ABCs of z/OS System Programming: Volume 5

Base and Parallel Sysplex, GRS, RRS, ARM, sysplex failure management

System Logger, z/OS system operation

GDPS, zSeries availability

Paul Rogers

Redbooks

ibm.com/redbooks

International Technical Support Organization

ABCs of z/OS System Programming: Volume 5

May 2011

SG24-6985-02

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

Third Edition (May 2011)

This edition applies to version 1 release 12 modification 0 of IBM z/OS (product number 5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Notices

Trademarks

Preface

The ABCs of z/OS® System Programming is an eleven-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information you need to start your research into z/OS and related subjects. If you would like to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection will serve as a powerful learning tool.

The contents of the volumes are:

Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation

Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, Language Environment®, and SMP/E

Volume 3: Introduction to DFSMS, data set basics, storage management hardware and software, VSAM, System-Managed Storage, catalogs, and DFSMStvs

Volume 4: Communication Server, TCP/IP and VTAM®

Volume 5: Base and Parallel Sysplex®, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, Automatic Restart Management (ARM), Geographically Dispersed Parallel Sysplex™ (GPDS), availability in the zSeries® environment

Volume 6: Introduction to security, RACF®, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries firewall technologies, LDAP, Enterprise identity mapping (EIM), and firewall technologies

Volume 7: Printing in a z/OS environment, Infoprint Server and Infoprint Central

Volume 8: An introduction to z/OS problem diagnosis

Volume 9: z/OS UNIX® System Services

Volume 10: Introduction to z/Architecture®, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and HMC

Volume 11: Capacity planning, performance management, RMF™, and SMF

Volume 12: WLM

Volume 13: JES3

The team who wrote this book

The current edition of this book was produced by a technical specialist working at the International Technical Support Organization, Poughkeepsie Center.

Paul Rogers is a Consulting IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM® classes worldwide on various aspects of z/OS. Before joining the ITSO 23 years ago, he worked in the IBM Installation Support Center (ISC) in Greenford, England providing JES support for IBM EMEA and the Washington Systems Center. He has worked for IBM for 43 1/2 years.

Previous editions of this book were produced by Paul and other technical specialists from around the world working at the ITSO Poughkeepsie Center.

Alvaro Salla is an IBM retiree who worked for IBM for more than 30 years in large systems. He has co-authored many Redbooks® and spent many years teaching S/360 to S/390®. He has a degree in Chemical Engineering from the University of Sao Paulo, Brazil. Alvaro was an author on both the first and second editions of this book.

The team who wrote the the first edition of this book included Paul Rogers, Alvaro Salla, and the following specialists.

Paola Bari is an Advisory Programmer at the International Technical Support Organization, Poughkeepsie Center. She has 23 years of experience as a systems programmer in OS/390®, z/OS, and Parallel Sysplex, including several years of experience in WebSphere® MQ and WebSphere Application Server.

Luiz Fadel is a Distinguished Engineer working for IBM Brazil. He has over 31 years of experience with the S/390 and zSeries platforms. He is responsible for supporting most zSeries customers in Latin America. He has contributed extensively to several Redbooks. Luiz worked in the ITSO from 1977 through 1980 and from 1988 through 1991.

Andreas Horn is an IT Specialist in the ITS Technical Support Center in Mainz, Germany. He holds a graduate degree in Electronic Engineering and joined IBM in 1999. He supports clients with handling defect and non-defect problems in all base components of z/OS.

Redelf Janssen is an IT Architect in IBM Global Services (Technical Support Services Sales and Consulting) in IBM Bremen, Germany. He holds a degree in Computer Science from the University of Bremen and joined IBM Germany in 1988. His areas of expertise include IBM zSeries, z/OS, and availability management. He has written Redbooks on OS/390 Releases 3, 4, and 10, and was one of the authors of *ABCs of z/OS System Programming Volume 3*.

Valeria Sokal is an MVS™ system programmer at an IBM customer. She has 16 years of experience as a mainframe system programmer.

Thomas Stoeckel is a z/OS SW specialist at the EMEA Backoffice in Mainz, Germany. He has six years of experience in BCP, Parallel Sysplex, RRS, and System Logger. His area of expertise includes z/OS performance analysis. Before joining the z/OS team, Thomas worked for 15 years at the large system SW support center for VM and VSE.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYJF Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Base and Parallel Sysplex

A sysplex has been available since 1990 when it was announced as a platform for an evolving large system computing environment. Sysplex provides a commercial platform that supports the nondisruptive addition of scalable processing capacity, in increments matching the growth of workload requirements for customers, without requiring re-engineering of customer applications or re-partitioning of databases. Sysplex also allows the computing environment to reach almost unimaginable levels of continuous availability or 24 by 7. Sysplex technology is built on existing data processing skills and runs existing applications with additional cost savings.

A sysplex (SYStems comPLEX) is not a single product that you install in your data center. A sysplex is a collection of z/OS systems that cooperate, using certain hardware, software, and microcode, to process workloads, provide higher continuous availability, easier systems management, and improved growth potential over a conventional computer system of comparable processing power. This chapter is an overview of a sysplex, including the following topics:

- Sysplex benefits, evolution, and philosophy
- Required software and hardware for sysplex
- Coupling facility
- The z/OS sysplex services components: XCF and XES
- Several types of couple data set (CDS)
- The sysplex configurations: base sysplex and Parallel Sysplex
- Sysplex exploiters
- Coupling facility structure rebuild and duplexing
- Coupling facility configuration and availability
- An overview of settings for sysplex
- Consoles and sysplex management

1.1 Evolution to a Parallel Sysplex

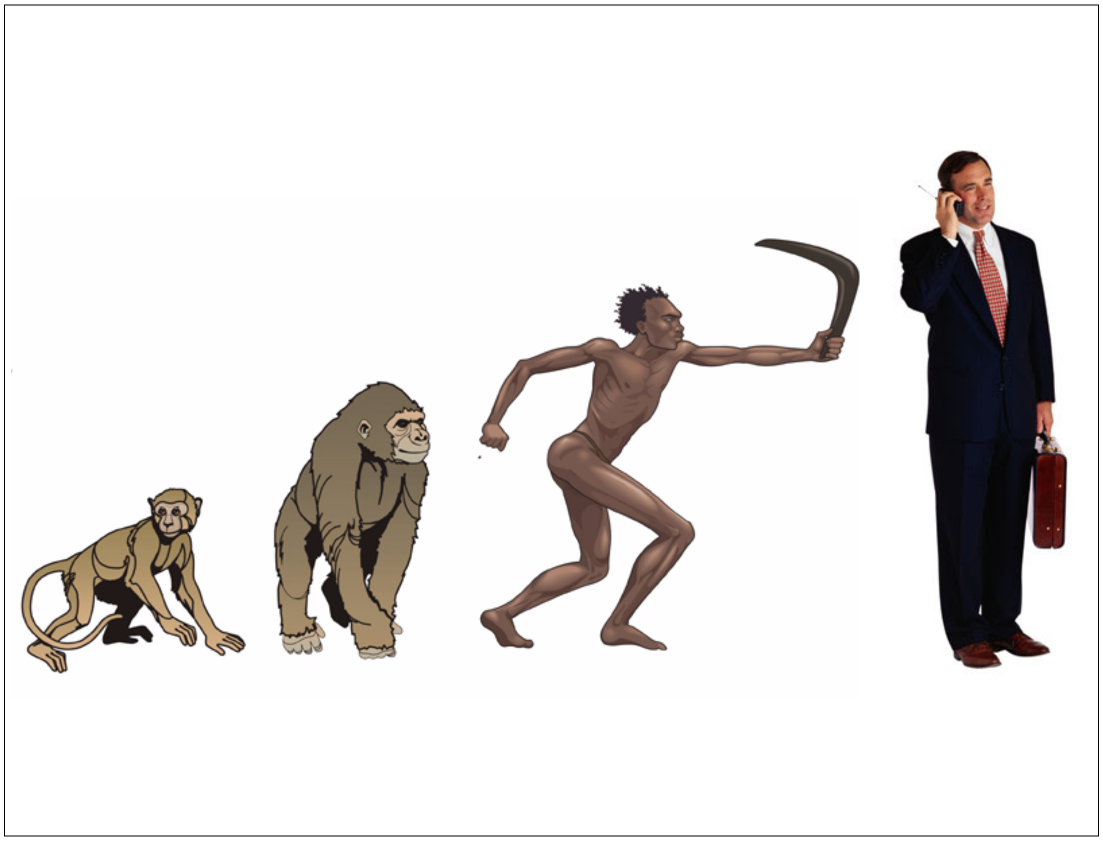


Figure 1-1 Evolution to a Parallel Sysplex

Evolution to a Parallel Sysplex

A Parallel Sysplex is the most recent development in the evolution of IBM large systems. Large system configurations have evolved from a single system to a Parallel Sysplex in the following progression:

- **Single system uniprocessor**

  A single copy of the z/OS (or its ancestor) operating system manages the processing of a central processor complex (CPC) that has a single Central Processor (CP), also called a CPU.

- **Tightly coupled multiprocessors**

  A single copy (also called an image) z/OS operating system manages more than one CP sharing the same central storage, thus allowing several transactions to have their programs executed in parallel.

- **Loosely coupled configuration**

  This configuration is when more than one CPC, possibly tightly coupled multiprocessors, share DASD but not central storage. The CPCs are connected by channel-to-channel communications and are managed by more than one z/OS images.

- **Base sysplex**

  A base sysplex is similar to loosely coupled but the z/OS system and applications programs use a standard communication mechanism for exchanging messages called XCF. This configuration makes the management easier because it provides a greater degree of communication and cooperation among systems and a more unified system image with a single z/OS console group to manage all components.

- **Parallel Sysplex**

  A Parallel Sysplex has up to 32 z/OS systems, as in a base sysplex, but it contains a coupling facility (CF). The coupling facility is a global and intelligent memory that provides multisystem data sharing (with total integrity), workload balancing, high performance communication, and many other advantages.

Table 1 on page 3 summarizes the characteristics in terms of availability, capacity, and system management according to the system configuration.

Table 1 Evolution to a Parallel Sysplex

	Capacity	**Continuous availability**	**Systems management**
Single system uniprocessor	Limited by the size of the largest single CP.	Single points of failure and disruptive changes.	Easy.
Tightly coupled multiprocessors	Limited by the maximum number of CPs in the CPC.	Single points of failure and disruptive changes.	Easy.
Loosely coupled configuration	Increased over tightly coupled.	Increased over tightly coupled.	Each system must be managed separately. Complexity grows with the number of systems.
Base sysplex	Same as loosely coupled.	Better than loosely coupled because of the cooperation.	Single z/OS console group to manage all components.
Parallel Sysplex	Ability to add incremental capacity to match workload growth.	Total (24 by 7), if there are not multiple concurrent failures.	Multisystem data-sharing capability, multisystem workload balancing, enhanced single-system image.

1.2 SYStems comPLEX or sysplex

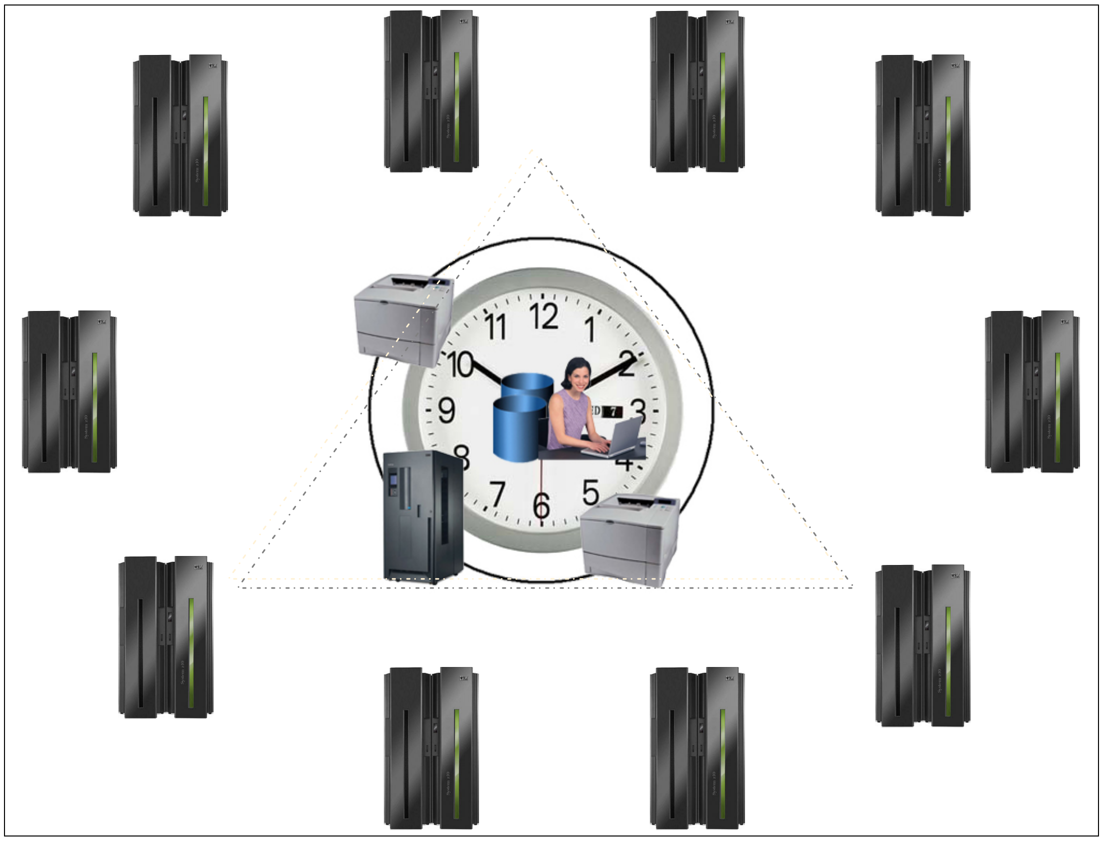


Figure 1-2 Systems complex or sysplex

SYStems comPLEX or sysplex

Parallel and clustered systems initially found in numerically intensive markets (engineering and scientific) have gained increasing acceptance in commercial segments as well. The architectural elements of these systems span a broad spectrum that includes massively parallel processors that focus on high performance for numerically intensive workloads, and cluster operating systems that deliver high system availability.

Parallel Sysplex clustering

Parallel Sysplex clustering contains innovative multisystem data-sharing technology, allowing direct concurrent read/write access to shared data from all processing images in a parallel configuration, without sacrificing performance or data integrity. Each image is able to concurrently cache shared data in a global electronic memory through hardware-assisted cluster-wide serialization and coherency controls. This in turn enables work requests associated with a single workload, such as business transactions or database queries, to be dynamically distributed for parallel execution on nodes in a sysplex cluster, based on available processor capacity. Through this state-of-the-art cluster technology, the power of multiple z/Series processors can be harnessed to work in concert on common workloads, taking the commercial strengths of the z/OS platform to improved levels of competitive price performance, scalable growth, and continuous availability. Prior to the Parallel Sysplex, S/390 (now called System z®) customers had been forced to contain capacity requirements of a workload within technology limits imposed by the largest single symmetric multiprocessor available (symmetric meaning all CPUs are the same).

1.3 The sysplex symmetry

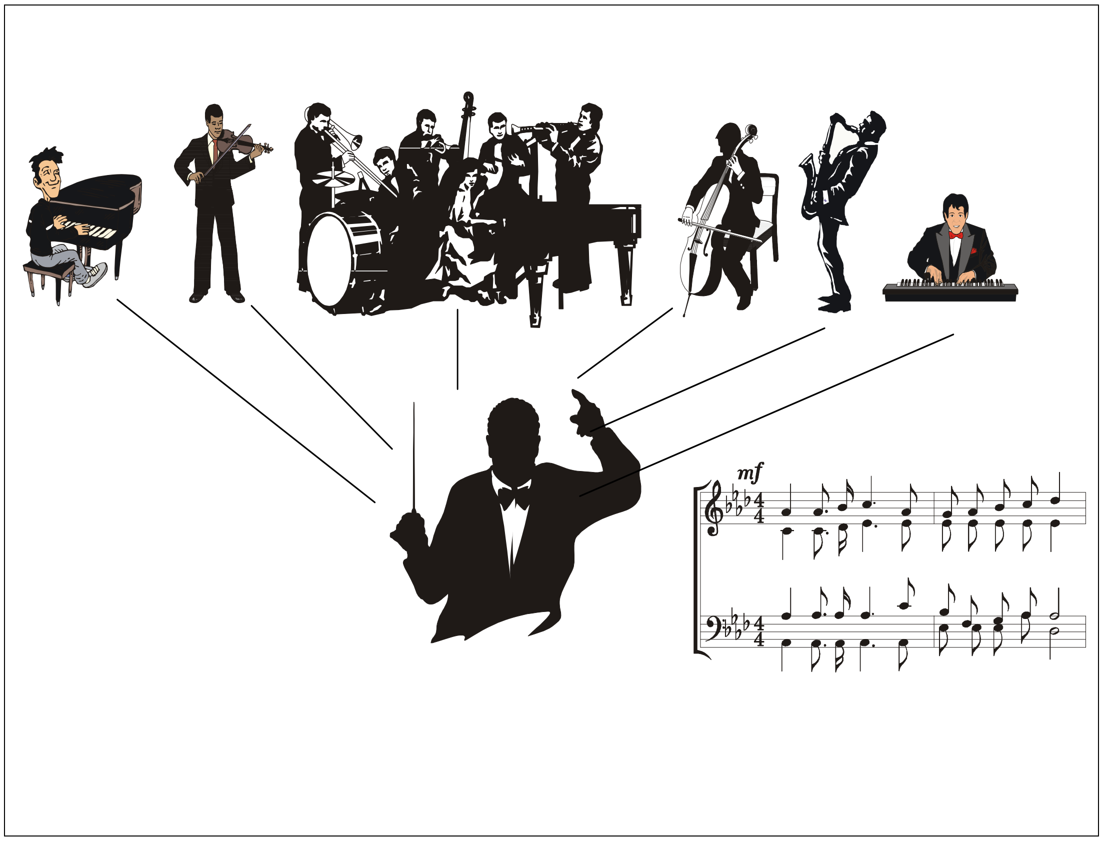


Figure 1-3 The sysplex symmetry

The sysplex symmetry

You can think of a sysplex as a symphony orchestra. The orchestra consists of violins, flutes, oboes, and so on. Think of each instrument as representing a different product (or component) in the sysplex. The fact that you have several of each instrument corresponds to having several images of the same product in the sysplex.

