OR REPLY START OR JOIN
```

On other systems, message ISG023E indicates that a ring disruption has occurred and message ISG025E indicates that automatic restart is not possible.

Recovery requires the following steps:

1. On SYS2, issue D GRS to check the state of the systems in the ring. In this case, the display shows that SYS1, SYS2, and SYS3 are all INACTIVE or QUIESCED but that all systems can communicate. The status of SYS4 is JOINING.
2. On SYS2, issue VARY GRS(ALL),RESTART to restart the ring that consists of SYS1, SYS2, and SYS3.

3. On SYS2 after the ring is rebuilt, issue D GRS to verify that there is an active ring.
4. On SYS4, continue with the IPL. Reply JOIN to message ISG009D.

Specifying RESTART(YES) in GRSCNFxx, which is the recommended action, eliminates most operator intervention in recovery from ring disruptions. If RESTART(YES) had been specified for all systems in this example, recovery would require only the last step.

Primary Link Failure

The primary link is the link global resource serialization is using to send the RSA-message. Issuing D GRS shows IN USE for the primary link. All other links appear as ALTERNATE, QUIET, or DISABLED.

A system in the ring detects a primary link failure when the link it is using to send the RSA-message receives unanticipated interrupts, cannot perform an I/O operation, or receives unanticipated messages. The cause of the failure is not necessarily the link itself; the cause might also be a channel or software error.

Global resource serialization detects a primary link failure almost immediately. It does not wait for the tolerance interval to expire. On the system that detected the link failure, global resource serialization reports the problem to the operator as a link failure and a ring disruption. Other systems in the ring report only the ring disruption; they are aware that there is a problem but not aware of the cause.

If a primary link fails, global resource serialization automatically selects an alternate link, if one is available. If there is no alternate available, global resource serialization tries to rebuild a valid ring using the connections that are available. A fully-connected complex with alternate links can always recover from the loss of a single link. If RESTART(YES) is specified, it can usually recover without operator intervention.

To recover the disrupted ring, global resource serialization varies the failed link offline and selects an alternate link to replace the failed primary link. If there is only one alternate and it is being used to send the ring acceleration signal, global resource serialization no longer sends the ring acceleration signal between the two affected systems. If there are two alternate links available, one becomes the primary link and the other is used to send the acceleration signal.

Issuing D GRS after ring processing resumes will show the links that are now IN USE. (Note that D GRS does not identify a link used for ring acceleration; if you need this information, use the RMF Monitor I report on communication equipment activity.)

Example: Primary Link Failure

Using the sample configuration shown in Figure 4-1 on page 4-2, assume that the primary link between SYS1 and SYS2 fails. SYS1 detects the error on the link. The device number of the failed link is A01. The installation has specified RESTART(YES) for all systems.

On SYS1, the following messages appear:

```
ISG046E CTC A01 DISABLED DUE TO HARDWARE ERROR FAILURE CODE=05
ISG047I CTC A01 DISABLED
ISG022E SYSTEM SYS1 DISRUPTED GLOBAL RESOURCE SERIALIZATION DUE TO
COMMUNICATION FAILURE - GLOBAL RESOURCE REQUESTORS WILL BE SUSPENDED
```

These messages indicate that SYS1 was using link A01 as the primary link, but it failed because of a hardware error. The operator should contact hardware support personnel to fix the problem. Global resource serialization disables the link.

On other systems, the following message appears:

```
ISG023E GLOBAL RESOURCE SERIALIZATION HAS BEEN DISRUPTED - GLOBAL
RESOURCE REQUESTORS WILL BE SUSPENDED
```

Whether this message or ISG022E appears on the other systems depends on how each system detected the error. This message appears if the system detected the error because the tolerance interval expired or because SYS1 began recovery from the ring disruption. Message ISG022E appears if the system detected the communication failure. In most cases, message ISG023E appears. Make sure that the operators understand that these messages simply reflect the different points at which the systems detected the error.

On SYS1, the system coordinating the ring recovery, the following message indicates that automatic restart has begun:

```
ISG024I SYSTEM SYS1 INITIATED AUTO RESTART PROCESSING
```

After a ring disruption, all systems defined with RESTART(YES) attempt to restart the ring. Only one actually coordinates the process; this message identifies that system, which will bring all of the others back into the ring.

On other systems (SYS2, SYS3, and SYS4 in this example), the following message appears:

```
ISG025E SYSTEM sysx UNABLE TO INITIATE AUTORESTART PROCESSING -
PERMISSION GRANTED TO SYS1
```

This message, where *sysx* is SYS2, SYS3, or SYS4, indicates that this system has given permission to SYS1 to rebuild the ring. It is a message that the operators should expect to see during automatic restart processing. It normally requires no operator action.

As SYS1 rebuilds the ring, assuming that SYS2, SYS3, and SYS4 rejoin the ring in that order, the following messages appear:

On SYS1:

```
ISG011I SYSTEM SYS1 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS1 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG011I SYSTEM SYS2 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS2 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG011I SYSTEM SYS3 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS3 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG011I SYSTEM SYS4 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

On SYS2:

```
ISG011I SYSTEM SYS2 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS2 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS3 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

On SYS3:

```
ISG011I SYSTEM SYS3 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS3 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

On SYS4:

```
ISG011I SYSTEM SYS4 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

Alternate Link Failure

An alternate link failure does not usually disrupt ring processing. If global resource serialization detects an alternate link failure, it issues messages ISG046E and ISG047I to describe the error. If the error was a hardware error, the operator should contact hardware support personnel to repair the problem. When the link is repaired, the operator can issue a VARY devnum,ONLINE command to make the link available again. If the error was a software error, the operator should follow installation procedures to deal with such problems and might need to IPL the system again at the earliest convenient time.

If global resource serialization was using the alternate link to send the ring acceleration shoulder-tap signal, it tries to locate another alternate link to use for ring acceleration. If one is available, it is used to send the signal. If another alternate link is not available, the ring acceleration signal is no longer sent between the two systems. The following message appears:

```
ISG083E GRS CANNOT SEND ACK-TAP SIGNALS TO SYSTEM sysname
```

Ring processing continues normally, but throughput might slow down on one of the systems using the link because all requests for global resources must wait for ring processing. If an alternate link becomes available, global resource serialization begins using it to send the ring acceleration signal and deletes message ISG083E.

System Failure

A system in the ring might fail because of a hardware or software error. The failure might be either a failure of relatively long duration or a temporary failure. Some of the conditions that can cause a temporary failure are:

- The system is in a spin loop.
- SDUMP has set the system nondispatchable to dump virtual storage.
- The operator stopped the system.

Global resource serialization detects the system failure when the tolerance interval expires. The system waiting to receive the RSA-message detects the problem. Regardless of the cause or duration of the failure, global resource serialization recovery is the same.

The first step in recovery from a system failure is rebuilding a ring that consists of all of the systems except the failed system. Thus, successful recovery requires a fully-connected complex with alternate links. If the systems are not fully connected, global resource serialization cannot always rebuild a ring that consists of all systems except the failed system.

Your installation controls the amount of operator intervention required to rebuild the ring:

1. Specify RESTART(YES) in GRSCNFxx to allow the system to automatically restart the ring. See "Example: Automatic Restart." for an example. IBM recommends that you specify RESTART(YES).
2. Specify RESTART(NO) to prevent the system from automatically restarting the ring. RESTART(NO) means an operator must intervene to restart the ring manually. See "Example: Manual Restart" on page 4-25.

The second step in recovery from a system failure is to bring the failed system back into the ring. "Example: Automatic Restart" and "Example: Manual Restart" describe the actions. The specific actions required depend on whether the system failed because of a problem that requires a re-IPL or because it was temporarily stopped. For a system failure that requires a re-IPL, operator intervention is required to purge the system before the re-IPL. For a temporary failure, your installation controls whether or not operator intervention is required to bring the stopped system back into the ring:

- Specify REJOIN(YES) to bring the system back automatically. IBM recommends that you specify REJOIN(YES).
- Specify REJOIN(NO) if you want the operator to bring the system back into the ring.

For complexes of three or more systems, specify RESTART(YES) to allow all systems to restart the ring automatically and REJOIN(YES) to allow all systems to rejoin the ring automatically. If there is a reason why you do not want all systems to restart the ring automatically, it is still good practice to specify REJOIN(YES) on all systems.

The recovery considerations for a two-system complex are slightly different. To avoid the possibility of split rings, described earlier under "Split Rings" on page 4-14, specify RESTART(YES) and REJOIN(YES) on only one system. Choose the more important of the two systems, or choose one arbitrarily. Specify RESTART(NO) and REJOIN(YES) on the other system. Following this recommendation means that the specific recovery procedure depends on which system fails. See "Example: Two-System Complex" on page 4-27.

Example: Automatic Restart

Using the fully-connected four-system complex shown in Figure 4-1 on page 4-2, assume that SYS1 has failed and that SYS2 detected the failure. Assume also that RESTART(YES) and REJOIN(YES) were specified for all systems.

<u>On SYS2</u>, the following message appears:

```
ISG022E SYSTEM SYS1 DISRUPTED GLOBAL RESOURCE SERIALIZATION DUE TO
        SOFTWARE FAILURE - GLOBAL RESOURCE REQUESTORS WILL BE SUSPENDED
```

On the other systems, the following message appears:

```
ISG023E GLOBAL RESOURCE SERIALIZATION HAS BEEN DISRUPTED - GLOBAL
        RESOURCE REQUESTORS WILL BE SUSPENDED
```

This message indicates that an error has disrupted ring processing, though it does not pinpoint the source of the error. All active systems become inactive; any task that tries to obtain or free a global resource is suspended. This message also appears on all of the other systems in the ring. All systems then attempt to restart the ring but only one actually coordinates the process. In this example, assume that SYS2 is the system that will recover the ring.

On SYS2, the system coordinating the ring recovery, the following message appears to indicate that automatic restart has begun:

```
ISG024I SYSTEM SYS2 INITIATED AUTO RESTART PROCESSING
```

On the other systems (SYS3 and SYS4 in this example), the following message appears:

```
ISG025E SYSTEM sysx UNABLE TO INITIATE AUTORESTART PROCESSING -
        PERMISSION GRANTED TO SYS2
```

This message, where *sysx* is SYS3 or SYS4, indicates that this system has given permission to SYS2 to rebuild the ring. It is a message that the operators should expect to see during automatic restart processing. It normally requires no operator action.

As SYS2 rebuilds the ring, assuming that SYS3 and SYS4 rejoin the ring in that order, the following messages appear as each system rejoins the ring.

On SYS2:

```
ISG011I SYSTEM SYS2 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS2 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG011I SYSTEM SYS3 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS3 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG011I SYSTEM SYS4 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

On SYS3:

```
ISG011I SYSTEM SYS3 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS3 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

On SYS4:

```
ISG011I SYSTEM SYS4 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

Global resource serialization has recovered the ring automatically without operator intervention. The recovered ring consists of the SYS2, SYS3, and SYS4. The operator on an active system must determine if the failure on SYS1 requires SYS1 to re-IPL or if the failure on SYS1 is occurred because SYS1 had stopped temporarily.