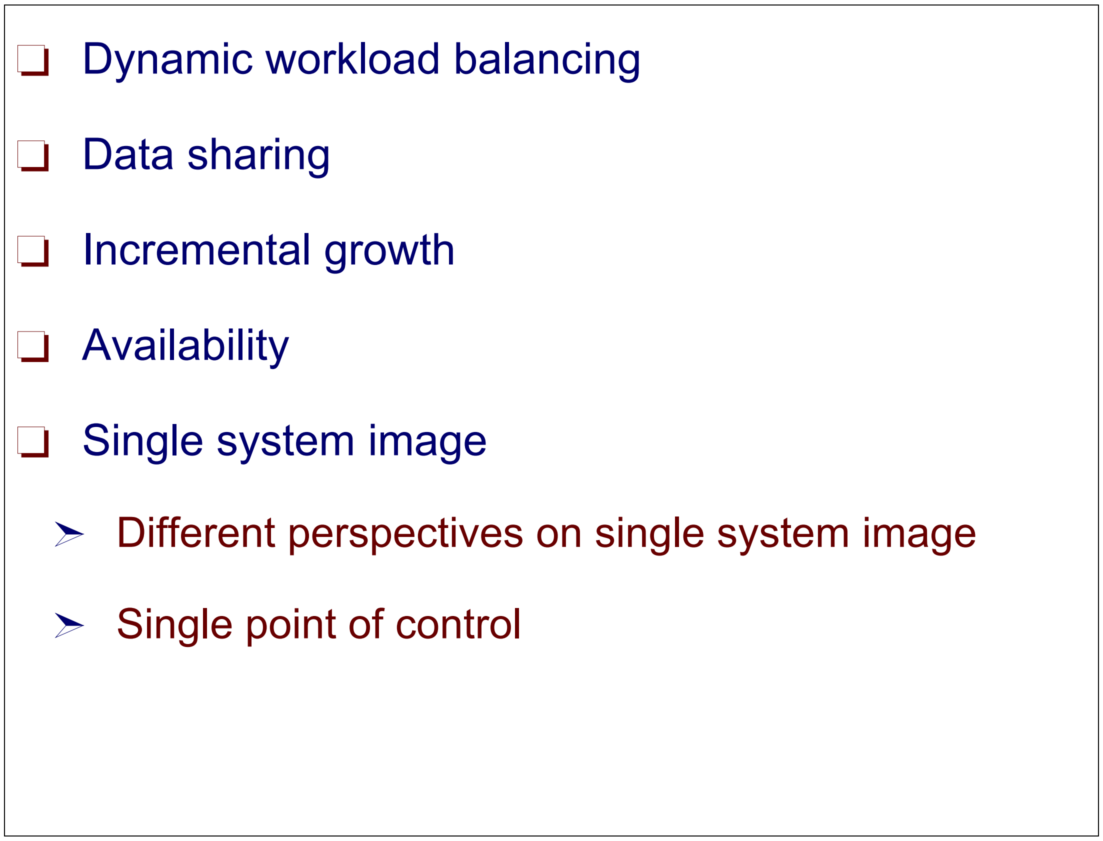
Think of symmetry in the orchestra in the following ways:

- All the violins (or whatever instrument) sound basically the same, and play the same musical part.
- All the instruments in the orchestra share the same musical score. Each instrument plays the appropriate part for that instrument.

Similarly in the sysplex, you can make all the systems, or a subset of them, look alike (clone systems) and do the same work. All the systems can access the same database, and the same library of programs, each one using the information it needs at any point in time. The concept of symmetry allows new systems to be easily introduced, and permits automatic workload distribution all the time, even in the event of failure or when an individual system is scheduled for maintenance. Symmetry also significantly reduces the amount of work required by the systems programmer in setting up the environment.

In an asymmetric sysplex, each system has its own software and hardware configurations, so some of the system management benefits of being in a sysplex are lost.

1.4 Sysplex philosophy



- Dynamic workload balancing
- Data sharing
- Incremental growth
- Availability
- Single system image
  - Different perspectives on single system image
  - Single point of control

Figure 1-4 Sysplex philosophy

Sysplex philosophy

A new violinist who joins the symphony orchestra receives a copy of the score, and begins playing with the other violinists. The new violinist has received a share of the workload. Similarly in a sysplex, if you add a system, the Customer Information Control System (CICS®) OLTP transaction workload can be automatically rebalanced so that the new system gets its share, provided you have set up the correct definitions in your sysplex.

Dynamic workload balancing

Theoretically, in the CICS OLTP environment, transactions coming into the sysplex for processing can be routed to any system. CICS uses the z/OS component called Workload Manager (WLM), along with CICSPlex® System Manager (CPSM), to dynamically route CICS transactions. For this to happen in the sysplex, you need symmetry; the systems across which you want to automatically balance the workload must have access to the same data, and have the same applications that are necessary to run the workload. In other words, no system affinity should be allowed.

Data sharing

We noted earlier that in the symphony orchestra, all the instruments share the same musical score, each playing the appropriate part. You can think of the musical score as a kind of database. Part of the definition of symmetry, as used in this book, is systems sharing the same resources. An important resource for systems to share is data and programs, either in the form of a database, or data sets. Symmetry through systems sharing the same database and program libraries facilitates dynamic workload balancing and availability. The coupling facility technology, together with the support in the database managers, provides the data sharing capability.

You improve application availability by using products that provide data sharing with the coupling facility technology, such as Information Management System Database Manager (IMS™ DB), DB2® Universal Data Base for z/OS (DB2), VSAM RLS, or Transactional VSAM Services (DFSMStvs).

Incremental growth

The conductor can add violins, or other instruments, to the orchestra one by one until the desired effect is achieved. The conductor would not want to hire five more violinists if only two are needed at the moment. A sysplex exhibits the same incremental growth ability. Rather than adding capacity in large chunks, most of which might remain idle, you can add small chunks closer to the size you need at the moment.

Also, the introduction of a new violinist is non disruptive. It is possible (although you might see this only in the most novel of musical pieces) that the violinist could walk onto the stage in the middle of the concert, take a seat, and begin playing with the others. There is no need to stop the concert. Similarly, with a sysplex, because of symmetry and dynamic workload balancing, you can add a system to your sysplex without having to bring down the entire sysplex, and without having to manually rebalance your CICS OLTP workload to include the new system.

Continuous availability

If a violinist gets sick and cannot be present for a given performance, there are enough other violinists so that the absence of one will probably not be noticeable. If a violinist decides to quit the orchestra for good, that violinist can be replaced with another. A sysplex exhibits similar availability characteristics. One of the primary goals of a sysplex is continuous availability. You can think of availability from these perspectives: the availability of your applications programs and the availability of your data.

With symmetry and dynamic workload balancing, your applications can remain continuously available across changes, and your sysplex remains resilient across failures. Adding a system, changing a system, or losing a system should have little or no impact on overall availability. With symmetry and data sharing, using the coupling facility, you also have enhanced database availability.

Automation plays a key role in availability. Typically, automation routines are responsible for bringing up applications, and if something goes wrong, automation handles the application's restart. While automation does not play much of a role in our symphony orchestra, the need for automation is quite important in the sysplex, for availability as well as other reasons.

A facility of z/OS called Automatic Restart Manager (ARM) provides a fast restart and automatic capability for failed subsystems, components, and applications. ARM plays an important part in the availability of key z/OS components and subsystems by decreasing the mean-time-to-repair (MTTR), which in turn affects the availability of data.

For example, when a subsystem such as CICS, IMS DB, or DB2 fails, it might be holding resources, such as locks, that prevent other applications from accessing the data they need. ARM quickly restarts the failed subsystem; the subsystem can then resume processing and release the resources, making data available once again to other applications. Note that System Automation for z/OS (SA z/OS), an IBM product that provides automation of operator functions such as start-up, shutdown, and restart of subsystems, has awareness of z/OS Automatic Restart Manager, so that restart actions are properly coordinated. A sysplex is also the framework that provides a single system image.

1.5 Single system image

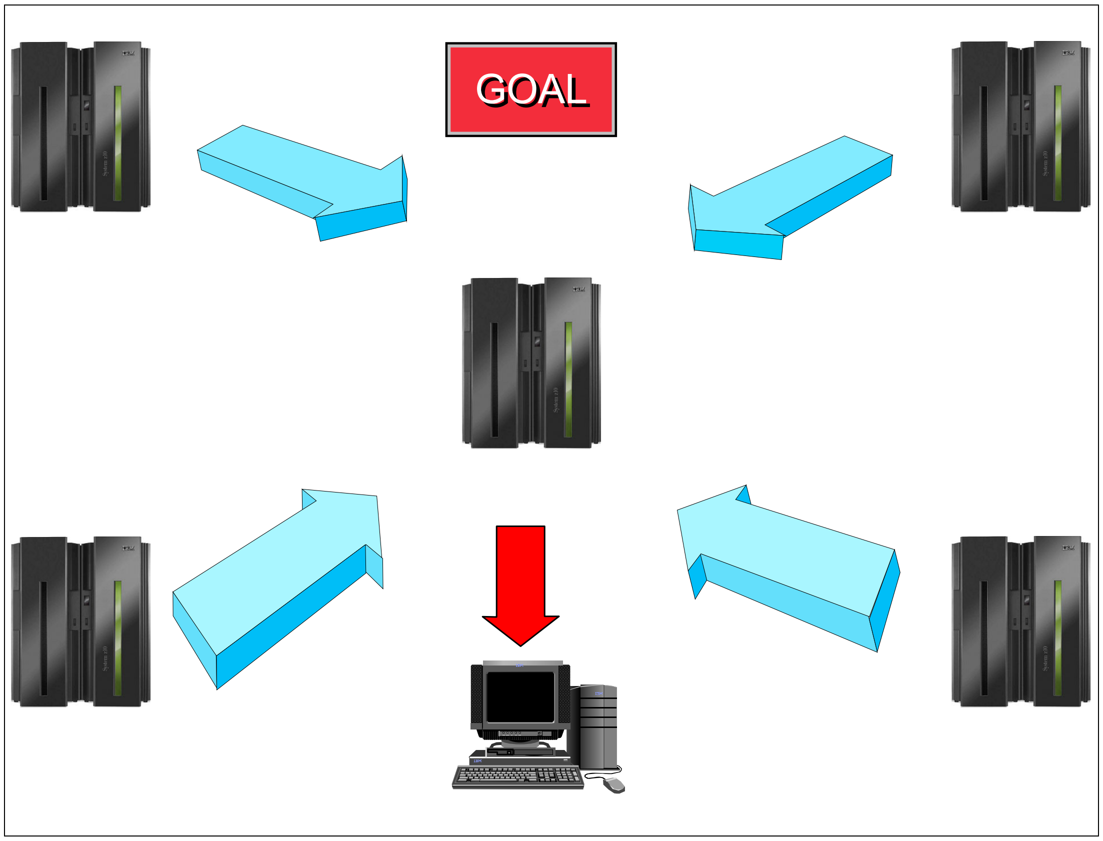


Figure 1-5 Single system image

Single system image

Think of all the violins in the symphony orchestra playing the same part. To the audience, they might sound like one giant violin. The entire orchestra is cooperating to produce the music that the audience hears. In this way, the audience perceives the orchestra as a single entity. This is a good way to picture single system image in the sysplex. You have multiple images of the same product, but they appear, and you interact with them, as one image. The entire sysplex is cooperating to process the workload. In this way, you can think of the collection of systems in the sysplex as a single entity.

Single system image is not a new concept. Many products already provide single system image capability to some degree, or have plans to implement it in the context of commercial enterprise-wide systems management. The important point is, single system image is a key theme in a sysplex. Implementing symmetry in your sysplex facilitates single system image; symmetry facilitates your ability to manage multiple systems in the sysplex as though they were one system. Now, you have the best of two worlds, a logical centralized topology implemented in a physically distributed one.

While single system image is the goal, different IBM and non-IBM products, and even different components within products, are at different stages of development on this issue. Attaining the goal depends on the installation choosing the right options on such products and components.

Different perspectives on single system image

The single system image goal provides different advantages depending on your perspective. The advantage for the end user is the ability to log onto an application in the sysplex, and to be able to access that application without being concerned about which system the application resides on.

For example, CICS uses the VTAM generic resources function, which allows an end user to log on to one of a set of CICS terminal-owning regions (TORs), such as TOR1, TOR2, and TOR3, through a generic name, such as TOR, thus providing single system image for VTAM access to CICS TORs. With dynamic workload management, provided by CICSPlex SM (a CICS component), the logons are then balanced across the CICS TORs; later, when the transactions start to arrive, they will be dynamically distributed through transaction application-owning regions (AORs) that are logically connected to the TOR at logon time. All of this provides a single system image from an application perspective.

The advantage to an operator is the ability to control the sysplex as though it is a single entity. For example, through commands with sysplex-wide scope, operators can control all the z/OS images in the sysplex as though only one z/OS image existed.

When TSO/E is part of a sysplex and exists on multiple sysplex members, you can assign a VTAM generic name to all TSO/E and VTAM application programs. A TSO/E and VTAM application on one z/OS system can be known by the same generic resource as a TSO/E and VTAM application on any other z/OS system. All application programs that share a particular generic name can be concurrently active. This means that a user can log on to a TSO/E generic name in the sysplex rather than to a particular system. The generic name can apply to all systems in the sysplex, or to a subset.

Eventually, when all the necessary products provides single system image capability, the result is greatly improved and simplified enterprise-wide systems management. Both IBM and non-IBM products work towards this goal.

Single point of control

The conductor of the symphony controls the entire orchestra from the podium. The conductor does not stand by the violins and conduct them for a while, and then run over and stand by the flutes to conduct them. In a sysplex, an operator or a systems programmer should be able to control a set of tasks for the sysplex from a given workstation.

The sysplex is a little different from the symphony orchestra in that single point of control in the sysplex does not imply a single universal workstation such as a console. The object is not to control every task for every person from one place. A given individual should be able to accomplish the set of tasks pertinent to that individual’s job from one place.

Ideally, you can have multiple consoles, each tailored to a particular set of tasks; for each such console, you can have either a duplicate of that console, or some other mechanism to ensure that for every task, there is an alternative way to accomplish the task in the event the console or its connection to the sysplex fails.

IBM and non-IBM products are furthering the ability to implement single point of control through integrating operations on a workstation. IBM provides an implementation of an integrated operations console through the Tivoli® Management Environment (TME) 10.

1.6 Parallel Sysplex workload balancing

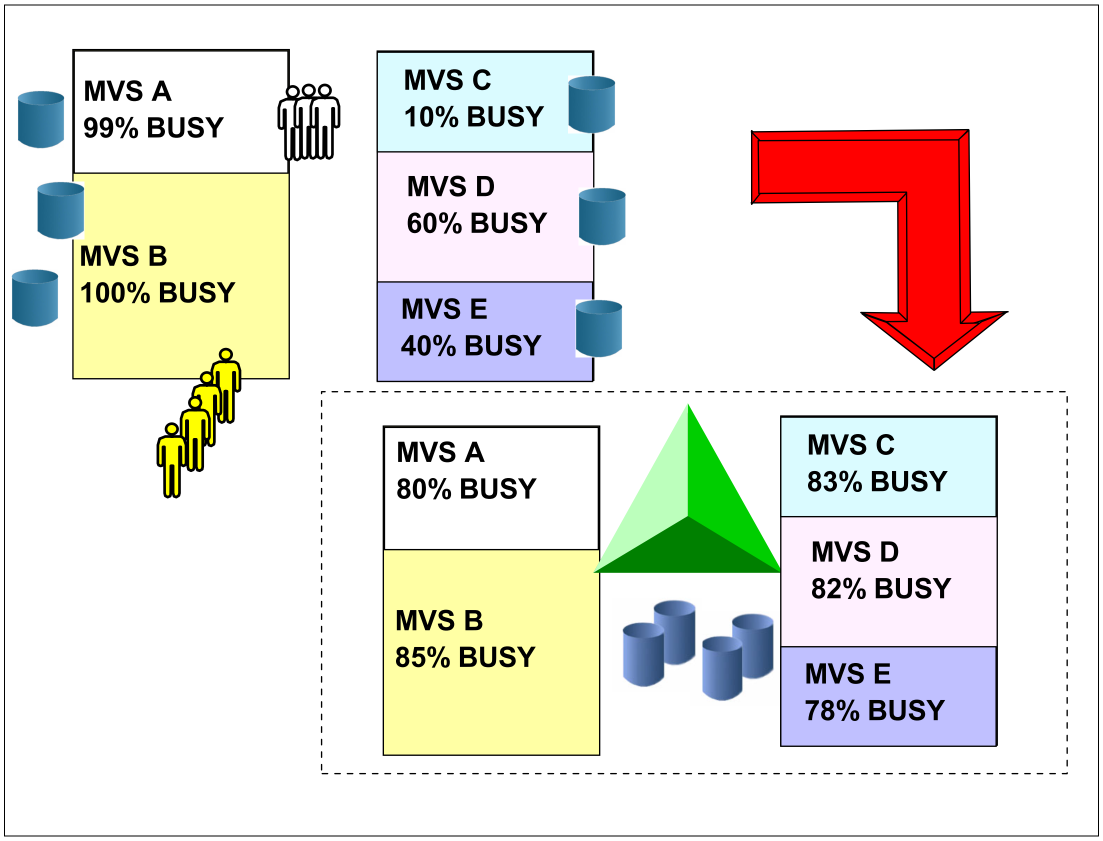


Figure 1-6 Parallel Sysplex workload balancing

Parallel Sysplex workload balancing

You might be wondering what a sysplex could do for you. If your data center is responsible for even one of the following types of work, you could benefit from a sysplex:

- Large business workloads that involve hundreds of end users, or deal with volumes of work that can be counted in millions of transactions per day.
- Work that consists of small work units, such as online transactions, or large work units that can be subdivided into smaller work units, such as queries.
- Concurrent applications on different systems that need to directly access and update a single database without jeopardizing data integrity and security.

Sharing the workload

A sysplex shares the processing of work across z/OS systems and as a result offers benefits such as reduced cost through more cost-effective processor technology using IBM software licensing charges in Parallel Sysplex.

Workload balancing

When you are in an environment with multiple systems, the set of performance issues changes. Existing mechanisms for managing system performance are complex and single-system oriented.

To reduce the complexity of managing a sysplex, MVS workload management provides dynamic sysplex-wide management of system resources. MVS workload management is the combined cooperation of various subsystems (such as CICS, IMS, and VTAM) with the MVS workload manager (WLM) component. An installation defines performance goals and a business importance to workloads through WLM. Workload management focuses on attaining these goals through dynamic resource distribution.

Workload management

A sysplex provides a different way of managing workloads than was previously used. The emphasis is on defining performance goals for work, and having MVS and the subsystems adapt to meet the goals. This provides for the following:

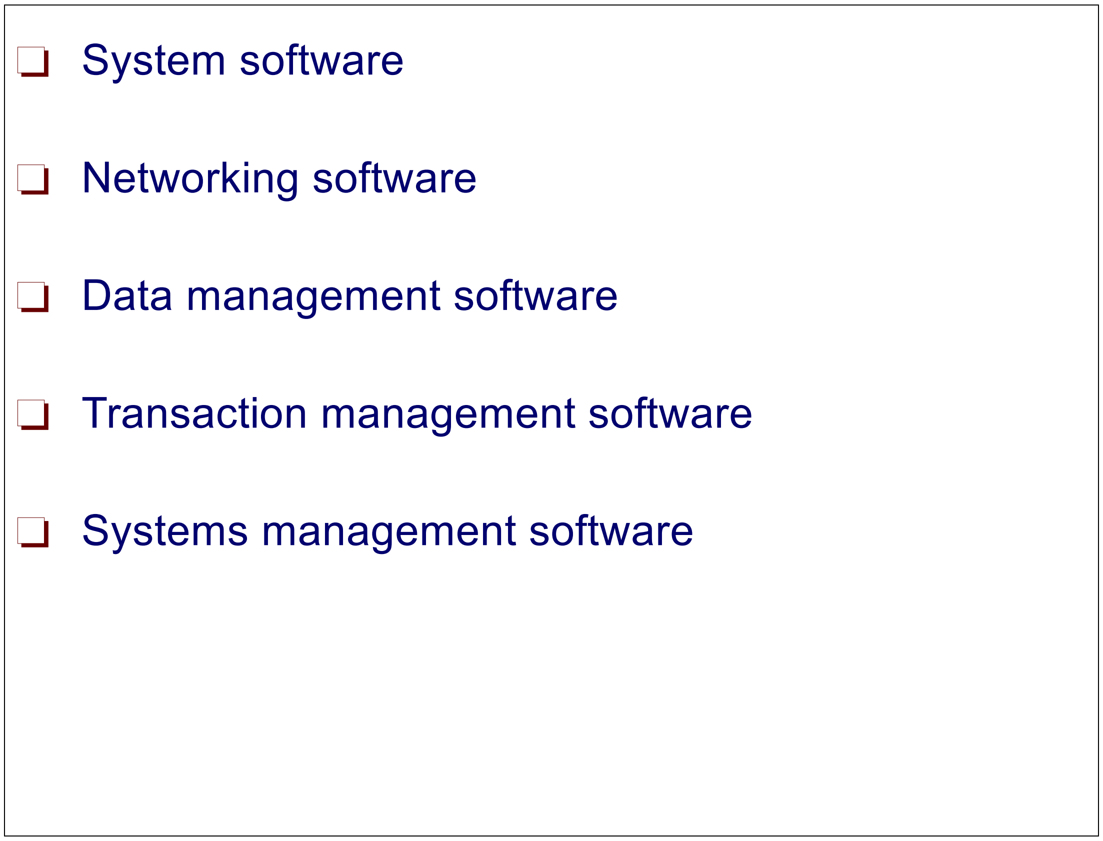
- A platform for continuous availability so that applications can be available 24 hours a day, 7 days a week, 365 days a year (or close to it).
- The ability to do more work to provide greater capacity and an improved ability to manage response time through the use of goals.
- Greater flexibility and the ability to mix levels of hardware and software.
- The ability to dynamically add systems, which allows an easy path for incremental growth.
- Considering resource sharing (of unit tapes, for example), resources are used in the system that needs them, instead of being dedicated. That enables better environment management, performance, and cost savings.
- Data sharing and resource sharing are the fundamental mechanisms that allow installations to have their workload dynamically redistributed between images in the Parallel Sysplex. Workload can be routed to systems where spare capacity exists, avoiding CEC upgrade and still meeting service level objectives, as shown in Figure 1-6.

  For example, CICS is usually implementing the function shipping capability toward the file owner region (FOR) in order to share VSAM files across multiple CICS application regions. Most of the time this FOR region becomes a bottleneck and a single point of failure. In a Parallel Sysplex environment, with data sharing, transactions can be routed to a CICS address space with workload balancing and better response time in the Parallel Sysplex. This requires a transaction management tool, like CICSPlex SM, but the basic framework is the Parallel Sysplex.

Sysplex configurations

A sysplex configuration can be either a base sysplex or a Parallel Sysplex. Later in this chapter, after introducing all the hardware and software required in a sysplex, both configurations are described. The configurations are described in “Base sysplex” on page 51 and “Parallel Sysplex” on page 52.

1.7 Sysplex software



- ❑ System software
- ❑ Networking software
- ❑ Data management software
- ❑ Transaction management software
- ❑ Systems management software

Figure 1-7 Sysplex software

Sysplex software

The following types of software exploit sysplex capabilities:

System	System software is the base software that is enhanced to support a sysplex. It includes the z/OS operating system, JES2 and JES3, and DFSMS.
Networking	Includes Virtual Telecommunications Access Method (VTAM) and Transmission Control Protocol/ Internet Protocol (TCP/IP), which support attachment of a sysplex to a network.
Data management	Data management software includes data managers that support data sharing in a sysplex, such as Information Management System Database Manager (IMS DB), DATA BASE 2 (DB2), and Virtual Storage Access Method (VSAM). We can also include here Adabas and Oracle.
Transaction management	Transaction management software includes transaction managers that support a sysplex such as Customer Information Control System (CICS Transaction Server), Information Management System Transaction Manager (IMS TM) and WebSphere Application Services.
Systems management	Systems management software includes a number of software products that are enhanced to run in a sysplex and exploit its capabilities. The products manage accounting, workload, operations (as DFSMShsm), performance (as RMF), security (as RACF), and configuration (as HCD).

1.8 Sysplex hardware

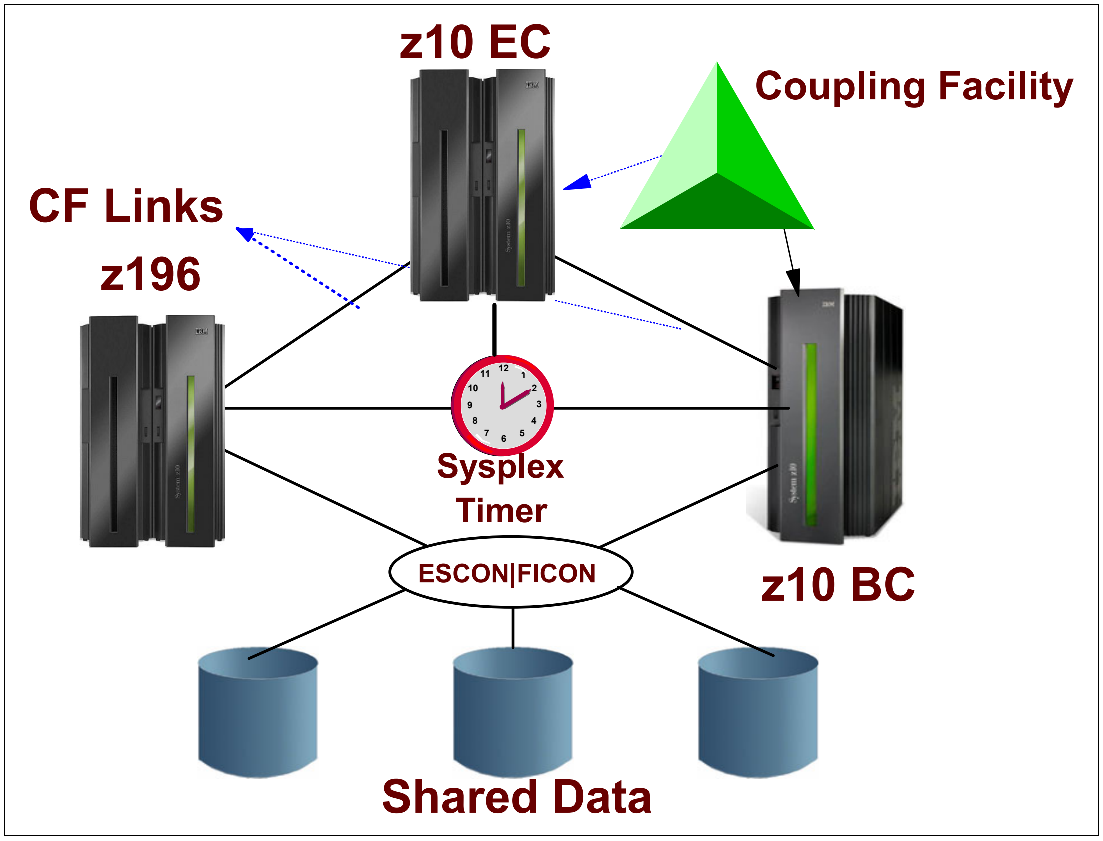


Figure 1-8 Sysplex hardware

Sysplex hardware

A sysplex is a collection of MVS systems that cooperate, using certain hardware and software products, to process work. A conventional large computer system also uses hardware and software products that cooperate to process work. A major difference between a sysplex and a conventional large computer system is the improved growth potential and level of availability in a sysplex. The sysplex increases the number of processing units and MVS operating systems that can cooperate, which in turn increases the amount of work that can be processed. To facilitate this cooperation, new products were created and old products were enhanced. The following types of hardware participate in a sysplex.

System z processors

Selected models of System z processors can take advantage of a sysplex. These include large water-cooled processors, air-cooled processors, and the processors that take advantage of (CMOS) technology.

Coupling facility

Coupling facilities enable high performance multisystem data sharing. Coupling facility links, called channels, provide high speed connectivity between the coupling facility and the central processor complexes that use it. These z/OS systems can be located in the same or in different CPCs that have the coupling facility.

Sysplex Timer

The Sysplex Timer® is an external time reference (ETR) device that synchronizes the time-of-day (TOD) clocks across multiple CPCs in a sysplex. The time stamp from the Sysplex Timer is a way to monitor and sequence events within the sysplex. Server Time Protocol (STP) is a server-wide facility providing capability for multiple z/OSs to maintain TOD time synchronization with each other and form a Coordinated Timing Network (CTN). It is a cost saving replacement for the Sysplex Timer.

FICON and ESCON

The term Fibre Connection (FICON®) represents the architecture as defined by the InterNational Committee of Information Technology Standards (INCITS), and published as ANSI standards. FICON also represents the names of the various System z server I/O features. ESCON® and FICON control units and I/O devices provide the increased connectivity necessary among a greater number of systems.

ESCON channels and directors, and FICON channels and directors (switches), are Enterprise Systems Connection (ESCON) and Fiber Connection (FICON) channels that enhance data access and communication in the sysplex. The ESCON directors and FICON switches add dynamic switching capability for those channels.

FICON is widely used in the System z environment, and provides additional strengths and capabilities compared to the ESCON technology. Many additional capabilities have been included in support of FICON since it was originally introduced. Some control units and control unit functions might require FICON use exclusively. For example, Hyper Parallel Access Volume requires the use of FICON and will not work with ESCON.

1.9 Sysplex Timer

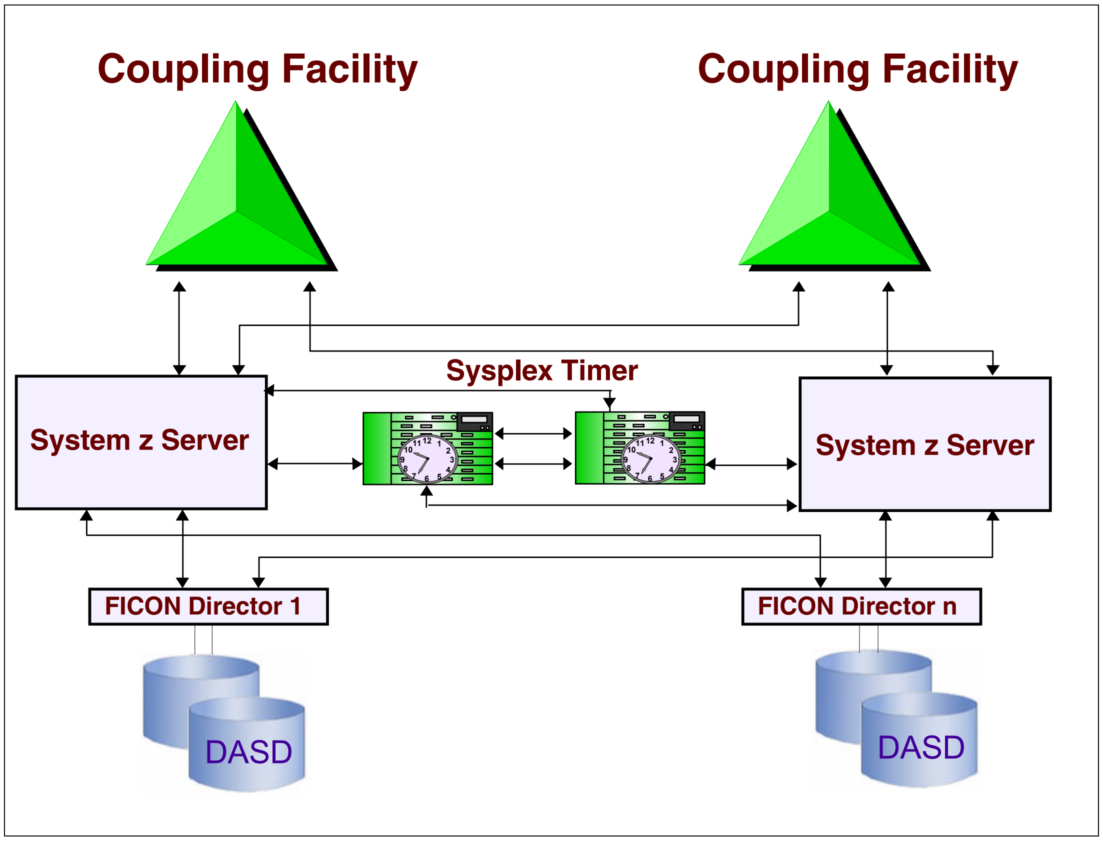


Figure 1-9 Sysplex Timer

Sysplex Timer

Time is a key variable for commercial programs. In physics there are three time scales:

- Universal international time (UIT) based on the earth revolution cycles. Not used for modern purposes due to the variability of such cycles.
- International atomic time (TAI) based on the radioactive properties of Cesium 133.
- Coordinated universal time (UTC) is derived from TAI, but kept not far from UIT by adding or deleting discrete units of leap seconds.

Several System z processors can execute several task programs in a data processing complex. Each of these processors has a time-of-day (TOD) clock, which is an internal 104-bit register incremented by adding a one in bit position 51 every microsecond. TOD clocks use a UTC time scale.

Multiple TOD clocks

When tasks are shared among different CPUs, multiple TOD clocks can be involved. These clocks might be in sync with one another. All CPU TODs from the same CPC are internally synchronized. Then, there is a need for a single time resource, that is, an External Time Reference (ETR) to synchronize the TOD clocks of CPUs located in distinct CPCs running z/OS in the same sysplex. The Sysplex Timer is hardware that is used when the sysplex consists of z/OS systems running in more than one CPC.

Note: Currently, you have the possibility of replacing a Sysplex Timer by Server Time Protocol (STP), as shown in "Server Timer Protocol (STP)" on page 17. The Sysplex Timer provides synchronization for the TOD clocks of multiple CPUs in distinct CPCs, and thereby allows events started by different CPUs to be properly sequenced in time. For example, when multiple CPUs update the same database, all updates are required to be time stamped in proper sequence in the database log.

Timer functions

There is a long-standing requirement for accurate time and date information in data processing. As single operating systems have been replaced by multiple, coupled operating systems on multiple servers, this need has evolved into a requirement for both accurate and consistent clocks among these systems. Clocks are said to be consistent when the difference or offset between them is sufficiently small. An accurate clock is consistent with a standard time source.

The IBM z/Architecture, Server Time Protocol (STP), and External Time Reference (ETR) architecture facilitates the synchronization of server time-of-day clocks to ensure consistent time stamp data across multiple servers and operating systems. The STP or ETR architecture provides a means of synchronizing TOD clocks in different servers with a centralized time reference, which in turn might be set accurately on the basis of an international time standard (External Time Source). The architecture defines a time-signal protocol and a distribution network, which permits accurate setting, maintenance, and consistency of TOD clocks.

External time reference (ETR)

External time reference hardware facility (ETR) is the generic name for IBM Sysplex Timer. The ETR architecture provides a means of synchronizing TOD clocks in different CPCs with a centralized time reference, which in turn can be set accurately on the basis of UTC time standard (External Time Source). The architecture defines a time-signal protocol and a distribution network (called the ETR network) that permits accurate setting, maintenance, and consistency of TOD clocks.

Timing services

Timing services are implemented in z/OS by a time supervisor component. It can be used by an application program to obtain the present date and time, and convert date and time information to various formats. Interval timing lets your program set a time interval to be used in the program logic, specify how much time is left in the interval, or cancel the interval. For programs that are dependent upon synchronized TOD clocks in a multi CPC environment, like a database, it is important that the clocks are in ETR synchronization. These programs can use the STCKSYNC macro to obtain the TOD clock contents and determine if the clock is synchronized with an ETR. STCKSYNC also provides an optional parameter, ETRID, that returns the ID of the ETR source with which the TOD clock is currently synchronized.

ETR attachments

The ETR feature in System z9® and System z10® servers provides the interface to a Network Time Protocol (NTP) server with pulse per second (PPS) support.

Note: The zEnterprise 196 does not support the IBM Sysplex Timer and ETR attachment.

1.10 Server Timer Protocol (STP)

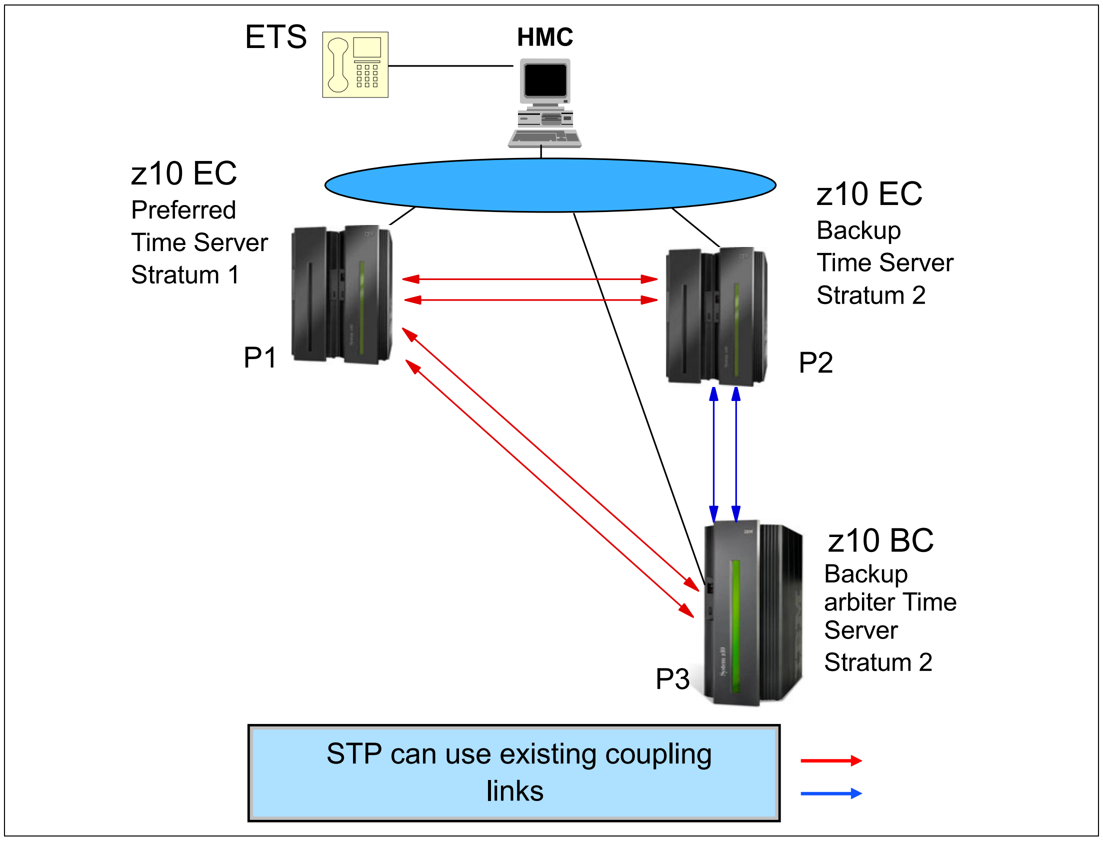


Figure 1-10 Server Time Protocol (STP)

Server Time Protocol (STP)

Server Time Protocol (STP) is designed to help multiple System z servers maintain time synchronization with each other, without the use of a Sysplex Timer. STP uses a message-based protocol in which timekeeping information is passed over externally defined coupling links, such as InterSystem Channel-3 (ISC-3) links configured in peer mode, Integrated Cluster Bus-3 (ICB-3) links, and Integrated Cluster Bus-4 (ICB-4) links. These can be the same links that already are being used in a Parallel Sysplex for coupling facility message communication.