If the failure is temporary, no further action is needed. When SYS1 resumes processing, it automatically rejoins the ring because REJOIN(YES) was specified for all systems. The following messages appear on all active systems:

```
ISG011I SYSTEM SYS1 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS1 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

If the failure requires SYS1 to re-IPL, recovering the failed system requires the following steps:

1. On an active system, issue VARY GRS(SYS1),PURGE to remove SYS1 from the ring. The following messages might appear:

   ```
   ISG016I SYSTEM SYS1 OWNS OR IS WAITING FOR GLOBAL RESOURCES
   ISG017D CONFIRM PURGE REQUEST FOR SYSTEM SYS1 - REPLY YES OR NO
   ```

   These messages indicate a possible data integrity exposure. In this example, the operator replies YES.

   The following message appears:

   ```
   ISG011I SYSTEM SYS1 - BEING PURGED FROM GRS COMPLEX
   ```

2. On SYS1, perform problem determination, then re-IPL the system with GRS=JOIN.

3. On the active systems, the following messages appear when SYS1 rejoins the ring:

   ```
   ISG011I SYSTEM SYS1 - RESTARTING GLOBAL RESOURCE SERIALIZATION
   ISG013I SYSTEM SYS1 - RESTARTED GLOBAL RESOURCE SERIALIZATION
   ```

While purging SYS1, global resource serialization detected a potential data integrity exposure, indicated by message ISG016I. When this message appears, the operator should follow the installation's procedures for resolving the problem, such as contacting the system programmer responsible for investigating the state of resources. More information about the problem normally appears in SYSLOG, where the following message describes any resources that might have been damaged by purging the system:

```
ISG018I REQUESTORS FROM SYSTEM SYS1 HAVE BEEN PURGED FROM RESOURCES
        NAMED xxxx,yyyy
```

Example: Manual Restart

Using the fully-connected four-system complex shown in Figure 4-1 on page 4-2, assume that SYS1 has failed and that SYS2 detected the failure. Assume also that RESTART(NO) and REJOIN(NO) were specified for all systems.

On SYS2, the following messages appear:

```
ISG022E SYSTEM SYS1 DISRUPTED GLOBAL RESOURCE SERIALIZATION DUE TO
        SOFTWARE FAILURE - GLOBAL RESOURCE REQUESTORS WILL BE SUSPENDED
ISG025E SYSTEM SYS2 UNABLE TO INITIATE AUTORESTART PROCESSING -
        THIS SYSTEM IS NOT AUTHORIZED
```

These messages indicate that an error has disrupted ring processing, and that the system is not authorized to restart the ring automatically. All active systems become inactive; any task that tries to obtain or free a global resource is suspended. Message ISG023E and message ISG025E appear on all of the other systems in the ring. Tasks suspended while waiting for resources will quickly affect throughput on all systems; it is therefore imperative that an operator take quick action. If you must use manual restart, select the operator in advance. Ensure that the operator issues VARY GRS(ALL),RESTART rather than trying to restart specific systems and that the command is issued only once. These actions avoid the possibility of split rings, described earlier under "Split Rings" on page 4-14.

This example assumes the operator on SYS2 is to initiate ring recovery actions.

On SYS2, the operator issues VARY GRS(ALL),RESTART. See "Split Rings" on page 4-14 for a description of messages that might occur if global resource serialization detects the possibility of split rings.

As SYS2 rebuilds the ring, assuming that SYS3 and SYS4 rejoin the ring in that order, the following messages appear as each system rejoins the ring.

On SYS2:

```
ISG011I SYSTEM SYS2 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS2 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG011I SYSTEM SYS3 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS3 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG011I SYSTEM SYS4 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

On SYS3:

```
ISG011I SYSTEM SYS3 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS3 - RESTARTED GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

On SYS4:

```
ISG011I SYSTEM SYS4 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS4 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

On SYS2, the operator, after the restart is complete, can issue D GRS to verify that the systems are active. The operator must determine if the failure on SYS1 requires SYS1 to re-IPL or if the failure on SYS1 occurred because the operator stopped SYS1 temporarily.

If the failure is temporary, recovering the system requires the following steps:

1. On SYS1, the operator resumes system processing.
2. On SYS2, after determining that SYS1 has resumed processing, the operator issues VARY GRS(SYS1),RESTART.
3. On the active systems, the following messages appear when SYS1 rejoins the ring:

   ```
   ISG011I SYSTEM SYS1 - RESTARTING GLOBAL RESOURCE SERIALIZATION
   ISG013I SYSTEM SYS1 - RESTARTED GLOBAL RESOURCE SERIALIZATION
   ```

If the failure requires SYS1 to re-IPL, recovering the failed system requires the following steps:

1. On an active system, issue VARY GRS(SYS1),PURGE to remove SYS1 from the ring. The following messages might appear:

   ```
   ISG016I SYSTEM SYS1 OWNS OR IS WAITING FOR GLOBAL RESOURCES
   ISG017D CONFIRM PURGE REQUEST FOR SYSTEM SYS1 - REPLY YES OR NO
   ```

   These messages indicate a possible data integrity exposure. In this example, the operator can reply YES.

   The following message appears:

   ```
   ISG011I SYSTEM SYS1 - BEING PURGED FROM GRS COMPLEX
   ```

2. On SYS1, perform problem determination, then re-IPL the system with GRS=JOIN.

3. On the active systems, the following messages appear when SYS1 rejoins the ring:

   ```
   ISG011I SYSTEM SYS1 - RESTARTING GLOBAL RESOURCE SERIALIZATION
   ISG013I SYSTEM SYS1 - RESTARTED GLOBAL RESOURCE SERIALIZATION
   ```

While purging SYS1, global resource serialization detected a potential data integrity exposure, indicated by message ISG016I. When this message appears, the operator should follow the installation's procedures for resolving the problem, such as contacting the system programmer responsible for investigating the state of resources. More information about the problem normally appears in SYSLOG, where the following message describes any resources that might have been damaged by purging the system:

```
ISG018I REQUESTORS FROM SYSTEM SYS1 HAVE BEEN PURGED FROM RESOURCES
        NAMED xxxx,yyyy
```

Example: Two-System Complex

For a two-system complex, specify RESTART(YES) and REJOIN(YES) for the more important system and RESTART(NO) and REJOIN(YES) for the other system. If there is no difference, choose the restartable system arbitrarily. Figure 4-7 shows the definition for a two-system complex.

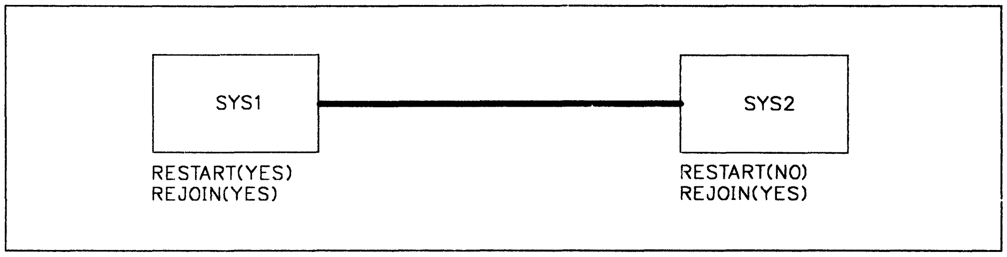


Figure 4-7. Two-System Complex

If SYS2 fails, SYS1 automatically restarts the ring. If SYS2 was temporarily stopped, it can automatically rejoin the ring when it resumes processing. In this case, the ring recovers without operator intervention. If SYS1 fails, however, recovering the ring requires operator intervention.

This example shows how to recover the ring when SYS1 fails.

On SYS2, the following messages appear:

```
ISG023E GLOBAL RESOURCE SERIALIZATION HAS BEEN DISRUPTED - GLOBAL
        RESOURCE REQUESTORS WILL BE SUSPENDED
ISG025E SYSTEM SYS2 UNABLE TO INITIATE AUTORESTART PROCESSING -
        THIS SYSTEM IS NOT AUTHORIZED
```

These messages indicate that an error has disrupted ring processing, and that SYS2 is not authorized to restart the ring automatically. SYS2 becomes inactive; any task that tries to obtain or free a global resource is suspended. Suspended tasks will quickly degrade performance on both systems. Thus, it is imperative that an operator take quick action.

On SYS2, the operator issues VARY GRS(ALL),RESTART.

Note: The operator trying to restart the ring must always issue VARY GRS(ALL),RESTART rather than VARY GRS(sysname),RESTART or VARY GRS(*),RESTART to restart a specific system. When the operator issues VARY GRS(ALL),RESTART, global resource serialization checks for the possibility of split rings.

On SYS2, the following messages appear:

```
ISG026I SYSTEM SYS2 MAY CREATE A SPLIT RING IF ANY OTHER GRS SYSTEM
        IS ACTIVE.  VERIFY THAT NO GRS SYSTEM IS ACTIVE BEFORE
        CONFIRMING RESTART
ISG027D CONFIRM RESTART RING FOR SYSTEM SYS2 - REPLY NO OR YES
```

Before replying to message ISG027D, the operator must issue D GRS and/or check with the other operator to verify that SYS1 is not active. If SYS1 is inactive, the operator can reply YES to continue the restart. If SYS1 is active, the operator must reply NO to avoid split rings. In this example, SYS1 is inactive; the operator can safely reply YES.

On SYS2, the operator replies YES to message ISG027D. The following messages appear:

```
ISG011I SYSTEM SYS2 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS2 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

On SYS2, the operator must determine if the failure on SYS1 requires SYS1 to re-IPL or if the failure on SYS1 occurred because the operator stopped SYS1 temporarily.

If the failure is temporary, no further action is needed. When SYS1 resumes processing, it automatically rejoins the ring because REJOIN(YES) was specified on both systems. The following message appears:

```
ISG012I RESTART REQUEST PASSED TO SYS2
```

The following messages appear on both systems when SYS1 rejoins the ring:

```
ISG011I SYSTEM SYS1 - RESTARTING GLOBAL RESOURCE SERIALIZATION
ISG013I SYSTEM SYS1 - RESTARTED GLOBAL RESOURCE SERIALIZATION
```

If the failure requires SYS1 to re-IPL, recovering the failed system requires the following steps:

1. On SYS2, issue VARY GRS(SYS1),PURGE to remove SYS1 from the ring. The following messages might appear:

   ```
   ISG016I SYSTEM SYS1 OWNS OR IS WAITING FOR GLOBAL RESOURCES
   ISG017D CONFIRM PURGE REQUEST FOR SYSTEM SYS1 - REPLY YES OR NO
   ```

   These messages indicate a possible data integrity exposure. In this example, the operator can reply YES.

   The following message appears:

   ```
   ISG011I SYSTEM SYS1 - BEING PURGED FROM GRS COMPLEX
   ```

2. On SYS1, perform problem determination, then re-IPL the system with GRS=JOIN.

3. On both systems, the following messages appear when SYS1 rejoins the ring:

   ```
   ISG011I SYSTEM SYS1 - RESTARTING GLOBAL RESOURCE SERIALIZATION
   ISG013I SYSTEM SYS1 - RESTARTED GLOBAL RESOURCE SERIALIZATION
   ```

While purging SYS1, global resource serialization detected a potential data integrity exposure, indicated by message ISG016I. When this message appears, the operator should follow the installation's procedures for resolving the problem, such as contacting the system programmer responsible for investigating the state of resources. More information about the problem normally appears in SYSLOG, where the following message describes any resources that might have been damaged by purging the system:

```
ISG018I REQUESTORS FROM SYSTEM SYS1 HAVE BEEN PURGED FROM RESOURCES
        NAMED xxxx,yyyy
```

Reactivating a System

Restarting a quiesced system requires the assistance of an active system — a system that is currently actively participating in the global resource serialization ring. The reason for this requirement is that an active system has current information about global resource requests. An inactive system also has current information about global resource requests; an inactive system can make itself an active system without introducing a data integrity exposure. A quiesced system, however, does not have current information about global resource requests; it cannot make itself an active system without introducing a data integrity exposure.

Under normal conditions, including most ring disruptions, at least one system is active and can restart a quiesced system, or an inactive system can make itself active and can restart a quiesced system.

Under unusual situations, all systems in the ring might be either quiesced or failed. This situation can occur when the operators quiesce all but one system and then the only system still active fails. When all systems are quiesced or have failed, there are two ways to recover the ring:

1. You can perform a complex-wide IPL. This process is cumbersome and time-consuming, particularly when MVS on the quiesced systems is still running.
2. You can allow global resource serialization to reactivate a system. **Reactivating a system** means turning a quiesced system into an active system. It avoids the complex-wide IPL but can introduce data integrity exposures or other problems related to global resources.

In balance, reactivating a system is less disruptive than a complex-wide IPL, and careful planning for the process can reduce the impact of the data integrity exposure.

Starting the process of reactivating a system requires no explicit operator action; reactivation is part of the processing global resource serialization performs when it executes a VARY GRS command with the RESTART operand. If the system on which the command was entered determines that all systems that it can communicate with are quiesced, an operator message indicates the need to reactivate a quiesced system. Before allowing the reactivate process to continue, the operator must:

1. Verify that all systems are quiesced, that none of the systems in the ring are currently active or inactive or temporarily stopped.
2. Verify that all CTC links are operational to ensure that the systems can communicate with each other

Issuing D GRS is an essential step. In a situation that requires reactivating a system, the display shows that all systems are not responding or QUIESCED. For the reactivation to succeed, the links between the quiesced systems must show up as ALTERNATE.

The operator can then allow the reactivate process to continue. To minimize the data integrity exposure, global resource serialization determines which of the quiesced systems has the most current information about global resources. The operator might have to enter the restart command again on a different system, the system with the most current information (identified in message ISG121I).

The system with the most current information then controls the reactivate process. It requests resource status information from every other system and reports the results to the operator:

1. The system responds to the request and supplies resource status. A message notifies the operator that the system can be restarted.
2. The system responds but does not send resource status. The system might have more current resource information, but it cannot reactivate the ring, probably because it is not a system that includes MVS/SP Version 3 or MVS/SP Version 2.2 with the fix for APAR OY13487. Before proceeding, the operator must determine the actual state of the system and, in most cases, stop the system.
3. The system does not respond. Before proceeding, the operator must determine the actual state of the system and stop it, if necessary. In most cases, the system is one that has failed.

After verifying the states of the systems, the operator can allow the reactivate process to continue.

Reactivating a quiesced system requires an operator who is well-versed in ring recovery procedures and very familiar with the ring configuration. "Example: Reactivating a Quiesced System" shows the detailed steps that it might require.

Reactivate Messages

The global resource serialization messages related to reactivating a system include ISG110E, ISG111D, ISG112I, ISG113I, ISG114I, ISG115I, ISG116E, ISG117D, ISG118I, ISG119I, ISG120I, and ISG121I.

Example: Reactivating a Quiesced System

This example shows how to use the reactivate function in a four-system complex. Assume that the following events have occurred:

1. At 10:00, the operator issues VARY GRS(*),QUIESCE on SYS4.
2. At 10:01, the operator issues VARY GRS(*),QUIESCE on SYS3.
3. At 10:02, the operator issues VARY GRS(*),QUIESCE on SYS2.
4. At 10:05, the only remaining active system, SYS1, fails.

Figure 4-8 shows the configuration of the complex and the state of each system at the time SYS1 failed. Issuing D GRS on any system but SYS1 would not show SYS1 but would show that all other systems are QUIESCED, and that the links between the quiesced systems are ALTERNATE.

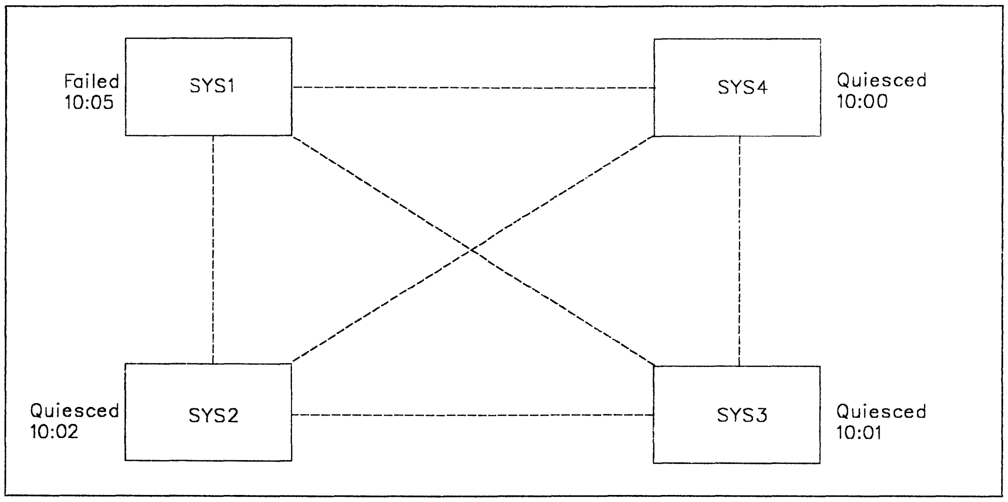


Figure 4-8. Reactivate Process Example

To resolve the problem, the installation chooses to use the reactivate process rather than perform a complex-wide IPL. The operator on SYS4 begins the process.

On SYS4, the operator issues VARY GRS(ALL),RESTART. Note that VARY GRS(*),RESTART or VARY GRS(SYS4),RESTART would have the same effect.

On SYS4, the following message appears:

```
ISG121I VARY REQUEST REJECTED - ENTER COMMAND ON SYSTEM SYS2
```

This message indicates that global resource serialization has determined that another system, SYS2 in this example, has more current resource information. It tells the operator to enter the restart command on SYS2, which minimizes but does not entirely eliminate the data integrity exposure connected to reactivating a quiesced system.

On SYS2, the operator enters VARY GRS(SYS2),RESTART. The following messages appear:

```
ISG110E NO ACTIVE SYSTEMS FOUND IN THE GRS COMPLEX. REACTIVATE
        SHOULD BE ATTEMPTED IF THERE ARE NO ACTIVE GRS SYSTEMS IN OPERATION.
        REACTIVATE MUST NOT BE ATTEMPTED IF THERE ARE ACTIVE SYSTEMS.
ISG111D CONFIRM REACTIVATE SHOULD BE ATTEMPTED - REPLY NO OR YES
```

These messages indicate that SYS2 has the most current resource information of all the quiesced systems and has thus accepted the command. They also warn the operator that the reactivate process is about to begin. Before replying YES to message ISG111D, the operator must verify that:

1. There is no ACTIVE or INACTIVE system in the complex.
2. All required links are operational.
3. No system has been temporarily stopped since SYS2 was quiesced.

In most installations, the operator should also contact the system programmer responsible for investigating the state of resources to describe the problem and consider the alternative of a complex-wide IPL. If the complex-wide IPL is preferable, the operator would reply NO to message ISG111D. In this example, the reactivate process is to continue.

On SYS2, the operator replies YES to message ISG111D. The following messages appear:

```
ISG112I SYSTEM SYS3 RESPONDED WITH GLOBAL RESOURCE STATUS -- IT WILL
        BE PERMITTED TO RESTART AFTER REACTIVATE COMPLETES
```

This message indicates that the system is responding and does not have the most current information about global resources.

```
ISG114I RESOURCE STATUS NOT RECEIVED FROM SYSTEM SYS4 -- IT MUST BE
        STOPPED BEFORE CONFIRMING REACTIVATE COMPLETION
```

This message indicates that the system is responding but did not supply information about global resources. The system might have more current information, but it cannot reactivate the ring. (Only systems that include MVS/SP Version 3 or MVS/SP Version 2.2 with the fix for APAR OY13487 can reactivate the ring.) The operator must make sure the system is stopped before proceeding.

```
ISG113I NO RESPONSE RECEIVED FROM SYSTEM SYS1 - IT MUST BE
        STOPPED BEFORE CONFIRMING REACTIVATE COMPLETION
```

This message indicates that the named system did not respond. In this example, this message confirms that SYS1 has failed. In other cases, the operator must investigate the state of the named system and make sure the system is stopped before proceeding.

```
ISG115I IF ANY SYSTEMS WERE NOT LISTED IN THE PREVIOUS LIST,
        THEY ARE UNKNOWN TO THIS SYSTEM - THEY MUST BE STOPPED BEFORE
        CONFIRMING REACTIVATE COMPLETION
ISG116E STOP INDICATED SYSTEMS BEFORE CONFIRMING REACTIVE
        COMPLETION.  REPLY NO TO CANCEL REACTIVATE
ISG117D CONFIRM REACTIVATE SHOULD BE COMPLETED - REPLY NO OR YES
```

These messages warn the operator to make a final check before allowing the reactivate process to complete. At this point, the operator could verify, for example, that no system had joined the ring after SYS2 was quiesced but before SYS1 failed. Message ISG112I, ISG113I, or ISG114I must appear for every system in the complex. If there is no message for a system, the operator must investigate the reason before proceeding. Issuing D GRS might show an ACTIVE system. In this case, or if there are any other potential problems, the operator must reply NO to message ISG117D. Replying YES allows the reactivate process to complete.

On SYS2, the operator replies YES to message ISG117D. The following message indicates that SYS2 is now an active system:

```
ISG118I REACTIVATE FUNCTION IS COMPLETE. SYSTEM SYS2 HAS RESTARTED
        AS A RING OF ONE SYSTEM.
```

On SYS2, the operator takes the following actions to complete the recovery:

1. Issues VARY GRS(SYS1),PURGE to purge SYS1 from the ring.
2. Issues VARY GRS(SYS4),PURGE to purge SYS4 from the ring.
3. Issues VARY GRS(SYS3),RESTART to bring SYS3 back into the ring.

On SYS1, the operator performs problem determination, then re-IPLs the system with GRS = JOIN.

On SYS4, the operator re-IPLs the system with GRS = JOIN.

At this point, the recovery process is complete.

Chapter 5. Installing and Tuning the Complex

There are many possible ways to install a global resource serialization complex. "Installing the Complex" describes one possible approach. You will also want to evaluate the performance of the complex and take actions that will minimize the performance cost of the availability benefits that a global resource serialization complex provides; see "Tuning the Complex" on page 5-3.

Installing the Complex

How you actually install the global resource serialization complex depends — like every other decision related to global resource serialization — on the needs of your installation. Figure 5-1 shows one possible approach to installing the complex. The approach is roughly based on the order of performing the work, and it breaks the work you must do into installation steps, planning tasks, and follow-on considerations.

This approach was chosen because it seems most likely that you would first install an initial complex to provide a quick solution to your most pressing resource protection problems. Then, while you are testing and learning from the initial complex, you can gradually move toward the final complex that meets all of your resource protection needs.

Figure 5-1 (Page 1 of 2). Installing a Global Resource Serialization Complex

Installation Steps	**Planning Tasks**	**Follow-on Considerations**
	Plan your migration to your new system.	
1. Install the new system without building a complex. To indicate to MVS that there is no complex, replace the default parameter GRS=JOIN with GRS=NONE. Global resource serialization then runs only within each system; it treats all resource requests as local resources and allows the system to issue the reserve for all RESERVE requests.	Identify the resources that you must include in the RNLs to get benefits from the complex as quickly as possible.	Define the long-term resource processing goals for your installation. That is, identify all the resources that you eventually want to protect as global resources. Evaluate the data set naming conventions at your installation and begin any changes required to make better use of global resource serialization. Evaluate the possibility of modifying existing applications to use global resource serialization more effectively. Determine how global resource serialization will affect the design of future applications.
	Design the initial complex, which might not include all of the systems that will eventually make up the complex. That is, you might want to start with a two-system complex, test it, and then add another system, repeating the process until the complex is complete.	If your initial complex is a subset of the complete or desired complex, design the complete complex and identify the steps you will follow to move from the initial complex to the complete complex.
	Prepare the initial educational and procedural material for the system operators.	If your initial complex is a subset of the complete complex, do your initial operations planning with the complete complex in mind.

Figure 5-1 (Page 2 of 2). Installing a Global Resource Serialization Complex

Installation Steps	**Planning Tasks**	**Follow-on Considerations**
2. Modify the SYSTEM inclusion and SYSTEMS exclusion RNLs. Leave the RESERVE conversion RNL empty. Create the GRSCNFxx parmlib member. Copy the member to the parmlib of each system in the complex. Update IEASYSxx on each system in the initial complex to include appropriate values for the global resource serialization system parameters.	Create the GRSRNLxx parmlib member. Copy the member to the parmlib of each system in the complex. Test GRSRNLxx to ensure that the RNLs contain no syntax errors.	If your initial complex is a subset of the complete complex, identify the tasks required to define the final complex to MVS.
3. IPL each system as a one-system complex. That is, use GRS=START or GRS=JOIN but use an empty RESERVE conversion RNL and ensure that all CTC link connections are offline (disabled). Each system will then act as if it were part of a complex, but no global resources actually exist. However, global resource serialization processes requests for such resources as if they did. Data sets on shared DASD volumes are protected by reserves.	Testing each system as a one-system complex enables you to monitor the performance and reliability of global resource serialization and estimate the processing overhead the complex creates. Some tuning actions can be performed at this time. For example, you can use RMF to sample ENQ, DEQ, and RESERVE activity and change the contents of the RNLs accordingly. See "Tuning the Complex" on page 5-3.	Evaluate your long-term goals in the light of experience gained with the global resource serialization complex. Begin to develop a long-term plan for measuring the effectiveness of the global resource serialization complex. See "Tuning the Complex" on page 5-3.
4. IPL the systems in the initial complex. That is, IPL each system with GRS=START or GRS=JOIN and ensure that CTC link connections are online (enabled). If any DASD volumes are shared either with systems outside the complex or with systems that you plan to include in the complex at a later date, continue to use an empty RESERVE conversion RNL.	Repeat the monitoring and tuning actions that you performed for the one-system complex. If the RESERVE conversion RNL is still empty, complete any shared DASD changes required to ensure that the complex does not share DASD volumes with systems outside the complex. These changes might include modifying the RNLs or adjusting your use of shared DASD volumes.	
5. When it is safe, modify the RESERVE conversion RNL on all systems and IPL the complex again to use global resource serialization to protect resources on shared DASD volumes.		
6. Repeat the planning and installation process for each new system you add to the complex until the complex design and resource processing match your long-term goals.	Continue to monitor, evaluate, and tune the performance of the global resource serialization complex.	

Tuning the Complex

A global resource serialization complex increases the availability of systems and resources, and might improve system performance, because converting reserves can:

- Reduce the number of system interlocks that occur because of reserves
- Decrease contention for resources on volumes now serialized by use of a reserve
- Remove the need to use job scheduling to serialize access to resources on shared DASD volumes
- Avoid the data integrity exposure that occurs when a system reset prematurely ends a reserve
- Avoid the situation when one processor can monopolize a shared volume

A global resource serialization complex does, however, increase the system's use of the processor, channels and devices, and storage. This additional overhead occurs for two main reasons:

1. Every system in the ring processes every request for a global resource and maintains information about the status of every global resource.
2. The RSA-message used to pass information about global resource requests from one system to another requires processing overhead each time it makes a cycle around the ring. This overhead occurs even when the RSA-message is empty.

In addition, every task that requests a global resource is suspended while the ring processes the request. The time that a task is suspended while it is waiting for access to a global resource is the global resource request response time.

Thus, tuning a global resource serialization complex involves balancing your installation's need for an acceptable response time while minimizing the effect that ring processing has on overall system performance. Except for very small processors or systems that are already running close to capacity, the effect of ring processing is normally not significant. Thus, focus your tuning work on response time — the delay that individual global resource requestors encounter. Your installation must determine what response time is acceptable.

If the response time is acceptable, there is no need to tune the complex. If the response time is not acceptable, then you need to investigate the problem and take action to resolve, or at least minimize, the difference between the actual response time and the acceptable response time.

The design of your complex, described in detail in Chapter 3, "Designing the Complex" on page 3-1, is very closely connected to ring performance. Thus, many of the factors that affect ring performance are relatively fixed; for example, if all other factors are equal, a three-system ring takes more time to process a global resource than a two-system ring. If you need three systems in the ring, however, you cannot resolve a performance problem by removing one system.

To evaluate the impact of these factors, you first need to understand how to calculate average response time. To illustrate the calculations, assume the following configuration:

- There are four systems in the ring.
- Communication links are through an IBM 3088 MCCU.
- The RESMIL value for all systems is 1 millisecond.
- The RSA-message size is 4K.

Average Response Time

Average response time is the average amount of time a requestor must wait before a request for a global resource can be granted. It depends on cycle time — the amount of time that the RSA-message requires to make a complete cycle around the ring. In tuning ring performance, the primary goal is to ensure that the average response time is as close as possible to the acceptable response time at your installation. You can use the following formulas to calculate average response time and predict the effect of changes you might make. The formulas you need depend on whether your ring is using ring acceleration.

If the ring is using ring acceleration, then **n** is the number of systems, and **a** is the ACCELSYS value. The calculations for cycle time and average response time are:

```
cycle time = (RESMIL * n) + (transmission delay * n)
response time = cycle time/2 + (RESMIL * (a - 1)) + (transmission delay * a)
```

Because the configuration includes a 3088, the transmission delay is about 1 millisecond for the RSA-message and 1 millisecond for the ring acceleration signal. Assuming ACCELSYS(2), the average response time is about 7 milliseconds, as follows:

```
cycle time = (1 millisecond * 4) + (1 millisecond * 4)
response time = 8 milliseconds/2 + 1 millisecond + 2 milliseconds
```

If the ring is not using ring acceleration, **n** is the number of systems, and the calculations for cycle time and average response time are:

```
cycle time = (RESMIL * n) + (transmission delay * n)
response time = cycle time * 1.5
```

In this case, the average response time is about 12 milliseconds, as follows:

```
cycle time = (1 millisecond * 4) + (1 millisecond * 4)
response time = 8 milliseconds * 1.5
```

Experience has shown that the cycle time is a stable value; it does not normally vary significantly during any given time period. Resource contention, however, can add to the response time. That is, average response time measures only the time a global resource requestor waits for ring processing to complete. It does not reflect the time spent waiting until a requested resource becomes available (this delay is not directly related to ring processing).

Tuning Factors

Achieving acceptable response time for global resource requests involves many factors:

- Number of systems in the ring
- Transfer rate on the communication links
- Level (version and release) of MVS
- RSA-message size
- Global resource request rate
- RESMIL value

The following sections describe the factors that can affect how successfully your complex meets your goals. Where applicable, the descriptions include specific actions for particular problems.

Number of Systems

It is obvious that a two-system complex can respond more quickly to requests for global resources than a four-system complex. The fewer the systems, the shorter the cycle. Adding a system to the ring, however, need not cause a corresponding increase in response time. Lowering the residency time (RESMIL value) on all systems can make the addition of another system transparent to global resource requestors.

Transmission Rate

The transmission rate reflects how quickly the systems in the ring can communicate. It is, like the number of systems in the ring, relatively fixed. Using an IBM 3088 to provide communication links yields significant improvement over using integrated CTC adapters. Following the channel placement recommendations in "Link Placement" on page 3-3 can avoid channel delay. The level of MVS installed can also affect the transmission rate.

Level of MVS

While the level of MVS installed on the systems in the complex can affect the transmission rate, it can also affect the RSA-message size, and the size of a global resource request.

For levels of MVS earlier than MVS/SP Version 2 Release 2, the RSA-message size is 4K.

With MVS/SP Version 2 Release 2, the default RSA-message size is 8K, and an installation can increase the size to 32K. Compression of requests, as well as the larger RSA-message, increase ring capacity — the number of global resource requests the ring can process in a given period of time.

For MVS/SP Version 3, the default RSA-message size is 32K, and the average size of a global resource request is further compressed. These changes promote increased ring capacity. Ring acceleration can also reduce response time; see "Ring Acceleration (ACCELSYS)" on page 3-12. In addition, MVS/SP Version 3 processes DEQ requests differently; when a task issues a DEQ for a single global resource, global resource serialization allows the task to resume execution almost immediately; it does not wait to send the request around the ring. This technique can reduce response time.

The degree to which the level of MVS installed affects ring performance is most noticeable when all systems in the ring are at the same level. For example, ring acceleration requires that all systems include MVS/SP Version 3. In a mixed complex, benefits are not as effective. An example of how a mixed complex can affect ring performance is the size of the RSA-message.

RSA-Message Size

The maximum size of the RSA-message, as well as the cycle time and the average global resource request size, determine ring capacity — the number of global resource requests the ring can process in a given period of time.

In a mixed complex, global resource serialization automatically adjusts the maximum size of the RSA-message to the lowest common value. For example, assume that the ring consists of two MVS/ESA systems; the maximum RSA-message size is 32K. Assume that an MVS/XA system joins the ring; it is an MVS/SP Version 2 Release 2 system using the default RSA-message size of 8K. Global resource serialization adjusts the maximum size of the RSA-message to 8K. If the MVS/XA system leaves the ring, global resource serialization again adjusts the size, in this case restoring it to 32K.

In this example, the MVS/XA system reduces ring capacity by limiting the maximum size of the RSA-message to 8K. Because the MVS/XA system is running MVS/SP Version 2 Release 2, you can change the default maximum RSA-message size to 32K, if you have applied the fix for APAR OY08389.

Whether ring capacity is a performance problem or not depends on the global resource request rate.

Global Resource Request Rate

The global resource request rate is the number of global resource requests that the systems in the ring actually generate, most conveniently expressed as the number of global resource requests per second.

If the systems generate more requests than the ring can process, the RSA-message cannot hold all of the requests. Some requests must wait for at least one additional cycle before getting into the RSA-message, thus increasing response time. Using the RNLs to reduce the number of unnecessary global resource requests is a possible solution. Lowering the residency time (RESMIL value) is often a better way to deal with a high global resource request rate. The lower RESMIL value increases ring capacity; the RSA-message makes more cycles around the ring in any given time period, and thus the ring can process more requests in the same amount of time.

RESMIL Value

The residency time (also called the RESMIL value) is the amount of time that the RSA-message spends in each system in the ring. Each system holds the RSA-message for at least as long as it takes the system to process the requests in the message. If this time is more than the RESMIL value, the system sends the RSA-message on as soon as it has finished processing the requests. If this processing time is less than the RESMIL value, the system holds the RSA-message until the RESMIL value expires, then sends it on to the next system.

"Residency Time Value (RESMIL)" on page 3-9 describes how to set an initial value when you set up your complex. Use the information in "Residency Time Value (RESMIL)" as you tune your complex. Setting the RESMIL value correctly is your best technique for tuning the complex. Experiments have shown that a RESMIL value as low as one millisecond in a complex of 3090 processors does not significantly affect processor utilization.

Establishing a response time objective is one way to select a value for RESMIL. (See "Measuring Response Time" on page 5-10 for one way to measure actual response time.) To establish a response time objective, determine what response time is acceptable or desirable, then set the RESMIL value to meet that objective.

Choose a low response time objective (10 milliseconds or lower) if all of the systems in the complex are 3090 processors or if any one of the following conditions is true:

- Your installation has chosen to convert catalog reserve (SYSIGGV2 and SYSZVVDS).
- Your installation has batch jobs that issue many global resource requests and that must complete in a fixed time.

Once you have set a response time objective, you next translate that objective into a RESMIL value, depending on the number of systems in the ring. Figure 5-2 on page 5-8 and Figure 5-3 on page 5-8 can help you with this process. The RESMIL values shown are based on the formulas shown earlier in "Average Response Time" on page 5-4, with the following assumptions:

- The RSA-message contains only one request.
- The transmission time is one millisecond.
- There is no contention time for the requested resource.

The RESMIL values shown have been rounded down to the nearest integer. An asterisk next to a value means that, even with the lowest possible RESMIL value (1 millisecond), the response time objective cannot be met.

As described earlier, there are many factors (such as data transmission rate, RSA-message size, and global resource request rate) that affect the actual response time. Thus, setting a particular RESMIL value does not guarantee meeting the corresponding response time objective. You can, however, use the RESMIL value in the table as a starting point. Use the values in Figure 5-2 if you use ring acceleration; use the values Figure 5-3 if you do not.

Figure 5-2. RESMIL Values with Ring Acceleration — ACCELSYS(2)

Response Time Objective (Milliseconds)	**2-System Ring**	**3-System Ring**	**4-System Ring**	**5-System Ring**
10	3	2	2	1
20	8	6	5	4
30	13	10	8	7
40	18	14	12	10
50	23	18	15	13

Figure 5-3. RESMIL Values without Ring Acceleration

Response Time Objective (Milliseconds)	**2-System Ring**	**3-System Ring**	**4-System Ring**	**5-System Ring**
10	2	1	1*	1*
20	5	3	2	1
30	9	5	4	3
40	12	7	5	4
50	15	10	7	5

* Even with a RESMIL value of 1 millisecond, the response time objective cannot be met.

Tuning Process

Tuning a global resource serialization complex, like any system tuning process, requires a disciplined approach of measuring the system's performance, setting specific goals, taking actions to reach the goals, then measuring and evaluating the results of the actions.

You will probably want to take a base set of measurements of your system performance before you begin the process of installing your complex and repeat the measurements at several points along the way. If, for example, you choose the approach described under "Installing the Complex" on page 5-1, you might want to measure your system:

1. Before you begin to install the complex
2. While each system is running as a one-system complex
3. After the initial complex is running
4. After each change to the design of the initial complex — such as adding a system
5. After each tuning change that you make

Obviously, each set of measurements should be taken under the same set of conditions. That is, come as close as you can to ensuring that the system is processing the same workload during each measurement period.

Measurements

Gathering the information you need to measure and evaluate ring performance is a very important part of the tuning process. RMF reports provide much useful information about system performance and about resource use. You can use GTF to gather information about the size of the RSA-message, and you can measure response time.

Using RMF

Ring performance is only one aspect of total system performance. Focus your monitoring efforts on total system performance and look more closely at ring performance only when it appears to be affecting how well your system meets your overall performance objectives.

RMF Monitor I session reports are particularly useful in determining how ring performance affects overall system performance. One key report is **Workload Activity**. Use this report to determine the resources (processor, I/O, and storage) that global resource serialization is using. To obtain the clearest picture of resource consumption, place global resource serialization in its own performance group or report performance group.

RMF Monitor II session reports can also provide information about how global resource serialization affects the system. For example, you can use the **Address Space State Data** report to determine how much processor time global resource serialization consumes and the number of page faults the GRS address space experiences. To provide the optimum response time, the page-in rate for the GRS address space should be close to zero.

RMF Monitor III (workload delay monitor) reports can also help you deal with problems related to delays of jobs or users. These reports can show address spaces delayed and whether resource contention is contributing to the delay, as well as the jobs affected. They can point out jobs that are being slowed by ENQ delays. If the reports indicate ENQ delays, possible reasons might be:

- A job is delayed because the system is waiting for the RSA-message to complete its cycle. Lowering the RESMIL value might resolve the problem.
- A job is delayed because the RSA-message is encountering a communication delay. Ensure that the links used to send the RSA-message are not installed on the same channels as devices that monopolize the channel for long periods of time.
- A job is delayed because there is contention for the resource; another job is using it. If the delay is caused by a reserve, investigate the possibility of converting the reserve to reduce contention for the resource.

The Monitor I **Enqueue Contention Activity** report can also help you to identify the resources that are causing the most significant contention delays.

Using GTF

Using the generalized trace facility (GTF) can provide additional information. You can use GTF to monitor:

- How frequently the RSA-message passes around the ring
- The actual size of the RSA-message
- The duration of the cycle time

To gather this information, trace the global resource serialization CTC channel program for a short period of time during peak processing. Place the following GTF trace options in a parmlib member that GTF reads when the trace is started:

```
TRACE=SSCHP,IOP,CCWP
IO=SSCH=(C44,C4C,C54)
CCW=(SI,DATA=8)
END
```

These sample statements trace the CTC links identified by device numbers C44, C4C, and C54. Replace these device numbers to trace the links that global resource serialization is using in your installation at the time of the trace. The data count (DATA=8) is small because the actual contents of the RSA-message are not of interest. The trace statements do record the number of bytes in the RSA-message; a length of 39 bytes indicates an empty RSA-message (an RSA-message that contains no resource requests).

If the RSA-message is almost always empty, you can probably set a higher RESMIL value without affecting response time. If the RSA-message is frequently close to full (more than 25K when the maximum size is 32K), set a lower RESMIL value to send the RSA-message around the ring more frequently. If the RSA-message is full even occasionally (more than five percent of the time), it is probably having an adverse effect on response time. Set a lower RESMIL value to send the RSA-message around the ring more frequently.

For more information about using GTF, see *Planning: Dump and Trace Services* and *Initialization and Tuning*.

When you examine the data, using either the EDIT function of IPCS (interactive problem control system) or your own post-processing program, be aware that the direction (clockwise or counter-clockwise) of the RSA-message is determined dynamically when the systems build the ring. Thus, an address used to send the RSA-message at one time might receive it at another time. Once determined, the direction of the message cannot change unless a system enters or leaves the ring.

Measuring Response Time

The cause of a response time problem that shows up either as an increase in interactive user response time or an increase in the time it takes to complete a batch job, is sometimes difficult to pinpoint. One way to obtain a very useful end user view of ring performance is to code a program that issues repetitive global resource requests (ENQ macros with a scope of SYSTEMS) and measures the elapsed time required for a response.

Figure 5-4 on page 5-12 shows an example of a program that measures global resource request response time from the perspective of the problem program issuing the request. You can take repetitive samples, allowing you to analyze response time over a longer period.

The program uses STIMER to suspend itself for a user-specified interval. When the interval has expired, the program issues an ENQ with a scope of SYSTEMS for a nonexistent resource. The program notes the value of the system TOD clock before issuing the ENQ request and again when control returns. The difference between the two values is the global resource request response time. The program then frees the resource and repeats the request.

The GENQRESP program encounters the same contention for the processor and storage as does any other problem program in your system. Thus, even after the system grants the global resource request, the program might not receive control immediately because of other factors, such as waiting for the processor, paging, or swapping. To ensure that the response time GENQRESP measures is as close as possible to the actual response time, it is a good idea to run GENQRESP on a lightly-loaded system.

Run the program on the system in the ring that has the lightest workload to minimize the effect of paging and swapping delays on the results. Also, try to run the program each time under the same set of conditions. That is, come as close as you can to ensuring that the system is processing the same workload each time you run the program.