STP is implemented in the Licensed Internal Code (LIC) of System z servers and CFs for presenting a single view of time to PR/SM™.

Note: A time synchronization mechanism, either IBM Sysplex Timer or Server Time Protocol (STP), is a mandatory hardware requirement for a Parallel Sysplex environment consisting of more than one server.

STP link

An STP link is a coupling facility connection that serves as a timing-only link. With STP links, you can allow multiple servers to form a Coordinated Timing Network (CTN), which is a collection of servers and coupling facilities that are synchronized to a time value called Coordinated Server Time. Establishing an STP link between two processors does not require a CF partition; an STP link can be established between two OS partitions. For an STP link, HCD generates a control unit of type “STP” on both sides of the connection. No devices are defined.

This server-wide facility provides capability for multiple z/OS systems to maintain TOD synchronization with each other and form a Coordinated Timing Network (CTN), that is, a collection of z/OS systems that are time synchronized to a time value. In Figure 1-10, the preferred time is as follows:

- P1 is the z/OS preferred timer server (stratum 1) that synchronizes the TODs of the z/OS systems (stratum 2).
- P2 is the backup time server, ready to replace P1.
- P3 is the arbiter time server that decides when the replacement should be done.

The External Time Reference connections are replaced by the implementation of STP, which makes use of coupling links to pass timing messages to the servers. Transition to STP makes it possible to have a Mixed Coordinated Network configuration. The Sysplex Timer provides the timekeeping information in a Mixed CTN. Once an STP-only configuration is established, the ETR connections are no longer needed. STP allows coexistence with Sysplex Timer in mixed configurations. The Sysplex Timer console is replaced by an HMC screen for each possible time zone.

Note: A z196 cannot be connected to a Sysplex Timer; consider migrating to an STP-only Coordinated Time Network (CTN) for existing environments. It is possible to have a z196 as a Stratum 2 or Stratum 3 server in a Mixed CTN, as long as there are at least two System z10 or System z9 servers attached to the Sysplex Timer operating as Stratum 1 servers.

Defining STP links in a sysplex

Server Time Protocol is designed to help multiple System z servers maintain time synchronization with each other, without the use of a Sysplex Timer. STP uses a message-based protocol in which timekeeping information is passed over externally defined coupling links, such as InterSystem Channel-3 (ISC-3) links configured in peer mode, Integrated Cluster Bus-3 (ICB-3) links, and Integrated Cluster Bus-4 (ICB-4) links. These can be the same links that already are being used in a Parallel Sysplex for coupling facility message communication.

An STP link is a coupling facility connection that serves as a timing-only link. With STP links, you can allow multiple servers to form a Coordinated Timing Network (CTN), which is a collection of servers and coupling facilities that are synchronized to a time value called Coordinated Server Time. Establishing an STP link between two processors does not require a CF partition; an STP link can be established between two OS partitions. For an STP link, HCD generates a control unit of type `STP` on each side of the connection. No devices are defined.

You can establish an STP link between two System z servers (z890, z990, z9 EC, or later). In the Connect to CF Channel Path dialog, select two CHPIDs defined for coupling facilities, and then specify the Timing-only link option to create an STP link.

1.11 Coupling facility

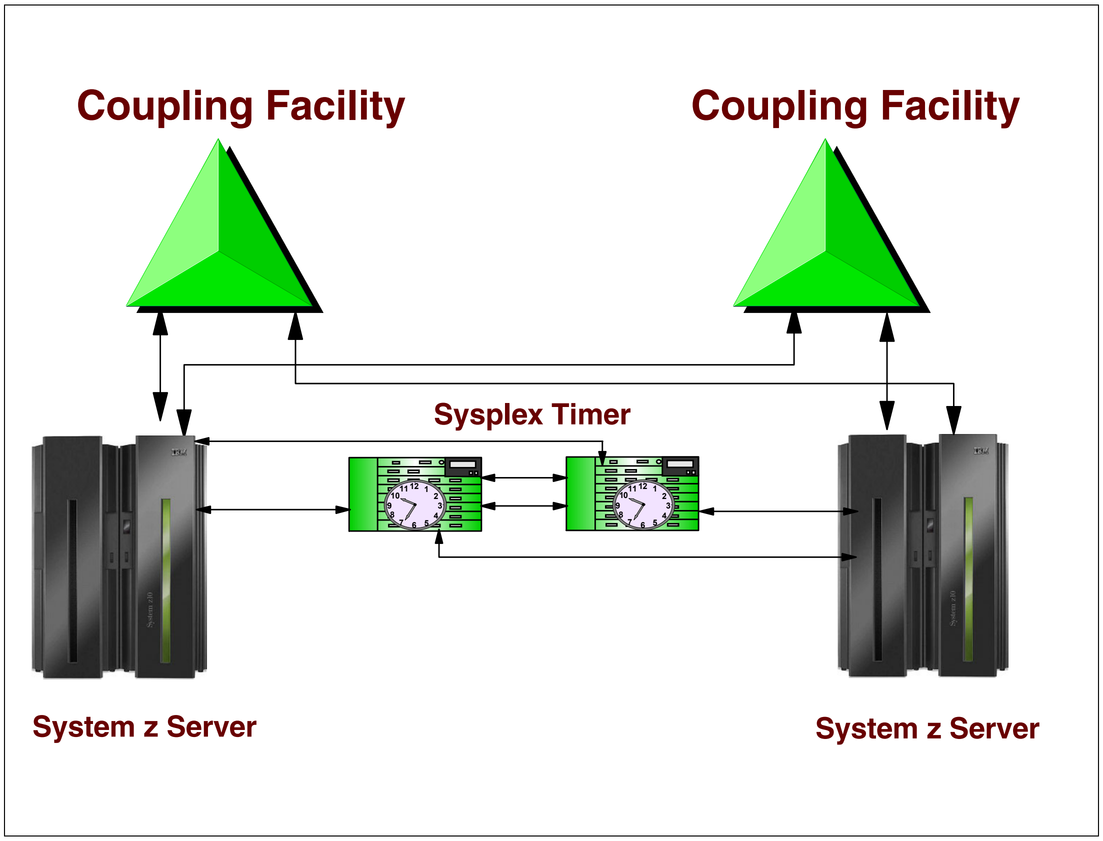


Figure 1-11 Coupling facility

Coupling facility

A coupling facility is a special logical partition that runs the coupling facility control code (CFCC) and provides high-speed caching, list processing, and locking functions in a sysplex. HCD enables you to specify whether a logical partition can be a coupling facility, operating system, or either on certain processors. You connect the coupling facility logical partition to a processor through the coupling facility channels.

With z/OS services, a component called XES allows authorized applications, such as subsystems and z/OS components, to use the coupling facility to cache data, exchange status, and access sysplex lock structures in order to implement high performance data sharing and rapid recovery from failures.

Coupling facility control code (CFCC)

IBM CFCC is licensed internal code (LIC) and *always runs under an LPAR*, regardless of whether the CF is in a standalone CPC or in a general purpose CPC (where CFCC LPs are together with z/OS LPs). A standalone CPC is a CPC only allowing LPs running CFCCs.

CFCC is a simple but efficient operating system where, for example, no virtual storage support is implemented. It has multiprocessing capabilities running multiple processors and when there is no work to do, it loops in the CF link waiting for work requests (the interrupt mechanism is not implemented).

A coupling facility (CF) runs the coupling facility control code (CFCC) that is loaded into main storage at power-on reset (POR) time. CFCC can run on a stand-alone CF server or in a logical partition.

Coupling facility logical partition (LP)

The coupling facility LP is defined through HCD and processor resource/systems manager (PR/SM) panels on the Hardware Management Console (HMC). Once you have defined an LP to be a coupling facility LP, only the CFCC can run in that LP. When you activate the coupling facility LP, the system automatically loads the CFCC from the laptop support element (SE) hard disk of the CPC. Its major functions are:

- Storage management
- Support for CF links
- Console services (HMC)
- Trace, logout, and recovery functions
- Provide support for the list, cache, and lock structures

1.12 Message time ordering

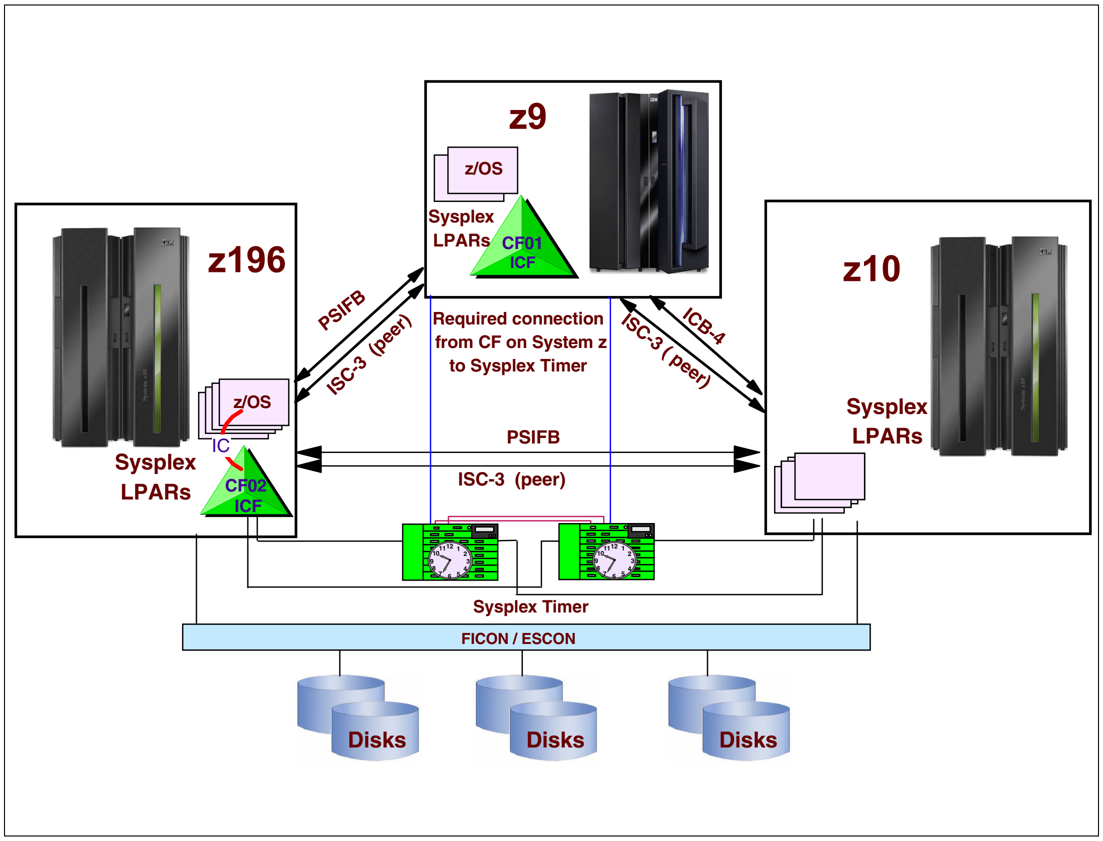


Figure 1-12 Message time ordering

Parallel Sysplex configurations

As server and coupling link technologies have improved over the years, the synchronization tolerance between operating system images in a Parallel Sysplex has become more rigorous. In order to ensure that any exchanges of time stamped information between operating system images in a Parallel Sysplex involving the CF observe the correct time ordering, time stamps are included in the message transfer protocol between the server operating system images and the CF. This is known as *message time ordering*.

Message time ordering

Figure 1-12 illustrates the components of a Parallel Sysplex as implemented within the zSeries architecture. Shown is an ICF connection between two z10 EC servers running in a sysplex, and there is a second integrated coupling facility defined within one of the z10s containing sysplex LPARs. Shown also is the connection required between the coupling facility defined on a z10 and the sysplex timer to support message time ordering. Message time ordering requires a CF connection to the Sysplex Timer.

Before listing the message time ordering facility rules, a short description of the message time ordering facility is included here. When the CF receives the message, it verifies that the message's time stamp is less than the CF's TOD clock. If the time stamp in the message is ahead of the CF's TOD clock, the message is not processed until the CF TOD clock catches up to the message time stamp value.

Mixed or STP-only Parallel Sysplex

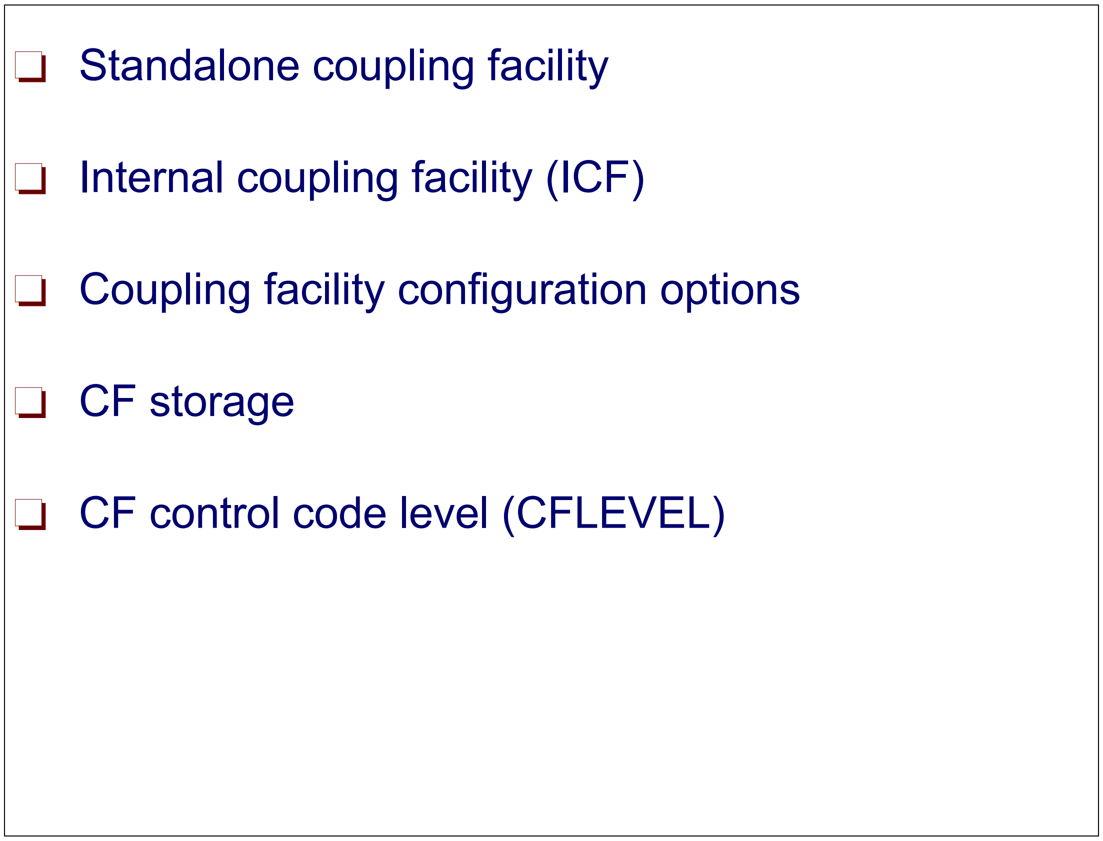
In a mixed or STP-only CTN in a Parallel Sysplex configuration, the requirement is that all servers support the message time ordering facility.

The following message time ordering facility rules are enforced when there is a mixed CTN in a Parallel Sysplex configuration:

- z/OS images running on STP-configured servers can connect to CFs that are on servers that are not STP capable only if the coupling facility supports message time ordering facility and is attached to the Sysplex Timer.
- CFs on an STP-configured server can connect to z/OS images running on servers that are not STP capable only if the non-STP-capable server supports message time ordering facility and is attached to the Sysplex Timer.

For more details about this topic, see the document *Server Time Protocol Planning Guide*, SG24-7280.

1.13 Coupling facility LPARs and CFCC code



- ❑ Standalone coupling facility
- ❑ Internal coupling facility (ICF)
- ❑ Coupling facility configuration options
- ❑ CF storage
- ❑ CF control code level (CFLEVEL)

Figure 1-13 Coupling facility LPARs and CFCC code

Standalone coupling facility (CF)

A standalone CF is a CPC where all the processor units (PUs), links, and memory are for CFCC use. This means that all LPs running in such a CPC run *only* CFCC code. The standalone CF is also called external coupling facility.

Internal coupling facility (ICF)

ICFs are PUs in a CPC configured to run only CFCC code. The PUs are not shipped with a fixed function assignment (personality), but are assigned during power-on reset (POR) or later non-disruptively by *on demand* offerings such as: CBU, CuOD, CIU, ON/OFF COD. Those offerings allow the customer to convert, in seconds, a non-characterizable PU in any PU personality type such as: CPU, ICF, IFL, zAAP, zIIP and SAP.

An ICF can reduce the cost of exploiting coupling facility technology because:

- ICFs are less expensive than CPs.
- An ICF has a special software license charge. Special PR/SM microcode prevents the defined ICF PUs from executing non-CFCC code such as z/OS.

Coupling facility configuration options

A coupling facility *always runs CFCC code within a PR/SM LPAR license internal code (LIC)*. As we already saw, a CF LPAR can be configured in one of two ways:

- In a standalone CPC, only coupling facilities are present

- In a general purpose CPC, the CF LP can co-exist with LPs running z/OS code, or even Linux® (under z/VM® or not). The z/OS LPs can be either in the same Parallel Sysplex as the coupling facility or not.

Decisions regarding where to configure CF LPs are based mostly on price/performance, configuration characteristics, CF link options, and recovery characteristics (availability).

CF storage

CFCC formats central storage in contiguous pieces called *structures*. Structures can be used to keep data by software exploiters (authorized programs) such as: z/OS components, subsystems, products. The exploiters may have several instances running in different z/OS systems in the sysplex. The major reason for having structures is to implement data sharing (with total integrity), although the structures may be used as high-speed caching memory. Structure types are:

- Lock structure
- List structure
- Cache structure

Each structure type provides a specific function to the exploiter. Some storage in the coupling facility can also be allocated as a dedicated dump space for capturing structure information for diagnostic purposes. In order to access the coupling facility structures, z/OS systems (running the exploiters) must have connectivity to the coupling facility through *coupling facility links*. Refer to “Coupling facility links” on page 25 for more information.

Coupling facility control code level (CFLEVEL)

The level (CFLEVEL) of the coupling facility control code (CFCC) that is loaded into the coupling facility LPAR determines what functions are available for exploiting applications. Various levels provide new functions and enhancements that an application might require for its operation. As more functions are added to the CFCC, it might be necessary to allocate additional storage to a coupling facility structure. Similarly, as new functions are added, the coupling facility itself may require additional storage. In any configuration, the amount of fixed storage required for the coupling facility is based on configuration-dependent factors.

To implement a coupling facility in your sysplex requires both hardware and software, as follows:

- CPC that supports the CFCC.
- CPCs on which one or more z/OS images run and which are capable of connecting to the coupling facility with CF links.
- Appropriate level of z/OS that allows an exploiter to access a desired function when managing the coupling facility resources.
- CFCC must implement the functions the exploiter needs.

To support migration from one CFCC level to the next, you can run several levels of CFCC concurrently as long as the coupling facility logical partitions are running on different servers (CF logical partitions running on the same server share the same CFCC level).

1.14 Coupling facility links

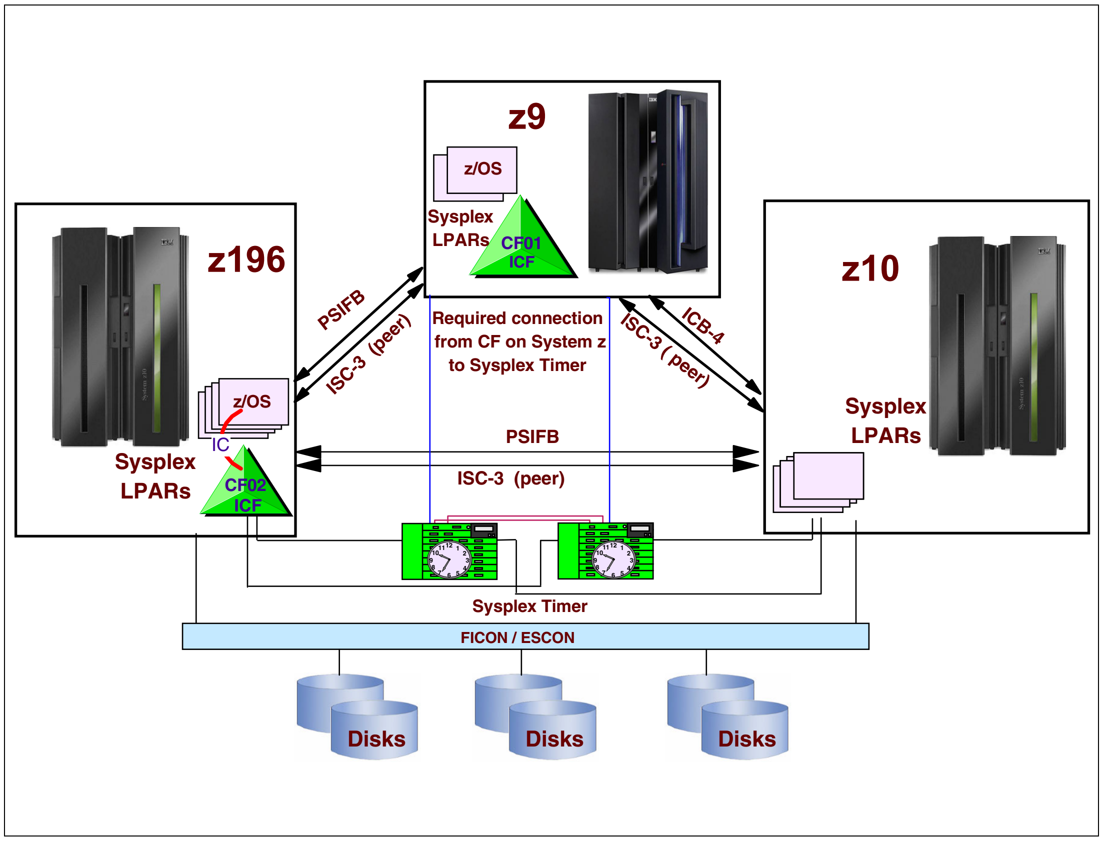


Figure 1-14 Coupling facility links

Coupling facility links

To enable the communication between a coupling facility (CF) logical partition (LP) and the z/OS (LPs), special types of high-speed CF links are required. These links are important because of the impact of link performance on CF request response times. For configurations covering large distances, time spent on the link can be the largest part of CF response time.

The coupling link types for the coupling facility are:

```
(PSIFB, IC, ICB, and ISC-3)
```

A CF link adapter can be shared between LPs, meaning the same adapter can transfer data from/to different z/OS systems to one CF, thus reducing the number of links needed. This is called multiple image facility (MIF), the same name used for FICON and ESON channels.

CF links in the System z servers work in a mode called *peer mode*. In this mode we have even more flexibility with connections. For example, a single link adapter can be connected (multiple image facility) to both z/OS and a CF.

Both the coupling facility LPs and the CF links must be defined to the I/O configuration data set (IOCDS). Hardware configuration definition (HCD) provides the interface to accomplish these definitions and also automatically supplies the required channel control unit and I/O device definitions for the coupling facility channels.

Internal coupling (IC) link

The IC is a zSeries or z9 connectivity option that enables high-speed connection (more than 3 GB/sec) between a CF LP and one or more z/OS LPs running on the same zSeries or z9 CPC. The IC is a *linkless connection* (implemented in Licensed Internal Code) and so does not require any hardware or cabling.

InterSystem Channel (ISC) link

InterSystem Channel provides connectivity through optical cables. ISC links are point-to-point connections that require a unique channel definition at each end of the link. There are two types of ISC features supported, ISC-3 and ISC-2. In modern CPCs such as System z servers (z800, z900, z890, z990 and z9), there is support for ISC-3 only. They are used for distances beyond 7 meters and allow a rate of up to 200 MBps.

Integrated Cluster Bus (ICB)

Integrated Cluster Bus provides connectivity through copper cables. They are faster than ISC links, attaching directly to a Self-Timed Interconnect (STI) bus of the CEC cage. They are the preferred method for coupling connectivity when connecting System z servers over short distances (up to 7 meters). For longer distances, ISC-3 links must be used. There are two types of ICB links available:

- *ICB-4* links provide 2 GB/sec coupling communication between z990, z890, and z9 CPCs.
- *ICB-3* links provide 1 GB/sec coupling communication between z800, z900, z990, z890, and z9 CPCs.

InfiniBand coupling links (PSIFB)

InfiniBand coupling links (PSIFB) are high speed links on z196, z10, and z9 servers. The PSIFB coupling links originate from three types of fanout. PSIFB coupling links of either type are defined as CHPID type CIB in HCD/IOCP, as follows:

- HCA2-O (FC 0163)

  The HCA2-O fanouts support InfiniBand Double Data Rate (IB-DDR) and InfiniBand Single Data Rate (IB-SDR) optical links. The HCA2-O fanout supports PSIFB coupling links at distances of up to 150 meters. PSIFB coupling links operate at 6 GBps (12x IB-DDR) when connecting a z196 or z10 to z196 and z10 servers, and at 3 GBps (12x IB-SDR) when connecting a z196 or z10 to a z9. The link speed is auto-negotiated to the highest common rate.

- HCA2-O LR (FC 0168)

  The HCA2-O LR fanout supports PSIFB Long Reach (PSIFB LR) coupling links for distances of up to 10 km and up to 100 km when repeated through a DWDM. This fanout is supported on z196 and z10. PSIFB LR coupling links operate at up to 5.0 Gbps (1x IB-DDR) between z196 and z10 servers, or automatically scale down to 2.5 Gbps (1x IB-SDR) depending on the capability of the attached equipment.

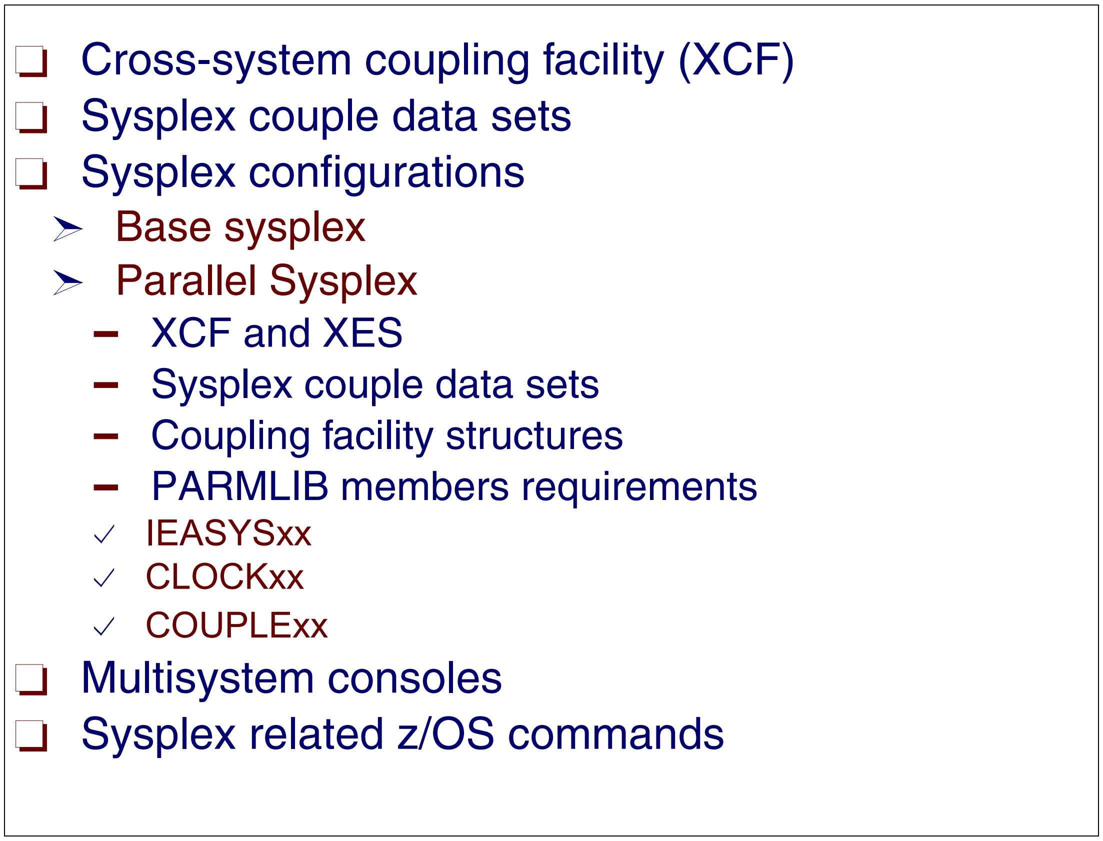
- HCA1-O (FC 0167)

  The HCA1-O fanout supports InfiniBand Single Data Rate (IB-SDR) optical links. The HCA1-O fanout supports PSIFB coupling links at distances of up to 150 meters. PSIFB coupling links operate at 3 GBps (12x IB-SDR) when connecting the z9 server to a z196 or z10 server.

Publications of interest

zSeries Connectivity Handbook, SG24-5444, *Processor Resource/Systems Manager Planning Guide,* SB10-7036, *z/OS MVS Setting Up a Sysplex,* SA22-7625, and *HCD Planning,* GA22-7525.

1.15 Sysplex overview



- ❑ Cross-system coupling facility (XCF)
- ❑ Sysplex couple data sets
- ❑ Sysplex configurations
  - ➢ Base sysplex
  - ➢ Parallel Sysplex
    - – XCF and XES
    - – Sysplex couple data sets
    - – Coupling facility structures
    - – PARMLIB members requirements
    - ✓ IEASYSxx
    - ✓ CLOCKxx
    - ✓ COUPLExx
- ❑ Multisystem consoles
- ❑ Sysplex related z/OS commands

Figure 1-15 Sysplex overview

Sysplex overview

Now that the pieces that make up a sysplex have been introduced, the remainder of this chapter presents an overview of the sysplex. Figure 1-15 is an overview of the remaining topics to be discussed in this chapter.

It is difficult to explain the sysplex without first explaining the *cross-system coupling facility* (XCF) and its services, so we start by describing XCF, its services and exploiters. Take note that despite having in the name the expression "coupling facility," XCF is not able to access such coupling facility directly.

Next, the sysplex configurations and how system consoles are used to enter sysplex-related commands are described, as follows:

- Fundamentals of a base sysplex (a function of XCF), what is required and how to define it.
- An overview of Parallel Sysplex and how to migrate from a base to a Parallel Sysplex.
- Which PARMLIB members you have to change in order to define a sysplex and which changes are necessary.
- How to use consoles in a sysplex environment; how many consoles you need and which ones are mandatory.
- The sysplex-related commands, how to direct a command to a specific console, and how to reply to messages.

1.16 Cross-system coupling facility (XCF)

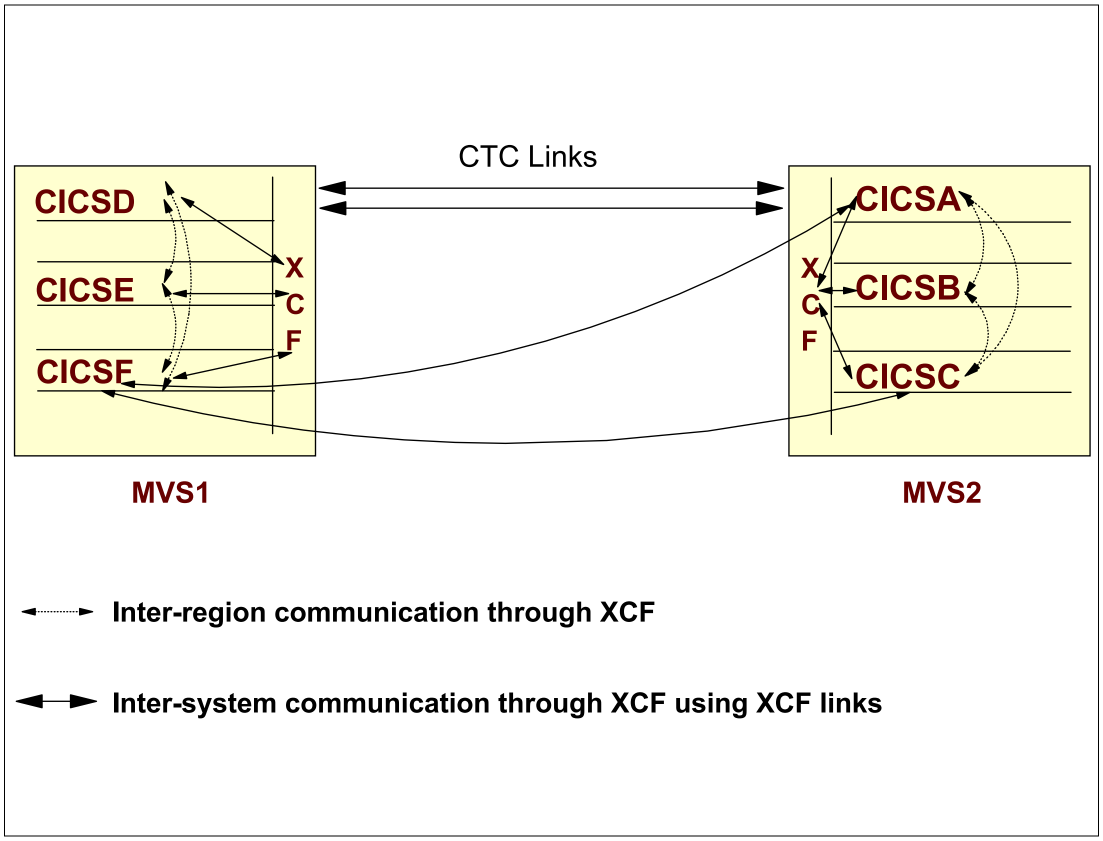

CTC Links
CICSD
CICSE
CICSF
XCF
CICSA
CICSB
CICSC
XCF
MVS1
MVS2
Inter-region communication through XCF
Inter-system communication through XCF using XCF links

Figure 1-16 Cross-system coupling facility (XCF)

Cross-system coupling facility (XCF)

The cross system coupling facility (XCF) component of z/OS provides simplified multisystem management. XCF services allow authorized programs on one system to communicate with programs on the same system or on other systems. If a system fails, XCF services also provide the capability for batch jobs and started tasks to be restarted on another eligible system in the sysplex.

XCF groups

An XCF group is a set of related members that a multisystem application defines to XCF. A member is a specific function, or instance, of the application. A member resides on one system and can communicate with other members of the same group across the sysplex.

Communication between group members on different systems occurs over the signaling paths that connect the systems; on the same system, communication between group members occurs through local signaling services. To prevent multisystem applications from interfering with one another, each XCF group name in the sysplex must be unique.

CICS address spaces

Each CICS address space in Figure 1-16 is an XCF instance. The terms *instance, exploiting instance,* or *exploiter* are used to denote the active subsystem or address space of a component, product, or function that directly exploits sysplex services. XCF is a z/OS component that provides services to allow multiple instances (named members) of an application or subsystem, running on different systems in a sysplex, to share status information and communicate through messages with each other. A set of same instance members is called a *group*. A member of an application can use XCF services to:

- Inform other members of their status (active, failed, and so forth).
- Obtain information about the status of other members, such as whether another member failed.
- Send messages to and receive messages from each other member, in the same or in another z/OS. The most frequent case is other z/OSs (inter-system communication).

Automatic Restart Manager (ARM)

If z/OS fails, XCF can call Automatic Restart Manager (ARM) services (a z/OS component) to provide the capability for batch jobs and started tasks to be restarted on another eligible z/OS in the sysplex. Then an application is allowed to:

- Request automatic restart in the event of application or system failure
- Wait for another job to restart before restarting
- Indicate its readiness to accept work
- Request that automatic restart no longer be performed
- Indicate that automatic restart should be performed only if the backup copy of the application data no longer exists

1.17 Base sysplex

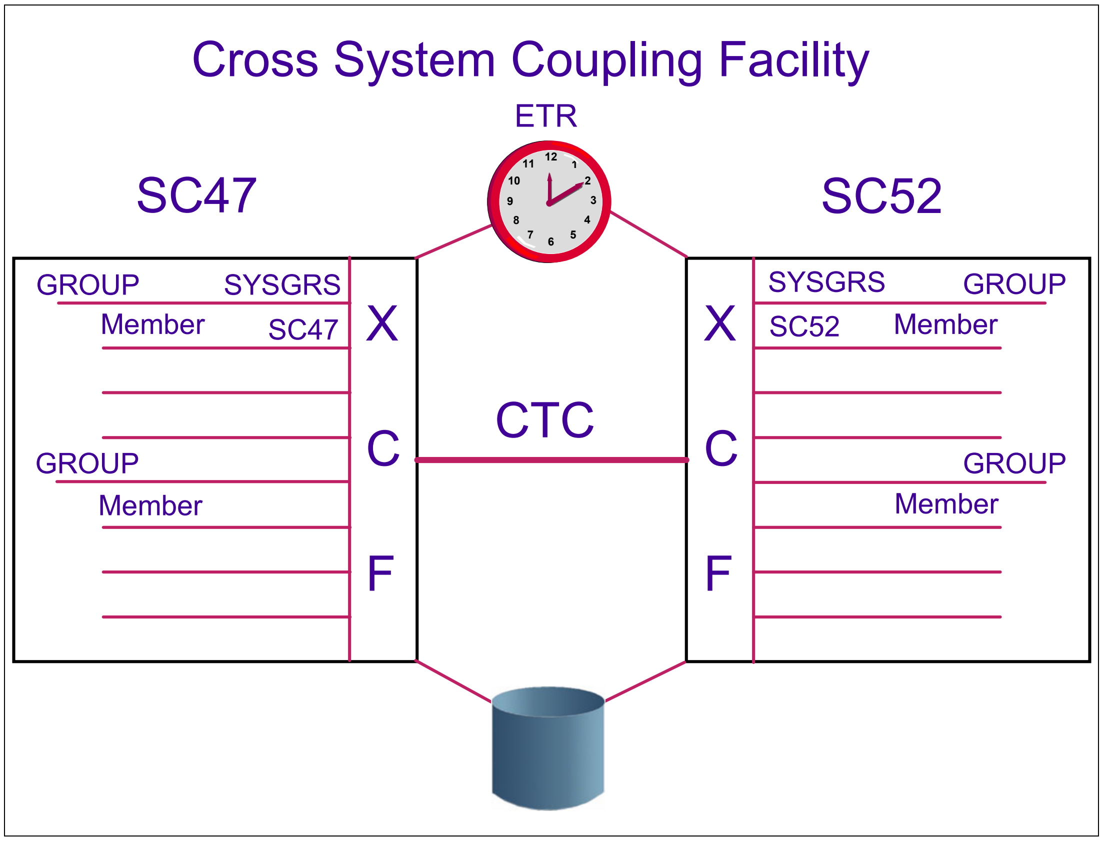


Figure 1-17 Base sysplex

Base sysplex

A base sysplex configuration is a sysplex with no coupling facilities. The base sysplex can be composed of one or more z/OS systems that have an XCF sysplex name and in which the authorized programs (members) use XCF services. XCF services are available in both single and multisystem environments. A multisystem environment is defined as two or more z/OS systems residing on one or more CPCs' logical partitions connected through CTCs.