```
//GENQRESP JOB accounting information...
//STEP1 EXEC ASMHCLG,PARM.L='MAP,LIST,LET,NCAL,SIZE=(50000,45000)',
// REGION.L=200K
//C.SYSPUNCH DD DUMMY
//C.SYSIN DD *
*
* THE NUMBER OF REQUESTS TO ISSUE BEFORE TERMINATING, AND THE
* INTERVAL BETWEEN REQUESTS ARE SPECIFIED AS CONSTANTS IN THE CODE.
*
* THE OUTPUT DATASET, DD 'SAMPLES', HAS THE FOLLOWING FORMAT:
*
* START   END
*  COL    COL    FORMAT   DESCRIPTION
* -----  -----  -------  ------------------------------------------
*   1      8    TODCLOCK TOD CLOCK VALUE WHEN ENQ WAS REQUESTED
*   9     12    BINARY   RESPONSE TIME FOR GLOBAL ENQ REQUEST
*                        IN MICROSECONDS (1 SECOND = 10**6 MICROSECONDS)
*
* BECAUSE THE PROGRAM SAVES ONLY 32 BITS OF THE TOD CLOCK, RESPONSE
* TIMES LONGER THAN ABOUT ONE HOUR ARE NOT RECORDED CORRECTLY.
*
* FOR THE MOST CONSISTENT RESULTS, ASSIGN THIS JOB TO A DOMAIN
* WITH A MINIMUM MPL HIGH ENOUGH TO KEEP IT FROM BEING SWAPPED OUT.
*
GENQRESP   CSECT
           STM    14,12,12(13)        SAVE REGISTERS ON ENTRY
           USING  GENQRESP,12         ESTABLISH ADDRESSABILITY
           LR     12,15
           OPEN   (MODDCB,OUTPUT)     OPEN OUTPUT DATASET
           L      6,SAMPLES           # SAMPLES PER CYCLE
LOOP       STCK   BEFORE              SAVE TIME BEFORE ENQ
           ENQ    (QNAME,RNAME,E,8,SYSTEMS)
           STCK   AFTER               WHEN DID WE GET BACK?
           DEQ    (QNAME,RNAME,8,SYSTEMS)
           LD     4,BEFORE            STORE TOD VALUE WHEN ENQ WAS
           STD    4,OUTBUF             ISSUED
           L      4,AFTER+4           LOOK AT LOW-ORDER WORD OF TIME
           SL     4,BEFORE+4          COMPUTE RESPONSE (AFTER-BEFORE)
           SRL    4,12                TRUNCATE AT MICROSECOND BIT
           ST     4,OUTBUF+8          SAVE ENQ RESPONSE IN BUFFER
           PUT    MODDCB,OUTBUF
           STIMER WAIT,BINTVL=INTVL
           BCT    6,LOOP              IF MORE SAMPLES TO TAKE, GO BACK
           CLOSE  (MODDCB)            ALL DONE.  CLOSE THE FILE,
           LM     14,12,12(13)
           SR     15,15               ZERO RETURN CODE
           BR     14
SAVE       DC     18A(0)
QNAME      DC     CL8'QNAMEABC'  FICTITIOUS NAME FOR ENQ/DEQ
RNAME      DC     CL8'ENQTIMER'
BEFORE     DC     2A(0) ENQ 'BEFORE' TIME
AFTER      DC     2A(0) ENQ 'AFTER' TIME
SAMPLES    DC     A(100)             TAKE THIS MANY SAMPLES
INTVL      DC     A(33)              HOW LONG TO SUSPEND BETWEEN
*                                     ENQ'S (HUNDREDTHS OF A SECOND)
           CNOP   0,4
OUTBUF     DC     80AL1(0)
MODDCB     DCB    DDNAME=SAMPLES,DSORG=PS,MACRF=PM
           END
//L.SYSIN   DD DUMMY
//G.SAMPLES DD DSN=ENQRESP.DATA,
// UNIT=SYSDA,
// DISP=(NEW,CATLG),
// DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB),
// SPACE=(TRK,(1,1))
```

Figure 5-4. Measurement Program Example

Actions

If any of these measurements reveal problems, the most important and useful action is to change the RESMIL value. Divide the change equally among all systems. There is no need to have a different RESMIL value for each system. "Residency Time Value (RESMIL)" on page 3-9 and "RESMIL Value" on page 5-6 contain more detailed information.

If your major area of concern is the response time for global resource requests (that is, your concerns are resource availability or delays for work because of ring processing), you can reduce the cycle time by reducing the RESMIL value for each system. Reducing the cycle time reduces the time global resource serialization requires to process each request for a global resource and makes the complex more responsive to individual resource requests. Reducing the cycle time, however, increases the number of times the system must process the RSA-message per second; it thus tends to increase the global resource serialization overhead — the demands it makes on the processor, channels, and devices.

If your major area of concern is the system overhead that global resource serialization causes (that is, your concern is system throughput), you can increase the cycle time by increasing the RESMIL value for each system. Increasing the cycle time reduces the system overhead. It decreases the number of times the system must process the RSA-message per second and thus tends to decrease the demands global resource serialization makes on the processor, channels, and devices. A longer cycle time, however, tends to increase the time global resource serialization requires to process each request for a global resource and makes the complex less responsive to individual requests for global resources.

There are two other basic tuning actions. One is to reduce the number of requests for global resources that the complex must process. That is, you want to ensure that all resources processed as global resources actually require global serialization. To reduce the number of requests for global resources, place entries in the SYSTEMS exclusion RNL to exclude from global serialization those resources, such as temporary data sets, that require only local serialization.

Another basic action is to convert as many reserves as possible. This action does not directly affect the performance of the complex, but it does increase the availability benefits your installation gets from the complex. Therefore, consider developing a long-term plan to work toward the goal of increasing the number of reserves converted to requests for global resources. For example, if you cannot convert reserves only because a volume can be accessed both by systems in the complex and systems outside the complex, change your use of the volume so that only systems in the complex can access it. You can then convert the reserves.

Appendix A. The RNL Syntax Checker

To prevent unnecessary IPLs caused by incorrect RNL entries, use the RNL syntax checker to verify resource names.

The RNL syntax checker lets you check the syntax of GRSRNL parameters specified on the JCL and resource names specified in GRSRNLxx member(s) of SYS1.PARMLIB.

The RNL syntax checker performs three major tasks.

- It searches for the specified GRSRNLxx members of SYS1.PARMLIB.
- It verifies the syntax of RNLDEF statements in each member.
- It issues information and error messages.

For information on the messages the RNL syntax checker issues, see "RNL Syntax Checker Messages" on page A-7.

When the RNL syntax checker returns a return code of zero, you can IPL the system successfully.

Note: If the RNLDEF keyword is missing from an RNL statement in the SYS1.PARMLIB member, the RNL syntax checker immediately issues an error message and does not check the rest of the statement. Thus, after correcting an error, execute the RNL syntax checker again on the specified SYS1.PARMLIB member(s) to verify the remaining RNLDEF statements.

Installing the RNL Syntax Checker

The RNL utility checker consists of three modules: ISGRNLCK, ISGRNLMS, and ISGRNLPR. To install the utility:

1. Assemble the source code for each module separately
2. Link-edit the three object modules into a single load module.

The steps are outlined next.

Assembling the Syntax Checker Source Modules

In the following JCL, underscores indicate values that you can replace with information specific to your installation.

The numbers are not part of the JCL statements but correspond to notes that follow the example. Do not code these as part of the JCL.

```
1.  //ASSEMBLE JOB
    //* INVOKING ASSEMBLER
2.  //ASMSTEP EXEC PGM=ASMBLR,REGION=1024K,
3.  //        PARM='LOAD,NODECK'
4.  //SYSLIB DD DSN=SYS1.MACLIB,DISP=(SHR,PASS)
    //SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE)
    //SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE)
    //SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1)),DISP=(NEW,DELETE)
5.  //SYSLIN DD DSN=RNLS.PGMS.OBJ(module),DISP=OLD
6.  //SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=3509),
    //        COPIES=1
    //SYSPUNCH DD SYSOUT=A,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),COPIES=1
7.  //SYSIN DD DSN=SYS1.SAMPLIB(module),DISP=OLD,UNIT=SYSDA
    //SYSUDUMP DD SYSOUT=A
    //SYSABEND DD SYSOUT=A
    //
```

Note: For **module** specify ISGRNLCK, ISGRNLMS, or ISGRNLPR. Each module must be assembled separately.

Notes on the JCL to Assemble RNL Syntax Checker Modules

Each numbered JCL statement in the previous example is described as follows:

1. The JOB statement specifies the beginning of a job and assigns a job name.
2. The EXEC statement specifies the beginning of a job step and identifies the assembler.
3. The PARM parameter specifies options for the assembler.
4. The SYSLIB DD statement specifies the macro library concatenation.
5. The SYSLIN DD statement specifies the data set that will contain the object module (ISGRNLCK, ISGRNLMS, or ISGRNLPR.) You must assemble each module separately.
6. The DCB (data set control block) specifies the DCB options.
7. The SYSIN DD statement specifies the data set that contains the source code.

Link-Editing the RNL Syntax Checker Modules

In the following JCL, underscores indicate values that you can replace with information specific to your installation.

The numbers are not part of the JCL statement but correspond to notes that follow the example. Do not code these as part of the JCL.

```
1.  //LINKEDIT JOB
2.  //LINK EXEC PGM=IEWL,
3.  //    PARM='XREF,REUS,LET,LIST,NCAL,SIZE=(750K,200K)'
    //SYSPRINT DD SYSOUT=A
4.  //DD1 DD DSN=RNLS.PGMS.OBJ
    //SYSUT1 DD SPACE=(CYL, (1,1)),UNIT=SYSDA
5.  //SYSLMOD DD DSN=RNLCHECK.LOAD,UNIT=XXX,VOL=SER=YYYYYY,DISP=OLD
6.  //SYSLIN DD  *
       INCLUDE DD1 (ISGRNLCK)
       INCLUDE DD1 (ISGRNLMS)
       INCLUDE DD1 (ISGRNLPR)
       ENTRY ISGRNLCK
       NAME ISGRNLCK(R)
/*
//
```

Note: The partitioned data set RNLS.PGMS.OBJ must contain the assembled members ISGRNLCK, ISGNRLMS, and ISGRNLPR.

Notes on the JCL to Link-Edit the RNL Syntax Checker Modules

Each numbered JCL statement in the previous example is described as follows:

1. The JOB statement specifies the beginning of a job and assigns a name to the job.
2. The EXEC statement specifies the beginning of a job step and identifies the linkage editor.
3. The PARM parameter specifies linkage editor options.
4. The DD1 statement identifies a data set containing the object modules ISGRNLCK, ISGRNLMS, and ISGRNLPR.
5. The SYSLMOD DD statement specifies the data set that receives the object modules.
6. SYSLIN DD * contains the linkage editor or control statements for the RNL syntax checker.

Using the RNL Syntax Checker

You can execute the RNL utility checker in background mode or in the foreground under TSO/E.

Executing the RNL Syntax Checker in Background Mode

To execute the job in the background mode, use the following JCL.

In the following JCL, underscores indicate values that you can replace with information specific to your installation. The numbers are not part of the JCL statements but correspond to notes that follow the example. Do not code these as part of the JCL.

```
1.  //CHECKRNL JOB
2.  //RNLSTEP EXEC  PGM=ISGRNLCK,
3.  //        PARM='OA,OB,OC,OD'
4.  //STEPLIB DD DSN=RNLCHECK.LOAD,DISP=SHR,
    //  UNIT=XXXX,VOL=SER=YYYYYY
    //SYSPRINT DD SYSOUT=A
    //SYSABEND DD SYSOUT=A
5.  //PARMLIB DD DSN=SYS1.PARMLIB,DISP=SHR
    /*
```

Notes on Running the RNL Syntax Checker in Batch Mode

Each numbered JCL statement in the previous example is described as follows:

1. The JOB statement specifies the beginning of a job and assigns a name to the job.
2. The EXEC statement specifies the beginning of a job step and includes the RNL syntax checker parameter, which identifies the programs to be executed.
3. PARM specifies the GRSRNL two-character suffix ('xx,yy') of the parmlib member(s). (You can specify up to 74 characters or 25 member suffixes including commas or blanks. This example specifies four parmlib members.)
4. The SETLIB DD statement specifies the private library that contains the ISGRNLCK load module.
5. The PARMLIB DD statement specifies the SYS1.PARMLIB that contains the GRSRNL parmlib members.

Executing the RNL Syntax Checker in Foreground Mode

To execute the job in the foreground under TSO/E, use the following JCL. (Underscores indicate values that you can replace with information specific to your installation.)

```
ALLOCATE DSN(*) FILE(SYSPRINT)
ALLOCATE DSN(*) FILE(SYSABEND)
ALLOCATE DD (PARMLIB) DSN(SYS1.PARMLIB)
CALL RNLCHECK(ISGRNLCK) 'OA,OB,OC,OD'
```

Output from the RNL Syntax Checker

The following example shows how to execute the RNL syntax checker as a batch job and shows the results. The JCL specifies that the RNL syntax checker verify seven GRSRNL members in SYS1.PARMLIB:

```
//CHECKRNL JOB
//         EXEC,PGM=ISGRNLCK,PARM='87,4D,8B,08,0E,23,04'
//STEPLIB  DD  DSN=RNLCHECK.LOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSABEND DD  SYSOUT=A
//PARMLIB  DD  DSN=SYS1.PARMLIB,DISP=SHR
/*
```

Output from the syntax checker lists the contents of each member and messages that describe any errors the syntax checker detects. (In this example, the information that appears in the boxes describes each member but does not appear in the output.)

GRSRNL87

There are valid comments between keywords and options (RNLCHK38 issued).

```
RNLDEF RNL /* RNL */ (EXCL) QNAME /* QNAME */ (SYSDSN) TYPE /* TYPE
     */ (SPECIFIC)

RNAME /* RNAME */ (SYS1.SVCLIB)

RNLCHK38 (I) - MEMBER GRSRNL87 WAS SUCCESSFULLY PARSED
```

GRSRNL4D

An RNAME with lower case and special characters is valid (RNLCHK38 issued).

```
  RNLDEF RNL (EXCL) TYPE(SPECIFIC)

  QNAME(SYSDSN)

  RNAME('@# $%¢&*()_+abcdefABCDEF')

  RNLCHK38 (I) - MEMBER GRSRNL4D WAS SUCCESSFULLY PARSED
```

GRSRNL8B

There is a valid single parameter for GRS testing (RNLCHK28 and RNLCHK38 issued).

```
RNLDEF LINKLIB(YES)

RNLCHK28 (I) - LINKLIB (YES) SPECIFIED, IGNORE SUBSEQUENT ERROR(s).

RNLCHK38 (I) - MEMBER GRSRNL8B WAS SUCCESSFULLY PARSED
```

GRSRNL08

For each RNL statement there is a valid single entry (RNLCHK38 issued).

```
RNLDEF RNL (INCL) TYPE(GENERIC)

QNAME(SYSDSN)

RNLDEF RNL (EXCL) TYPE(GENERIC)

QNAME(SYSDSN)

RNAME(SYS.DUMP)

RNLDEF RNL (CON) TYPE (SPECIFIC)

QNAME(SYSDSN)

RNAME(SYS1.PAGE)

RNLCHK38(I) - MEMBER GRSRNL08 WAS SUCCESSFULLY PARSED
```

GRSRNL0E

Hexadecimal specification for QNAME and RNAME is valid for the RNL statement (RNLCHK38 issued).

```
RNLDEF RNL (EXCL) TYPE(SPECIFIC)

QNAME(x'C1C1C3C4C5C6C7C8')

RNAME(X'F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF7A7B7C7D7E7F')

RNLCHK38 (I) - MEMBER GRSRNL0E WAS SUCCESSFULLY PARSED
```

GRSRNL23

A keyword is misspelled (RNLCHK19, RNLCHK17, and RNLCHK39 issued).

```
RNLDEF RNL (EXCL) TYPE(SPECIFIC) QNAME(SYSDSN) RNAME

    (SYS1.BROADCAST)

RNLCHK19 (E) - TYPE OPTION NOT SPECIFIED CORRECTLY

RNLCHK17  (E) - SYNTAX ERROR(S)DETERMINED DURING PROCESSING

RNLCHK39  (E) - SYNTAX ERROR SPECIFIED FOR MEMBER GRSRNL23
```

GRSRNL04

There is a duplicate or conflicting keyword in the RNL statement (RNLCHK32, RNLCHK17, and RNLCHK39 issued).

```
RNLDEF TYPE(GENERIC) TYPE(SPECIFIC) RNL(INCL)

QNAME(SYSDSN)

RNAME(SYS1.BRODCAST)

RNLCHK32 (E) - TYPE KEYWORD SPECIFIED MORE THAN ONCE

RNLCHK17 (E) - SYNTAX ERROR(S) DETERMINED DURING PROCESSING

RNLCHK39 (E) - SYNTAX ERROR SPECIFIED FOR MEMBER GRSRNL04
```

Notes:

1. In the output (not shown in the previous example), the RNL syntax checker indicates some errors by an 'X' under the error.
2. When LINKLIB(YES) is followed by a delimiter on an RNLDEF statement, for example,

   ```
   RNLDEF  LINKLIB (YES))
   ```

   The syntax checker verifies the statement but an error occurs during IPL. To prevent such an error, make sure to place a blank between LINKLIB (YES) and the delimiter:

   ```
   RNLDEF  LINKLIB (YES) )
   ```

   See *Initialization and Tuning* for more information about RNLDEF statements.

RNL Syntax Checker Messages

The following messages are issued by the RNL syntax checker and appear in the program listing or on a TSO/E terminal. For information about RNLDEF statements and options specified for GRSRNLxx members of SYS1.PARMLIB, see *Initialization and Tuning.*

The characters in parentheses that follow the message id have the following meanings:

Code	Meaning
I	Informational message; the RNL checker informs you of a completed action.
E	Error; the RNL checker has detected a syntax or other error in the statement.
T	Terminating error; the RNL checker is unable to continue processing.

RNLCHK01 (T) UNABLE TO OPEN SYSPRINT FILE

Programmer Response: Specify the SYSPRINT DD statement in the JCL that invokes the syntax checker, or allocate SYSPRINT FILE on TSO/E. (The syntax checker issues a return code of 20 in register 15.)

RNLCHK02 (I) PARMLIB COULD NOT BE OPENED

Programmer Response: Specify the PARMLIB DD statement on the JCL that invokes the syntax checker, or allocate PARMLIB FILE on TSO/E. (The syntax checker terminates with a return code of 20 in register 15.)

RNLCHK03 (T) NO INPUT. PARMLIB COULD NOT BE OPENED

Programmer Response: Specify the RNL syntax checker parameter for the JCL step that invokes the syntax checker or on TSO/E CALL. (The syntax checker issues a return code of 16 in register 15.)

RNLCHK04 (I) NO ERRORS DETECTED IN PARM STRING

Explanation: The values specified for the RNL syntax checker parameter are valid.

RNLCHK05 (T) PARM STRING IS TOO LONG. MAXIMUM ACCEPTABLE LENGTH IS 74.

Programmer Response: Specify values for the RNL syntax checker parameter that are from 1 to 74 characters. (The syntax checker terminates with a return code of 12 in register 15.)

RNLCHK06 (I) No PARM = INPUT

Explanation: No values were specified for the RNL syntax checker parameter. The syntax checker also issues RNLCHK10 to indicate that it is using the default member.

Programmer Response: Specify one or more values for the RNL syntax checker parameter, or use the default value GRSRNL00.

RNLCHK07 (I) INPUT PARM STRING IS: suffix

Explanation: For **suffix**, the RNL syntax checker indicates the suffix character(s) of the SYS1.PARMLIB members specified for the syntax checker parameter.

RNLCHK08 (T) INVALID CHARACTER IN PARM STRING

Programmer Response: Specify a two-character suffix for the SYS1.PARMLIB member name. Characters can be A - Z and 0 - 9. (The syntax checker terminates with a return code of 12 in register 15.)

RNLCHK09 (T) SUFFIX TOO LONG. MAXIMUM ACCEPTABLE LENGTH IS 2.

Programmer Response: Specify a two-character suffix for the SYS1.PARMLIB member name. Characters can be A - Z and 0 - 9. (The syntax checker terminates with a return code of 12 in register 15.)

RNLCHK10 (I) USING DEFAULT MEMBER GRSRNL00

Explanation: No members were specified for the RNL syntax checker parameter. The system uses the SYS1.PARMLIB default member GRSRNL00. This message is issued after RNLCHK06.

RNLCHK11 (T) THE PARM = STRING CONTAINS MORE THAN 25 UNIQUE SUFFIXES

Explanation: Specify from one to 25 unique member name suffixes for the RNL syntax checker parameter. (The syntax checker terminates with a return code of 12 in register 15.)

RNLCHK12 (I) SUFFIX '--' SPECIFIED MORE THAN ONCE. IT WILL BE READ ONCE.

Explanation: A member name suffix for the RNL syntax checker parameter was specified more than once. The RNL syntax checker reads the member only once.

RNLCHK13 (E) MEMBER GRSRNL-- NOT FOUND IN DATASET NAMED ON PARMLIB DD

Explanation: SYS1.PARMLIB member GRSRNLxx was not found; the return code from the FIND macro is 4.

For information about the FIND macro, see the data administration macro reference book for your version of DFP. (The syntax checker issues a return code of 8 in register 15.)

Programmer Response: Update SYS1.PARMLIB to include the specified member, or modify the RNL syntax checker parameter.

RNLCHK14 (E) SYSTEM ERROR WHILE SEARCHING FOR MEMBER GRSRNL

Explanation: SYS1.PARMLIB member GRSRNLxx was not found; the return code from the FIND macro is 8.

For information about the FIND macro, see the data administration macro reference book for your version of DFP. (The syntax checker issues a return code of 8 in register 15.)

Programmer Response: Update SYS1.PARMLIB with the specified member, or investigate the I/O error.

RNLCHK15 (T) I/O ERROR CAUSED BY GET OR PUT MACRO INSTRUCTION

Explanation: The RNL syntax checker could not retrieve the SYS1.PARMLIB member. (The SYNAD exit specified on the DCB macro received control because of an unrecoverable error.)

Programmer Response: Look at the SYNADRTN routine in ISGRNLCK of the RNL syntax checker program to verify the input. For information about the SYNAD exit, see the data administration macro reference book for your version of DFP.

RNLCHK16 (E) END OF FILE WHILE INSIDE A COMMENT

Programmer Response: Specify a delimiter (*/) to end a comment for an RNLDEF statement .

RNLCHK17 (E) SYNTAX ERROR(S) DETERMINED DURING PROCESSING

Explanation: The RNL syntax checker found a syntax error in the specified RNLDEF statement. Before it issues RNLCHK17, the syntax checker issues any messages that describe the error.

Programmer Response: Correct the error(s) described in the message(s) preceding RNLCHK17.

RNLCHK18 (E) RNL OPTION NOT SPECIFIED CORRECTLY

Explanation: The RNL option or keyword was omitted, misspelled, or not specified on the RNLDEF statement.

Programmer Response: Specify RNL with one following options:

- RNL(EXCL) - for system exclusion
- RNL(INCL) - for system inclusion
- RNL(CON) - for reserve conversion

RNLCHK19 (E) TYPE OPTION NOT SPECIFIED CORRECTLY

Explanation: The TYPE option was omitted, misspelled, or not specified on the RNLDEF statement.

Programmer Response: Specify TYPE with one of the following options:

- TYPE(SPECIFIC) - for a specific QNAME RNAME resource
- TYPE(GENERIC) - for a generic QNAME resource

RNLCHK20 (E) INCORRECT QNAME INPUT OR RNAME INPUT TO PROCESS

Explanation: The QNAME or RNAME option was omitted or misspelled, or a name was not specified.

Programmer Response: Specify QNAME or RNAME followed by a valid qname or rname as follows:

- QNAME(qname)
- RNAME(rname)

RNLCHK21 (E) RNLDEF KEYWORD SPECIFIED AS HEX INPUT STRING

Explanation: The RNL syntax checker found the RNLDEF keyword as part of a hexadecimal value specified for QNAME or RNAME. A closing single quotation or parenthesis is probably missing from QNAME or RNAME.

Programmer Response: Check the hexadecimal value specified for QNAME or RNAME.

RNLCHK22 (E) HEX STRING TOO LONG FOR qname rname

Programmer Response: When you define a non-displayable character or hexadecimal value for QNAME or RNAME, specify a two-digit hexadecimal value to represent each character as follows:

- For QNAME, 1 to 16 two-digit hexadecimal values
- For RNAME, 1 to 510 two-digit hexadecimal values

RNLCHK23 (E) COLUMN 72 NOT BLANK - IT WILL BE IGNORED

Programmer Response: Specify a blank in column 72.

RNLCHK24 (E) ANY CHARACTER STRING NOT SPECIFIED FOR QNAME or RNAME

Explanation: A non-displayable character was specified but not defined as a hexadecimal value. You must specify non-displayable characters as hexadecimal values.

Programmer Response: For QNAME or RNAME, specify non-displayable characters as hexadecimal values:

```
QNAME(x'52')
```

RNLCHK25 (E) INCORRECT HEX INPUT STRING SPECIFIED FOR QNAME OR RNAME

Explanation: An incorrect hexadecimal value has been specified.

Programmer Response: To represent non-displayable characters or valid hexadecimal values, specify only the values 0-9, A-F.

RNLCHK26 (E) INCORRECT STRING SPECIFIED FOR QNAME OR RNAME

Explanation: The RNL syntax checker did not find a parenthesis after the QNAME or RNAME option.

Programmer Response: Enclose the name for QNAME and RNAME in parentheses.

RNLCHK27 (E) RNLDEF KEYWORD NOT SPECIFIED

Explanation: The RNL syntax checker could not find the RNLDEF keyword, or RNLDEF was not the first keyword on the RNL statement.

Programmer Response: Include RNLDEF as the first keyword on the RNL statement.

RNLCHK28 (I) LINKLIB(YES) SPECIFIED, IGNORE SUBSEQUENT ERROR(S)

Explanation: Because LINKLIB(YES) was specified on the RNLDEF statement, the RNL statements in SYS1.PARMLIB are not used during global resource serialization.

RNLCHK29 (E) RNAME KEYWORD NOT SPECIFIED

Programmer Response: When the TYPE option is SPECIFIC, be sure to specify both QNAME (name) and RNAME (name).

RNLCHK30 (E) ANY CHARACTER STRING TOO LONG FOR qname rname

Programmer Response: When specifying as part of QNAME or RNAME displayable characters other than A - Z, 0 - 9,#, @, $, or a period (.), or when specifying a blank as part of the name:

- Enclose the entire name in single quotations
- For QNAME, specify from 1 to 8 characters
- For RNAME, specify from 1 to 255 characters

The following is a valid name:

```
QNAME ('MY RESOURCE!')
```

RNLCHK31 (E) COMMON CHARACTER STRING TOO LONG for qname rname

Programmer Response: When specifying only displayable characters A - Z, 0 - 9, #, @, $, or a period (.):

- For QNAME, specify from 1 to 8 characters
- For RNAME, specify from 1 to 255 characters

RNLCHK32 (E) TYPE KEYWORD SPECIFIED MORE THAN ONCE

Programmer Response: Specify only one TYPE keyword for each RNLDEF statement.

RNLCHK33 (E) RNL KEYWORD SPECIFIED MORE THAN ONCE

Programmer Response: Specify only one RNL keyword for each RNLDEF statement.

RNLCHK34 (E) QNAME KEYWORD SPECIFIED MORE THAN ONCE

Programmer Response: Specify only one QNAME for each RNLDEF statement.

RNLCHK35 (E) RNAME KEYWORD SPECIFIED MORE THAN ONCE

Programmer Response: Specify only one RNAME for each RNLDEF statement.

RNLCHK36 (E) MANDATORY KEYWORDS NOT SPECIFIED (QNAME, TYPE, OR RNL)

Programmer Response: You must specify QNAME, TYPE, and RNL keywords on each RNL statement. (If TYPE is SPECIFIC, you must also specify RNAME.)

RNLCHK37 (I) SPECIFIED RNL ENTRY (ENTRIES) BUILT

Explanation: An internal error might have occurred while building RNL entries.

Programmer Response: Call your IBM representative for support.

RNLCHK38 (I) GRSRNLxx WAS SUCCESSFULLY PARSED

Explanation: The RNL syntax checker did not find any errors in the RNL statements for the specified GRSRNL member. (The syntax checker issues a return code of 0 in register 15.)

RNLCHK39 (E) SYNTAX ERROR SPECIFIED FOR MEMBER GRSRNLxx

Explanation: The RNL utility found syntax errors in the RNL statement(s) for the specified GRSRNL member. Before it issues this message, the syntax checker issues RNLCHK17 for each RNLDEF statement in error and the message that describes the error.

RNLCHK40 (I) WARNINGS ISSUED, RNL ENTRIES MIGHT WORK

Explanation: ISGRNLPR detected an error when parsing an RNL statement. The syntax checker issues a return code of 4 in register 15.

Programmer Response: Call your IBM representative for support.

RNLCHK41 (E) HEX INPUT STRING MIGHT CONTAIN AN ODD NUMBER OF DIGITS

Programmer Response: To represent each hexadecimal number in QNAME or RNAME, specify a two-digit hexadecimal value. For example, the following specifies four two-digit hexadecimal values:

```
QNAME(x'01234A')
```

RNLCHK42 (E) LENGTH OF ZERO SPECIFIED FOR HEX INPUT STRING

Explanation: Hexadecimal values were indicated for QNAME or RNAME, but no values were specified. For example, the following is incorrect:

```
QNAME(x'')
```

Programmer Response: Specify valid hexadecimal values (A - Z, 0 - 9) for QNAME or RNAME.

RNLCHK43 (E) RNLDEF KEYWORD SPECIFIED AS ANY CHARACTER INPUT STRING

Explanation: The RNL syntax checker detected the RNLDEF keyword as part of QNAME or RNAME. A closing quotation or parenthesis is probably missing from QNAME or RNAME.

RNLCHK44 (E) RNLDEF KEYWORD SPECIFIED AS COMMON CHARACTER STRING

Explanation: The RNL syntax checker detected the RNLDEF keyword as part of QNAME or RNAME. A closing quotation or parenthesis is probably missing from QNAME or RNAME.

RNLCHK45 (E) LENGTH OF ZERO SPECIFIED FOR COMMON CHARACTER STRING

Explanation: No displayable characters were specified for QNAME or RNAME:

```
QNAME()
```

Programmer Response: Specify valid displayable characters (A - Z, 0 - 9, #, @, $, or a period with no blanks) for QNAME or RNAME.

RNLCHK46 (E) LENGTH OF ZERO SPECIFIED FOR ANY CHARACTER STRING

Explanation: No displayable characters were specified within quotes for QNAME or RNAME. For example, the following is incorrect:

```
QNAME('')
```

Programmer Response: Specify valid displayable characters. For displayable characters other than A - Z, 0 - 9, #, @, $, or a period, or to include blanks as part of QNAME or RNAME, enclose the entire name in single quotes for QNAME or RNAME.

RNLCHK47 (E) LINKLIB OPTION NOT SPECIFIED CORRECTLY

Programmer Response: Specify either LINKLIB(YES) or LINKLIB(NO).

RNLCHK48 (E) LINKLIB KEYWORD SPECIFIED MORE THAN ONCE

Programmer Response: Specify LINKLIB only once for each RNLDEF statement.

RNLCHK49 (E) ONLY DELIMITER(S) SPECIFIED ON PARM FIELD

Programmer Response: For the RNL syntax checker parameter, you must specify a valid two character GRSRNL member name suffix enclosed in single quotes. (The syntax checker issues a return code of 18 in register 15.)

RNLCHK50 (I) EMPTY MEMBER SPECIFIED, VALID FOR GRS

Explanation: The RNL syntax checker has found an empty SYS1.PARMLIB GRSRNL member. Global resource serialization accepts empty parmlib members at initialization time.

Return Codes

For errors accessing SYS1.PARMLIB or SYSPRINT, the syntax checker issues a message based on one of the following return codes (register 15):

12	SYS1.PARMLIB processing failed. (See messages RNLCHK05, RNLCHK08, RNLCHK09, and RNLCHK11.)
16	SYS1.PARMLIB could not be opened. (See message RNLCHK03.)
18	SYS1.PARMLIB members were not specified on the JCL or TSO/E commands. (See message RNLCHK49.)
20	SYSPRINT data set could not be opened. (See message RNLCHK01 and RNLCHK02.)

For each SYS1.PARMLIB member it processes, the syntax checker issues a message based on the following return codes (register 15):

0	The syntax checker found no errors in the RNL statements. (See message RNLCHK38.)
4	The syntax checker issues a warning message about the RNL statements but continues processing. (See message RNLCHK40.)
8	A system error has occurred while the syntax checker was searching for the member. (See messages RNLCHK13 and RNLCHK14.)
28	The syntax checker has found a syntax error in an RNL statement. for the member. (See messages associated with RNL statement syntax errors.)

Appendix B. RNLs in LINKLIB

A system that includes MVS/SP Version 1 Release 5, MVS/SP Version 2 Release 1.1, or an earlier release, requires you to define the RNLs in LINKLIB. In all other cases, IBM recommends that you define the RNLs in parmlib, as described in "Defining the RNLs" on page 2-19.

An entry in a LINKLIB RNL has the following parts:

- One-byte flag field that contains:
  - X'00' if the entry is a specific name
  - X'40' if the entry is a generic name
  - X'80' if the entry is the last entry
- One-byte rname length field that contains:
  - X'00' if the entry includes only a qname (or if the entry is the last entry)
  - X'nn' if the entry includes an rname, where X'nn' is the length of the rname
- Eight-byte qname field, padded to the right with blanks if the name is shorter than eight bytes, or the end-of-list indicator:
  - RNLESIEX for the SYSTEM inclusion RNL
  - RNLESEEX for the SYSTEMS exclusion RNL
  - RNLERCEX for the RESERVE conversion RNL
- An optional rname field, which can be from 1 to 255 bytes long

The last entry in each RNL must contain the special name; the exit search routine uses the name as an end-of-list indicator. When you modify an RNL, make sure to preserve the last entry.

Global resource serialization requires all three RNLs; you can have an empty RNL, but you cannot have a missing one.

IBM supplies, in member ISGGRNLS of SYS1.ASAMPLIB, a replacement for the RNLs. ISGGRNLS contains many of the entries recommended earlier under "RNL Candidates" on page 2-16. You can use ISGGRNLS as a starting point, modifying the entries to create RNLs that meet your installation's needs. When your RNLs are complete, use the following procedure to replace an RNL.

Replacing the RNLs

All three RNLs reside in the same load module, ISGGRNL0 in SYS1.LINKLIB. The global resource serialization vector table (GVT) contains a pointer to each resource name list.

To include your changes:

1. Use ISGGRNL0 as the csect name of the module; use ISGSIRNL as the label for the SYSTEM inclusion RNL; use ISGSERNL as the label for the SYSTEMS exclusion RNL; and use ISGRCRNL as the label for the RESERVE conversion RNL.
2. Assemble csect ISGGRNL0.
3. Link ISGGRNL0 with the REPLACE option, replacing it with your csect into DISTLIB ALINKLIB, using ISGSIRNL, ISGSERNL, and ISGRCRNL as aliases.
4. Copy ISGGRNL0, ISGSIRNL, ISGSERNL, and ISGRCRNL from DISTILIB ALINKLIB into SYS1.LINKLIB.
5. Update SMPCDS to reflect the change for regression messages.

Once SYS1.LINKLIB on each system includes the new RNLs, all systems must re-IPL before global resource serialization uses the new RNLs. Remember that the RNLs must be the same on all systems in the complex.

Index

Reader's Comments

MVS/ESA
Planning: Global Resource Serialization

MVS/System Product:
JES2 Version 3
JES3 Version 3

Publication No. GC28-1818-2

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Company or Organization

Phone No.

Address

Reader's Comments
GC28-1818-2

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO BOX 950
POUGHKEEPSIE NY 12602-9935

Fold and Tape **Please do not staple** Fold and Tape

GC28-1818-2

Cut or Fold
Along Line

Reader's Comments

MVS/ESA
Planning: Global Resource Serialization

MVS/System Product:
JES2 Version 3
JES3 Version 3

Publication No. GC28-1818-2

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Phone No.

Reader's Comments
GC28-1818-2

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO BOX 950
POUGHKEEPSIE NY 12602-9935

Fold and Tape | **Please do not staple** | Fold and Tape

GC28-1818-2

Cut or Fold
Along Line

Reader's Comments

MVS/ESA
Planning: Global Resource Serialization

MVS/System Product:
JES2 Version 3
JES3 Version 3

Publication No. GC28-1818-2

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Phone No.	

Reader's Comments
GC28-1818-2

IBM®

Cut or Fold
Along Line

Fold and Tape   **Please do not staple**   Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL   PERMIT NO. 40   ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO BOX 950
POUGHKEEPSIE NY 12602-9935

Fold and Tape   **Please do not staple**   Fold and Tape

GC28-1818-2

Cut or Fold
Along Line

Reader's Comments

MVS/ESA
Planning: Global Resource Serialization

MVS/System Product:
JES2 Version 3
JES3 Version 3

Publication No. GC28-1818-2

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Phone No.	

Reader's Comments
GC28-1818-2

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO BOX 950
POUGHKEEPSIE NY 12602-9935

Fold and Tape | **Please do not staple** | Fold and Tape

GC28-1818-2

Cut or Fold
Along Line

Program Number
5685-001
5685-002

File Number
S370-34

GC28-1818-2

Printed in U.S.A.