A base sysplex is the first step to implementing a Parallel Sysplex. A Parallel Sysplex is a base sysplex plus the use of the coupling facility. So, when you introduce the coupling facility, XCF exploits the coupling facility, using it as a link between z/OS systems.

Figure 1-17 shows how XCF works in a multisystem sysplex. Each z/OS has an XCF component that handles groups for the participating members. Here you can see as an example the group named SYSGRS, which is very important for global resource serialization (in the group the name of each member is the name of the z/OS system). This group has one member (GRS component of z/OS) in each z/OS. Other groups can be: consoles (SYSMCS), JES2, JES3, WLM, and others. These groups are described in detail in 1.23, "XCF exploiters" on page 39.

The communication link between XCFs could be through channel-to-channel adapters (CTCs) that allow data movement between XCF buffers in the systems through an I/O operation. Another option for linking is a CF list structure, which is discussed later in this chapter.

1.18 XCF application, member, and group

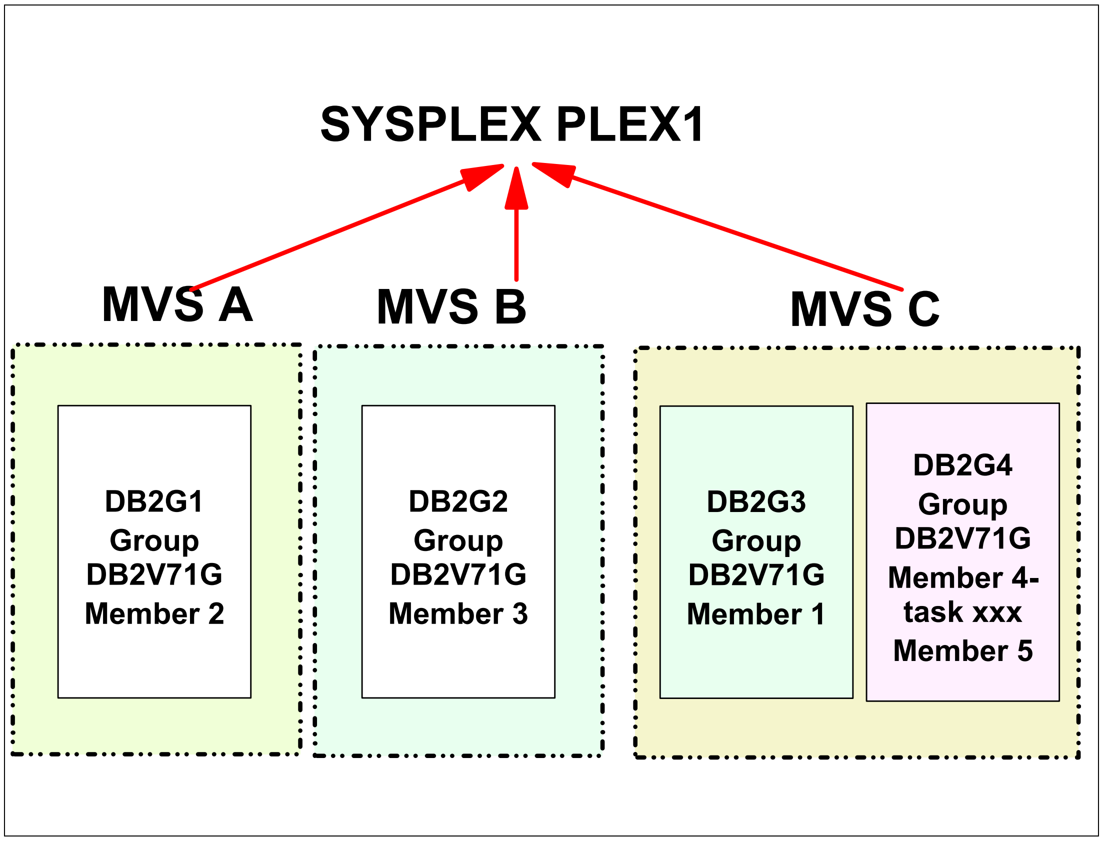

SYSPLEX PLEX1
MVS A
MVS B
MVS C
DB2G1 Group DB2V71G Member 2
DB2G2 Group DB2V71G Member 3
DB2G3 Group DB2V71G Member 1
DB2G4 Group DB2V71G Member 4- task xxx Member 5

Figure 1-18 XCF application, member, and group

XCF application, member, and group

An application in a XCF context is a program that has various functions distributed across z/OS systems in a multisystem environment. An application or subsystem might consist of multiple instances, each running on a different system in the same sysplex. Typically, each instance performs certain functions for the application as a whole. Alternatively, each instance could perform all the application's functions on a given system. XCF services is available to authorized applications, such as subsystems and z/OS components, to use sysplex services.

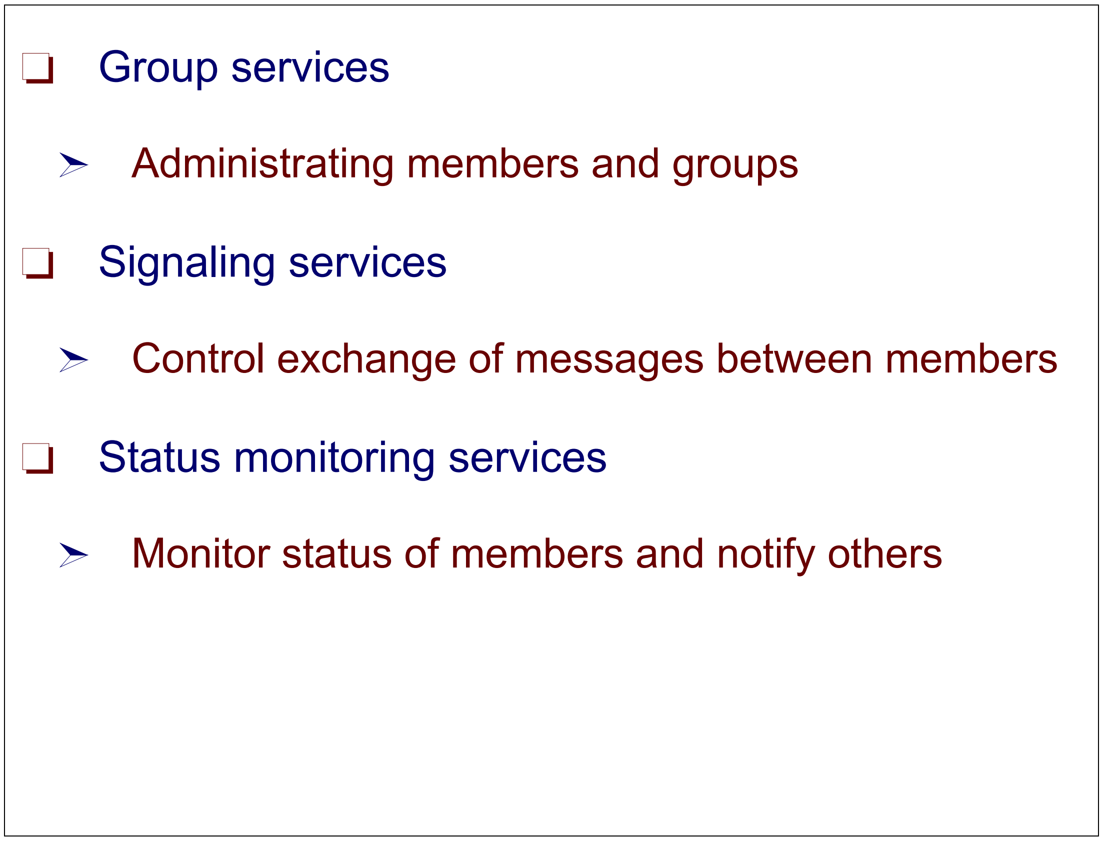
Member

A member is a specific function (one or more routines) of a multisystem application that is joined to XCF and assigned to a group by the multisystem application. A member concept applies to all authorized routines running in the address space that issued the IXCJOIN macro service. Only for termination purposes (resource clean-up), the member can be associated with an address space, job step, or task. XCF terminates the member when its association ends. The same address space can have more than one group.

Group

A group is the set of related members defined to XCF by a multisystem application in which members of the group can communicate with other members of the same group. A group can span one or more of the systems in a sysplex and represents a complete logical entity to XCF.

1.19 XCF services



- Group services
  - Administrating members and groups
- Signaling services
  - Control exchange of messages between members
- Status monitoring services
  - Monitor status of members and notify others

Figure 1-19 XCF services

XCF services

z/OS XCF allows up to 32 z/OS systems to communicate in a sysplex. XCF provides the services that allow multisystem application functions (programs) on one z/OS system to communicate (send and receive data) with functions on the same or other z/OS systems. The communication services are provided through authorized assembler macros and are as follows:

- Group services
- Signalling services
- Status monitoring services

Group services

XCF group services provide ways for defining members to XCF, establishing them as part of a group, and allowing them to find out about the other members in the group. A member introduces itself to XCF through the IXCJOIN macro. If a member identifies a group exit routine, XCF uses this routine to notify this member about status changes that occur to other members of the group, or systems in the sysplex; thus, members can have the most current information about the other members in their group without having to query each other.

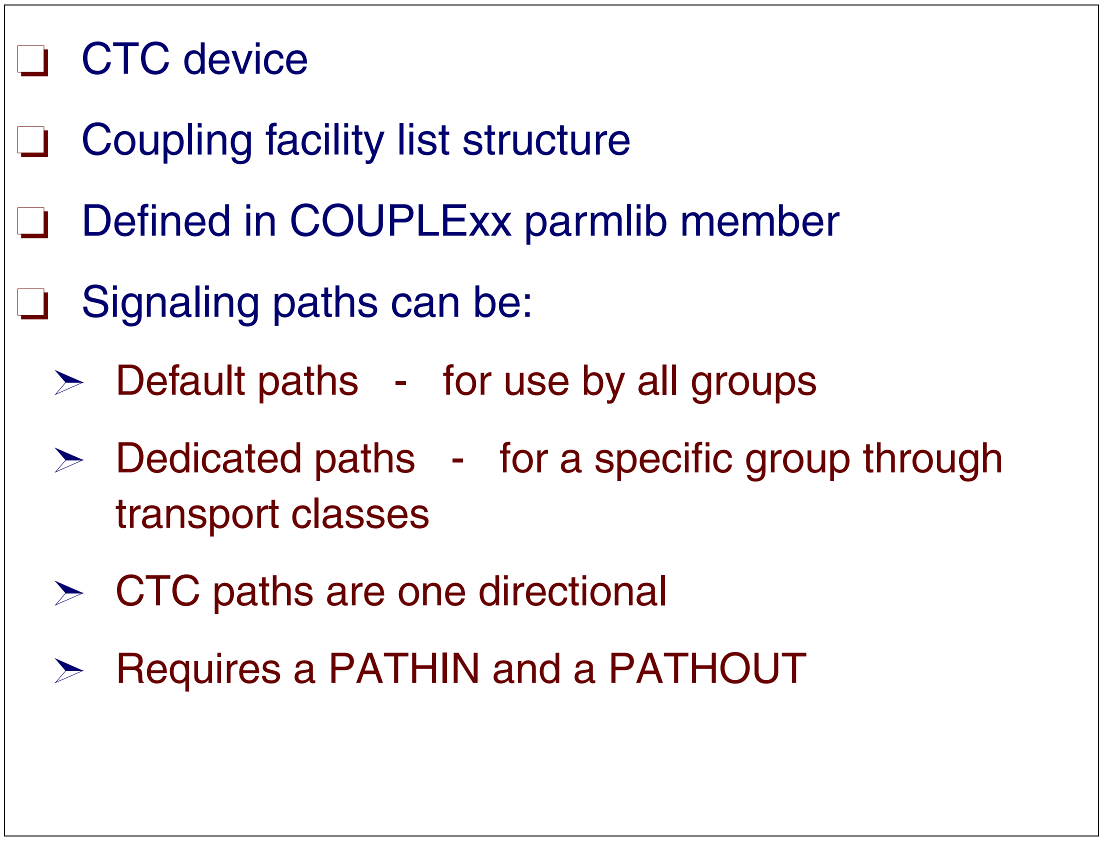
Signaling services

The signaling services control the exchange of messages between members of an XCF group. The sender of a message requests services from XCF signaling services. XCF uses buffer pools to communicate between members in the same system, and it uses buffer pools plus signaling paths (CTCs or a CF list structure) to send messages between systems in the sysplex.

Status monitoring services

Status monitoring services provide a way for members to determine their own operational status and to notify the other members of the group when that operational status changes. An installation-written status exit routine identified to XCF determines whether a member is operating normally. An installation-written group exit routine identified to XCF allows a member to maintain current information about other members in the group, and systems in the sysplex.

1.20 XCF signaling paths



- CTC device
- Coupling facility list structure
- Defined in COUPLExx parmlib member
- Signaling paths can be:
  - Default paths - for use by all groups
  - Dedicated paths - for a specific group through transport classes
  - CTC paths are one directional
  - Requires a PATHIN and a PATHOUT

Figure 1-20 XCF signaling paths

XCF signaling paths

Whether you implement signaling through CTC connections or coupling facility list structures, you need to specify information in the COUPLExx parmlib member. After an IPL, you can issue the **`SETXCF START,PATHOUT`** or **`SETXCF START,PATHIN`** commands to specify outbound or inbound paths. To establish signaling through a coupling facility, you also need to define a coupling facility resource management (CFRM) policy.

Then, XCF group members use the signaling mechanism to communicate with each other. These communication paths can be:

- Channel-to-channel adapter (CTC) communication connections.
- Coupling facility through list structures. XCF calls XES services when the path of communication is a coupling facility.
- A combination of both, CTC and list structures. In this case XCF selects the faster path.

A message generated in a member has the following path:

- From the sending member to an XCF buffer in the output buffer pool.
- From the XCF output buffer through the signaling path:
  - If a CTC goes directly to a buffer in the XCF input buffer pool.
  - If CF list structure is stored for a while in the structure memory and from it to an XCF buffer in the input buffer pool. It is like a mail box.

- From the XCF input buffer to a receiving member.

The communication path is determined at IPL by the PATHIN and PATHOUT definitions in the COUPLExx PARMLIB member and can be modified by the SETXCF operator command. This is explained later in this chapter

CTC paths and devices

To establish signaling paths through CTC devices, you specify their device numbers in HCD. Each CTCA has a logical controller with a certain amount of logical devices. CTC devices are used by XCF in a uni-directional mode. That is, on each system, messages sent to other systems require an outbound path, and messages received from other systems require a separate inbound path. Then, for each z/OS in the sysplex, you must specify all devices for outbound paths and inbound paths on the DEVICE keyword of the PATHOUT and PATHIN statements in COUPLExx.

Coupling facility list structures

When you define signaling paths through coupling facility list structures, you define to z/OS which list structures to use and how to use them (as PATHIN or PATHOUT), and XCF (through z/OS) creates the logical connections between z/OSs that are using the structures for signaling.

Signaling paths and transport classes

Signaling path is the set of outbound paths plus outbound message buffers in one XCF. A transport class is a set of signalling paths that can be associated with XCF groups. Transport classes are defined in the COUPLExx member in PARMLIB. The reasons for associating an XCF group with a transport class include:

- To isolate the messages (signalling path wise) of these groups from the others

  An example from COUPLExx is:

  ```
  CLASSDEF CLASS(CICS) GROUP(DFHIR0000)
  PATHOUT STRNAME(IXC_CICS) CLASS(CICS)
  ```

  In this example, the group DFHIR0000 is associated with a transport class named CICS and all the messages out from the group use the CF list structure named IXC_CICS.

- To optimize the size of the message with the size of the output buffer

1.21 XCF channel-to-channel connection (CTC)

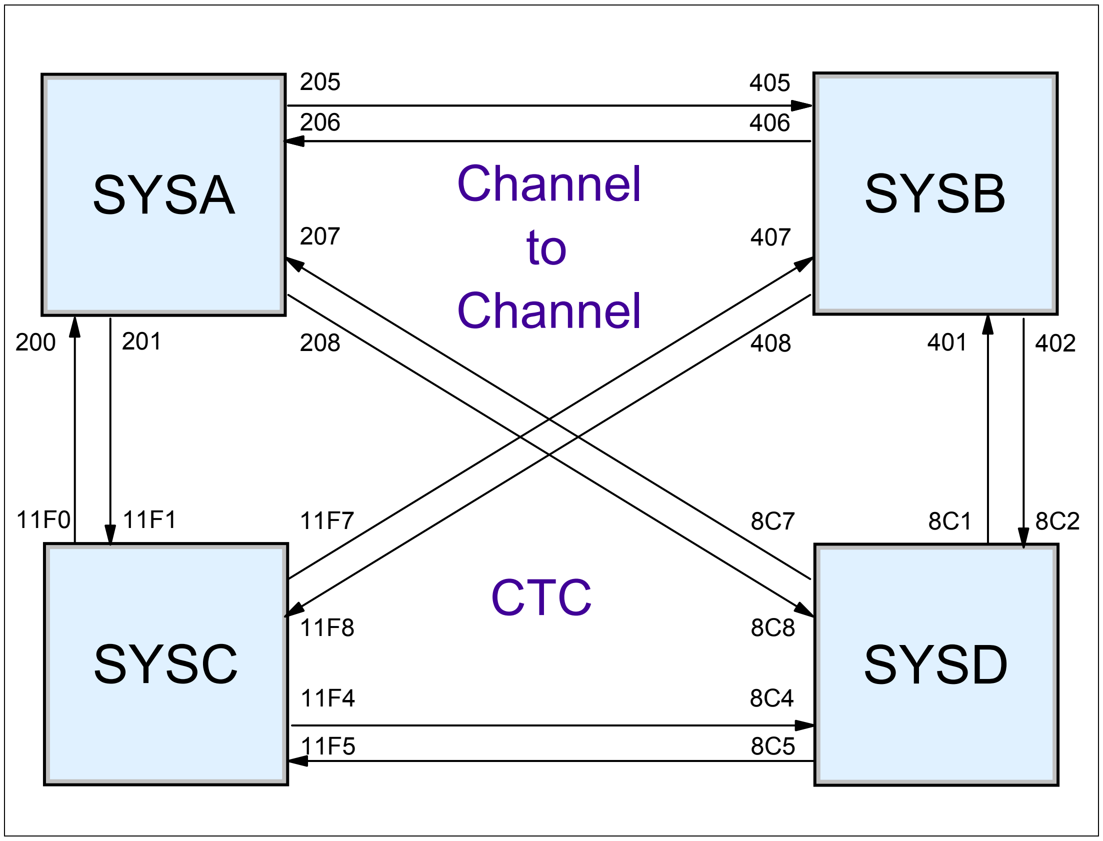


Figure 1-21 XCF channel-to-channel connection (CTC)

XCF channel-to-channel adapter connection (CTCA)

Full signaling connectivity is required between all z/OSs in a sysplex; that is, there must be an outbound and an inbound path between each pair of systems in the sysplex.

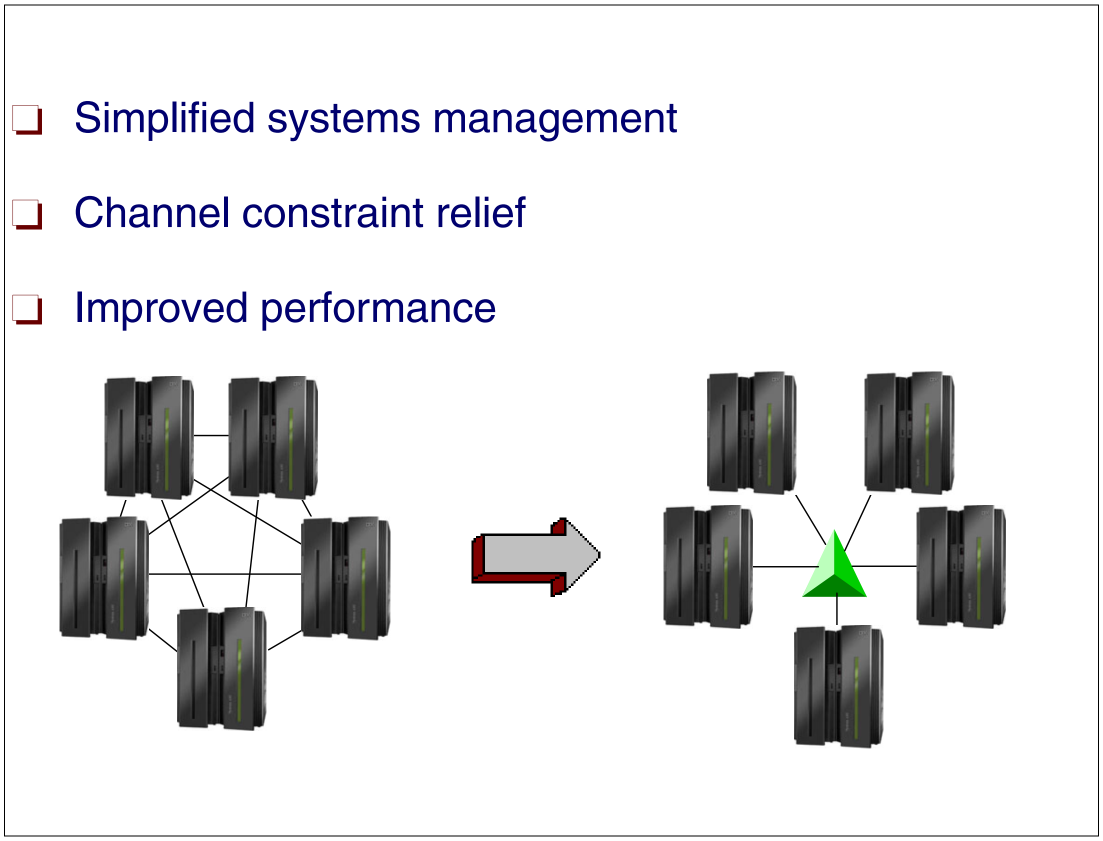
To avoid a single point of signaling connectivity failure, use redundant connections between each pair of systems, through either CTC connections or coupling facility list structures.

CTC signaling paths

CTC signaling paths are uni-directional and require at least four signaling paths between each z/OS (two inbound and two outbound paths). CTCA connections and their device numbers become more complex to manage as the number of z/OSs in the configuration increases. For example, it is almost impossible to have the same COUPLExx member for all the z/OSs in the sysplex. The formula producing the number of connections is: N x (N-1) where N is the number of z/OSs.

Coupling facility signaling paths are bi-directional; this offers better performance in general and is much less complex to maintain. We discuss coupling facility structures next.

1.22 XCF using coupling facility list structures



- ❑ Simplified systems management
- ❑ Channel constraint relief
- ❑ Improved performance

Figure 1-22 XCF using coupling facility list structures

List structures

A list structure consists of a set of lists and an optional lock table of exclusive locks, which you can use to serialize the use of lists, list entries, or other resources in the list structure. Each list is pointed to by a header and can contain a number of list entries. A list entry consists of list entry controls and can also include a data entry, an adjunct area, or both. Both data entries and adjunct areas are optional. However, data entries are optional for each list entry while a list structure either has or doesn't have adjunct areas.

Simplified systems management

When XCF uses coupling facility structures for signaling, each other XCF in the sysplex can automatically discover its connectivity to other systems via the CF, without having to define point-to-point connections on every XCF in the configuration. Furthermore, in the event that a signaling path or signaling structure is lost, the recovery of CF signaling structures is automated and greatly simplified if more than one structure is allocated.

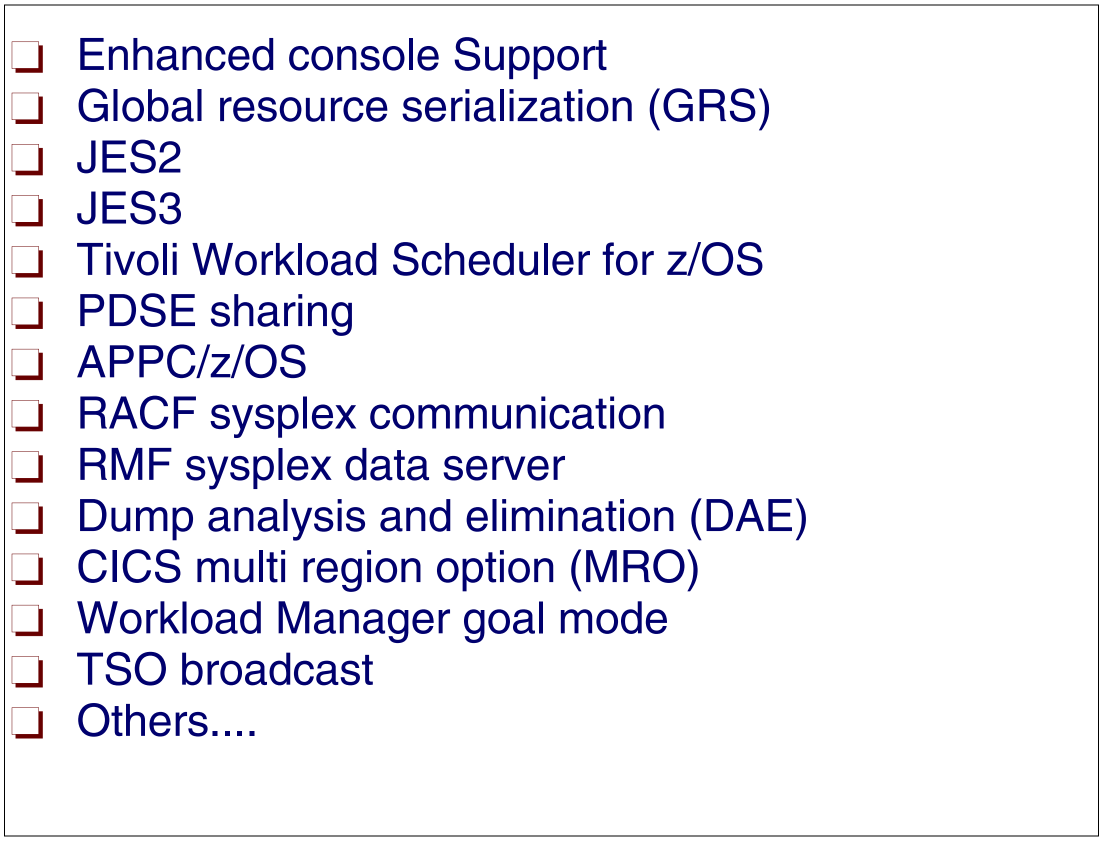
Channel constraint relief

XCF allows customers to consolidate CTC links used by VTAM, JES, and GRS into XCF communication, freeing up channel paths for other uses in constrained configurations. These advancements in CF coupling technologies combined with simplified systems management, ease of recovery, and better cost efficiencies, make XCF the clear choice for configuring XCF signaling paths in a Parallel Sysplex cluster.

List services (IXLLIST)

IXLLIST provides high-performance list transition monitoring that allows you to detect when a list changes from the empty state to the non-empty state (in which it has one or more entries) without having to access the coupling facility to check the list. For instance, if you are using the list structure as a distribution mechanism for work requests, list transition monitoring allows users to detect easily the presence or absence of incoming work requests on their queues.

1.23 XCF exploiters



- Enhanced console Support
- Global resource serialization (GRS)
- JES2
- JES3
- Tivoli Workload Scheduler for z/OS
- PDSE sharing
- APPC/z/OS
- RACF sysplex communication
- RMF sysplex data server
- Dump analysis and elimination (DAE)
- CICS multi region option (MRO)
- Workload Manager goal mode
- TSO broadcast
- Others....

Figure 1-23 XCF exploiters

XCF exploiters

Figure 1-23 shows some sample z/OS components and products that exploit XCF services.

Enhanced console support

Multisystem console support allows consolidation of consoles across multiple z/OS images. A console address space of one z/OS image can communicate with console address spaces of all other images in the sysplex by means of XCF signaling. With this support, any console in the sysplex has the capability to view messages from any other system in the sysplex and to route commands in the same way.

Global resource serialization (GRS)

With the introduction of GRS Star, a new method of communicating GRS allocation requests was introduced. GRS Star uses a lock structure in the coupling facility as the hub of the GRS complex and eliminates the delays and processing overhead inherent in the traditional ring configuration. However, all quiescing and purging of images from GRS is done automatically by XCF. Also the execution of the GRSCAN function is done through XCF communication. This function is requested by performance monitors such as RMF in order to discover contention situations to be reported. To detect global contention the GRSs talk among themselves through XCF links.

JES2

When in multi-access spool (MAS) configuration (several JES2 member sharing the same spool data set), JES2 uses XCF to communicate with other members. Previously, all communication between members was via the JES2 checkpoint data set and, in the event of a system failure by one member of a JES2 MAS, the operator had to manually issue a reset command to requeue jobs that were in execution on the failing image and make them available for execution on the remaining images in the MAS complex. If running in a sysplex, this can be done automatically because of the XCF communication capability.

JES3

JES3 uses XCF services to communicate between JES3 systems in a complex. Previously, JES3 had to manage its own CTCs for this communication. This enables you to reduce the overall number of CTCs that need to be managed in your installation.

Tivoli Workload Scheduler for z/OS

IBM Tivoli Workload Scheduler for z/OS (TWS) is a subsystem that can automate, plan, and control the processing of a batch production workload. In a sysplex environment, TWS subsystems in separate z/OS images can take advantage of XCF services for communication with each other.

PDSE sharing

In partitioned data sets extended (PDSEs) access, XCF services is used for the exchange of locking information between the sharing systems in a sysplex (extended sharing only). Multiple images in a sysplex can concurrently access PDSE members for input and output, but not for update-in-place (the protection is implemented through XCF communication). That is, any member of a PDSE data set (pay attention that a PDSE member is not an XCF member), while being updated-in-place, can only be accessed by a single user. A sharer of a PDSE can read members, create new members or new copies of existing members concurrently with other sharers on the same or other images, input/output, but not for update-in-place.

APPC/z/OS

APPC/z/OS uses XCF to communicate with transaction-scheduler address spaces on the same image that APPC/z/OS is running on.

RACF sysplex communication

RACF can be enabled for sysplex communication between RACF members in the same sysplex. Doing this enables the subsequent commands to be propagated to RACF members in the sysplex other than the member who issued the command, thus simplifying multisystem security management.

RMF sysplex data server

The RMF data server is an in-storage area that RMF uses to store SMF data, which can then be moved around the sysplex to provide a sysplex-wide view of performance, via RMF Monitor III, without having to sign on to each of the systems individually.

Dump analysis and elimination (DAE)

Sharing the SYS1.DAE data set between images in a sysplex can avoid taking multiple dumps for the same problem if it is encountered on different systems in the sysplex.

CICS multi region option (MRO)

With CICS running in a sysplex, MRO has been extended to include cross-system MRO. This is achieved using XCF services for the cross-system communication, rather than having to use VTAM ISC links, as was previously required. This can give considerable performance benefits over current ISC implementations. MRO allows splitting of all the CICS functions that before were jammed in just one address space into several specialist address spaces, such as: terminal owning region or TOR (interface with VTAM to receive the incoming transaction), application owning region or AOR (where the programs containing the transaction logic are executed), file owning region (FOR) where VSAM data sets are accessed.

Workload manager in goal mode

WLM uses XCF for communication between WLM address spaces in the same sysplex. This communication is needed because the goals are global and not z/OS local.

TSO broadcast

If all z/OSs in the sysplex share the same SYS1.BRODCAST data set (used to send welcome messages to logged on users), and SYSPLXSHR(ON) is declared in IKJTSOxx PARMLIB member, then TSO NOTICEs are communicated via XCF signaling. Consequently, the I/O to the SYS1.BRODCAST data set is eliminated.

There are more XCF exploiters, such as DB2, IMS, DFSMS, VSAM, VTAM, and others.

1.24 Sympathy sickness

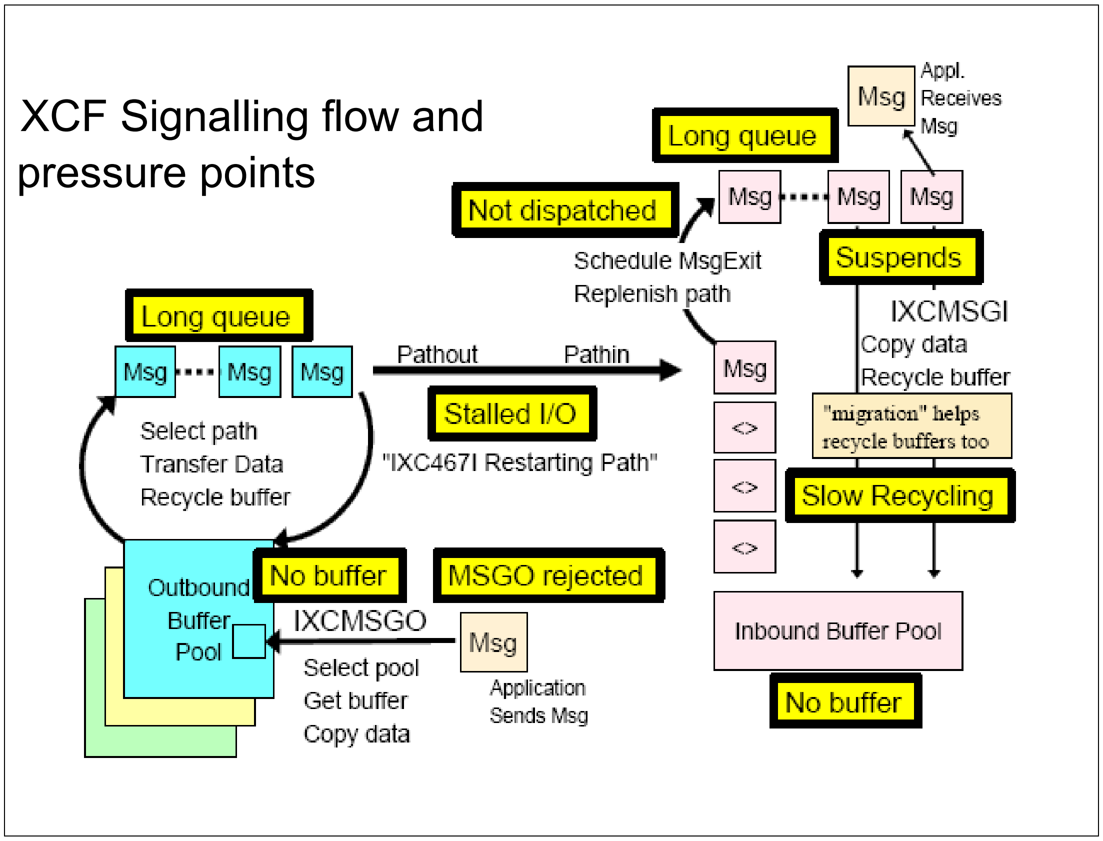


Figure 1-24 XCF Sympathy sickness

XCF sympathy sickness

There is one structural problem when messages from different XCF groups flow through common resources (message buffer pools and paths). This problem is not only with XCF, but is common with all message traffic.

The problem description is: If an input member fails (or slows down) in taking the messages out from the input buffer pool, this can hurt “innocent bystanders” in this and in other z/OS systems.

This can have the related effect of hurting users in other systems. It is known as “sympathy sickness.”

In the very busy Figure 1-24, if the application receiving messages (top right corner) is not taking its messages from the input buffer pool, this pool will be jammed, causing congestion in the normal flow. After a while, the application sending a message in another z/OS (bottom left corner) will receive back “MSG Rejected” because the output buffer pool is full. Depending on its logic this application may abend.

The objective of a new function, in z/OS 1.8 base and 1.6 with APAR OA09194, is to improve availability for innocent bystanders by automatically removing the stalled task that owns the messages that are filling the most PATHIN buffers. This function must be enabled through the SFM policy containing new MEMSTALLTIME parameter.

1.25 Sysplex couple data sets

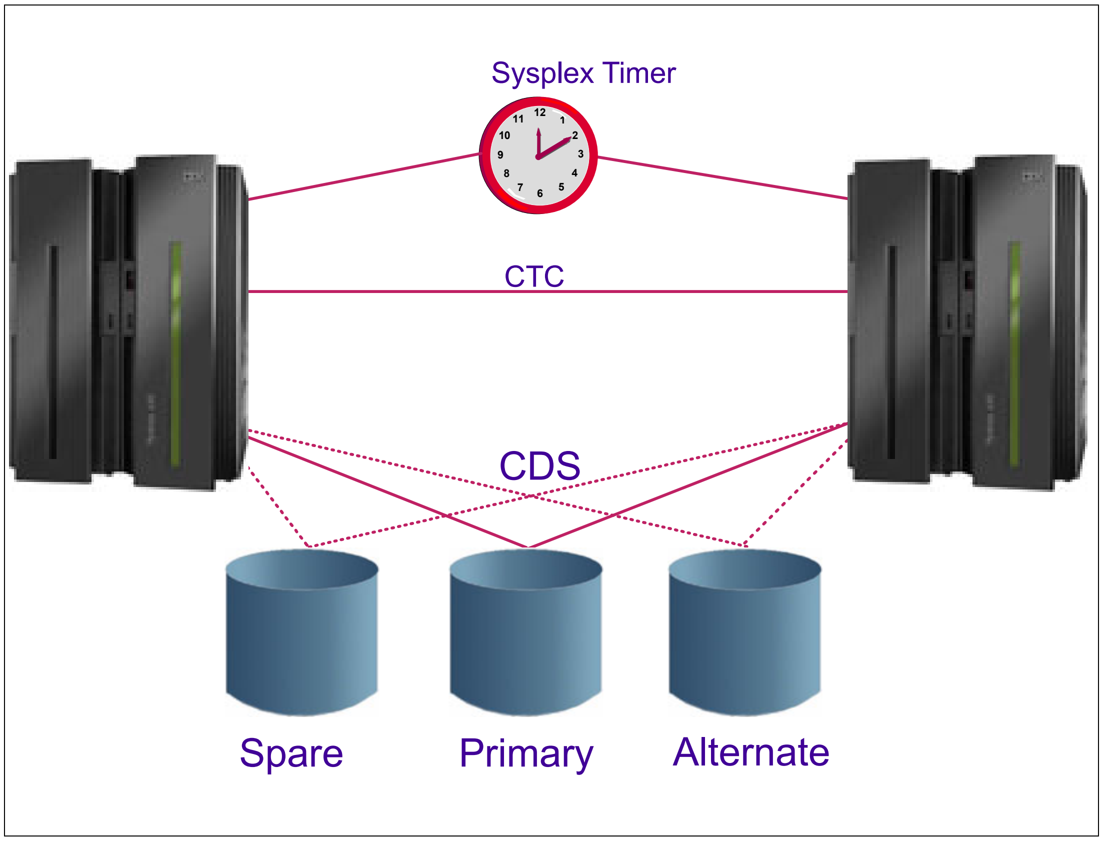


Figure 1-25 Sysplex couple data sets

Couple data sets

Figure 1.25 shows a sysplex of two z/OS systems, where both systems are connected to a Sysplex Timer and can access the sysplex couple data sets. The sysplex might also be configured so the systems can access one or more of the following couple data sets: ARM (automatic restart management), LOGR, SFM, WLM, z/OS UNIX, and a sysplex couple data set.

Sysplex couple data set

XCF requires a sysplex couple data set to be shared by all XCFs in the sysplex. The sysplex couple data set resides on a DASD device and is used to:

- Hold general status information about the z/OS images, XCF groups, and the XCF group members running in the sysplex
- Hold a description of pathouts and pathins together with buffer pools and transport classes
- Point to other couple data sets used in the sysplex
- Contain the heart beat data which inspects all systems to verify if they are alive and working

The couple data sets in a sense are used as a PARMLIB, that is, they have parameters set by the installation to customize the system. However, couple data sets are designed to be shared among z/OS systems in a sysplex (and are not actually a true PARMLIB). When a system programmer is creating their contents, care should be taken with the syntax of the statements.

All coupled data sets need to be created, formatted, and finally filled with installation options.

Primary and alternate data sets

To avoid a single point of failure, it is recommended to format a *primary* and an *alternate* couple data set on different devices, physical control units, and channels. All the updates in the primary are immediately copied in the secondary by XCF itself. If the primary data set fails, XCF switches automatically to the alternate data set. The alternate is now the primary couple data set. You can also perform a manual switch with the **`SETXCF COUPLE,PSWITCH`** command. Note that this command forces the alternate to be the primary, but the primary does not become the alternate, it is just deallocated. This allows you, for example, to:

- Increase the size of the primary couple data set
- Move the couple data set to a different device
- Change sysplex couple data set definitions
- Reformat the couple data set

Spare couple data set

It is recommended to pre-format a *spare* couple data set. With the **`SETXCF COUPLE,ACOUPLE`** command it is possible to define the spare data set as a new alternate couple data set, avoiding potentially having a single point of failure. The sysplex couple data sets (one type of couple data set) are *defined to the XCFs in the sysplex* with the PCOUPLE and ACOUPLE statements in the COUPLExx PARMLIB member.

Figure 1-30 on page 50 shows an example of formatting the sysplex couple data sets.

Tip: It is recommended that you place the primary sysplex couple data set on a different volume than the primary CFRM couple data set due to their high activities. You might allocate the primary sysplex couple data set together with the alternate CFRM couple data set on one volume, and vice versa.

1.26 Other couple data sets

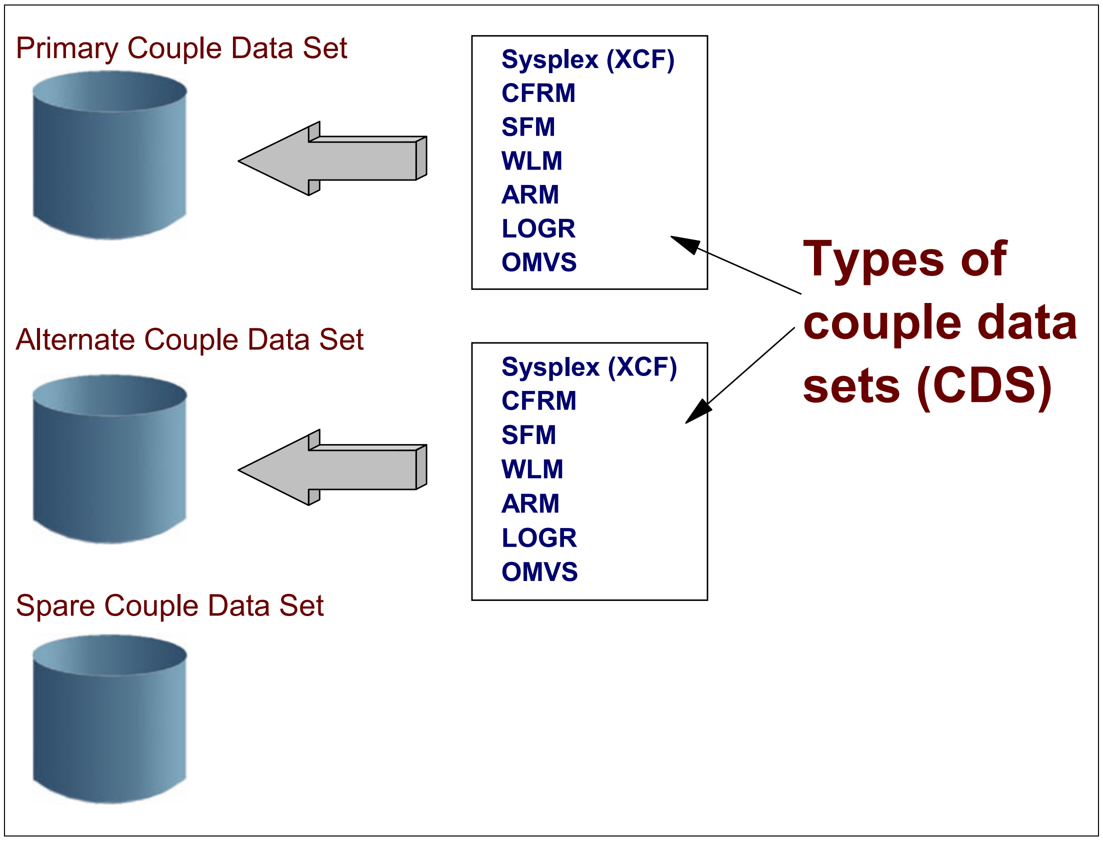


Figure 1-26 Other couple data sets

Couple data sets policy

A policy is a set of rules and actions that z/OS systems in a sysplex follow when using certain z/OS services. A policy allows z/OS to manage specific resources in compliance with system and resource requirements but with little operator intervention. Depending on the policy type, you can set up a policy to govern all z/OS systems in the sysplex or only a set of selected z/OS systems. You might need to define more than one policy to allow for varying workloads, configurations, or other installation requirements at different times. For example, you might need to define one policy for your prime shift operations and another policy for other times. Although you can define more than one policy of each type (except for System Logger), but *only one policy of each type can be active at a time.*

The same primary and alternate CDS rules apply as described for the sysplex CDS. There are different CDS types (pointed to by the sysplex CDS) that hold the different policies.

Coupling facility resource management (CFRM) CDS

This CDS contains several CFRM policies. A CF structure is created by CFCC when the first exploiter connects to it (IXLCONN macro). Each structure has several attributes. Parallel Sysplex design requires that some of them should be informed by the exploiter software and others by the installation. The CFRM CDS contains a CFRM policy, that describes the structures attributes that should be decided by the installation. In the CFRM policy, you define:

- The coupling facility HW accessed by the sysplex. Major parameters are:

  - CF name, sequence number, partition, cpcid, dumpspace
- All structures and their attributes, for example:
  - Structure name, size, initsize
  - The preferred CF name for the structure location
  - CF duplexing, allowance of structure size altering and structure full monitoring function

Any structure used in the Parallel Sysplex must be defined in the CFRM policy.

Sysplex failure management (SFM) CDS

This CDS contains several SFM policies. SFM policy allows the installation to define responses to:

- Signaling connectivity failures among XCFs
- Signaling connectivity failures in CF links
- System failures, indicated by a status update (heart beat) missing condition
- Reconfiguring systems in a PR/SM environment
- Sympathy sickness

Workload Manager (WLM) CDS

This CDS contains several WLM policies. In the WLM policy you define workloads, service classes describing goals for workloads, application and schedule environments, goals for non-z/OS partitions and others related to system performance management.

Automatic Restart Manager (ARM) CDS

This CDS contains several ARM policies. In the ARM policy you define how to process restarts and other events for started tasks and batch jobs that failed but previously have registered with Automatic Restart Manager through macros.

One example is CICS and DB2 running dependent on each other. ARM handles this so-called *restart group* as defined in the ARM policy, through the policy options:

- Start the *restart group* only on the same system
- Which system in the sysplex should do the restart
- Number of restart attempts
- Restart order

Important: It is important to know that Automatic Restart Manager is not to be used as a replacement for system automation, but an addition to.

System Logger (LOGR) CDS

This CDS contains several LOGR policies. You use the LOGR policy to define, update, delete and manage log streams. A log stream is a sequential data set describing events, where the records are sorted by a time stamp. In a sysplex environment is almost mandatory to have just one CICS log (for example) instead of one log per each z/OS where CICS address spaces are running. The LOGR policy is used by System Logger services to manage log data across the z/OSs in the Parallel Sysplex. It guarantees the single image of a log merging in chronological order all the log records. Exploiters of System Logger services are: CICS, VSAM TVS, RRS, LOGREC, OPERLOG, and others.

UNIX System Services (OMVS) CDS

The OMVS CDS *does not contain a policy*. It's used by XCF to maintain all the information to support UNIX System Services file system sharing across the sysplex. Recall that UNIX System Services is a z/OS component interfacing with a UNIX application running in z/OS.

1.27 Parallel Sysplex with couple data sets

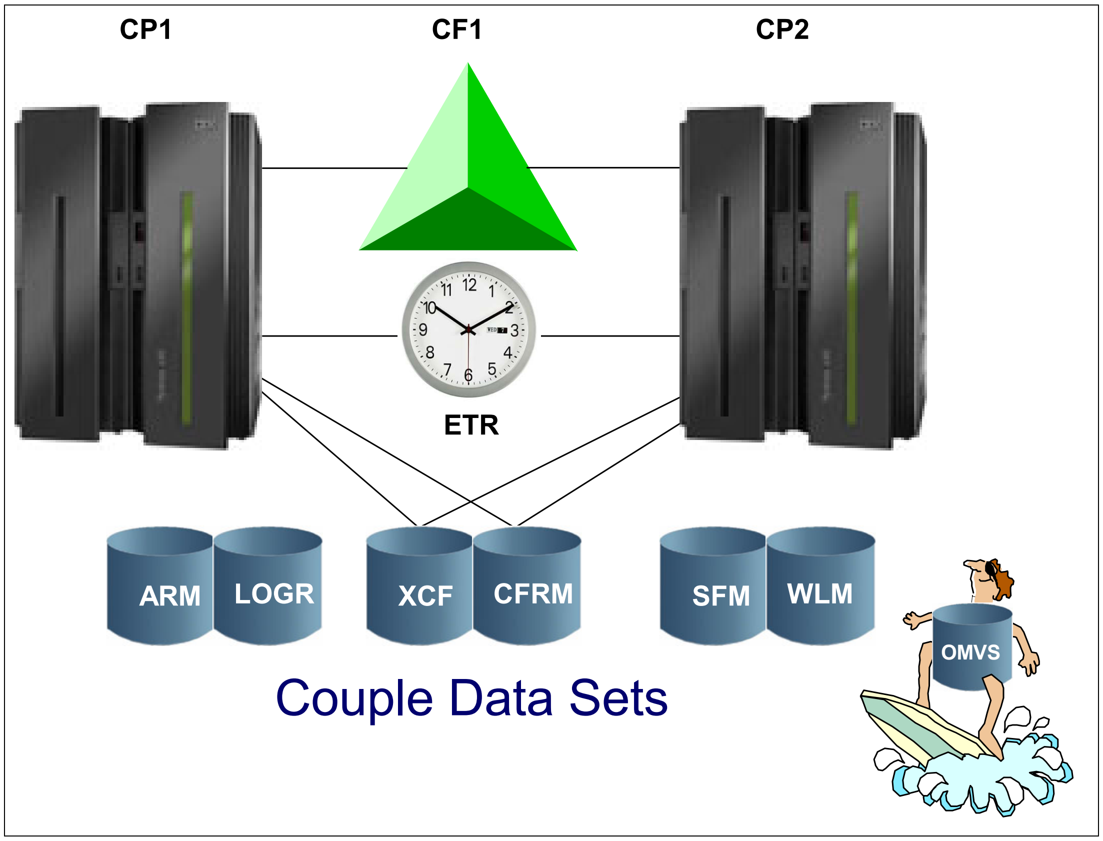


Figure 1-27 Parallel Sysplex and CDSes

Parallel Sysplex and CDSes

In order to full implement a Parallel Sysplex, you need to create and format the CDSes on DASD using the IXCL1DSU batch program utility. At least two of them are mandatory such as the sysplex CDS and CFRM CDS. See how to use the utility in “Format utility for couple data sets” on page 49.

Ensure that the CDSs are in DASD shared access by all sysplex members.

Defining an administrative policy

Once the CDSes for a service are formatted, you can define the administrative policies that reside on the data sets.

The CFRM, SFM, LOGR, and ARM administrative policies are defined, updated and deleted using the IXCMIAPU administrative data utility. IXCMIAPU is also used to provide reports describing the policies. Figure 1-28 on page 48 shows an example of how to use this utility.

The WLM policies are defined using the WLM ISPF (TSO) administrative application.

The sysplex CDS is created by using the COUPLExx PARMLIB member at IPL and can be altered later through the **SETXCF** console command.

Activating a policy

You activate a policy by issuing the following operator command:

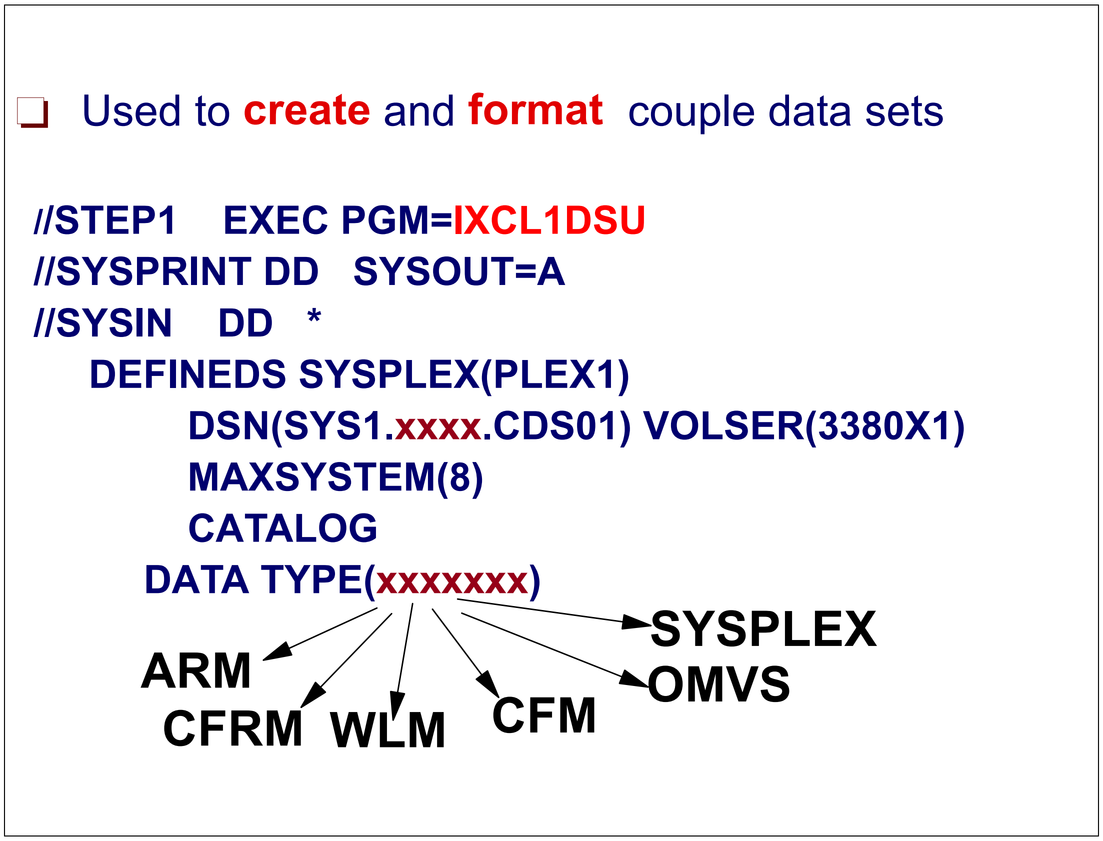
```
SETXCF START,POLICY,TYPE=type,POLNAME=policy name
```

This causes the system to access the administrative policy from the *type* CDS. The new policy replaces the old policy definitions and will be used to manage the related resources.

```
//STOECKE JOB  NOTIFY=&SYSUID,MSGLEVEL=(1,1),MSGCLASS=K,CLASS=A
//*****************************************************************
//*
//*   CFRM POLICY FOR PLEX1 CURRENT MEMBERS
//*
//*   /SETXCF START,POL,TYPE=CFRM,POLNM=POLX <===
//*
//*****************************************************************
//IXCCFRMP EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
     DATA TYPE(CFRM) REPORT(YES)
     DEFINE POLICY NAME(POL5) REPLACE(YES)
        CF NAME(CF1) TYPE(SIMDEV) MFG(IBM) PLANT(EN)
           SEQUENCE(0000000CFCC1) PARTITION(0) CPCID(00)
           DUMPSPACE(2000)
        CF NAME(CF2) TYPE(SIMDEV) MFG(IBM) PLANT(EN)
           SEQUENCE(000000CFCC2) PARTITION(0) CPCID(00)
           DUMPSPACE(2000)
        STRUCTURE NAME(IXCLST01) SIZE(40000) INITSIZE(4000)
                  PREFLIST(CF1,CF2) EXCLLIST(IXCLST02)
                  REBUILDPERCENT(1)
        STRUCTURE NAME(IXCLST02) SIZE(40000) INITSIZE(4000)
                  PREFLIST(CF1,CF2) EXCLLIST(IXCLST01)
                  REBUILDPERCENT(1)
        STRUCTURE NAME(ISGLOCK) SIZE(120000) INITSIZE(20000)
                  PREFLIST(CF1) REBUILDPERCENT(1)
        STRUCTURE NAME(LOGREC) SIZE(128000) INITSIZE(32000)
                  PREFLIST(CF1,CF2) REBUILDPERCENT(1)
        STRUCTURE NAME(OPERLOG) SIZE(128000) INITSIZE(32000)
                  PREFLIST(CF1,CF2) REBUILDPERCENT(1)
        STRUCTURE NAME(IRLMLOCKTABL) SIZE(8192)
                  PREFLIST(CF1,CF2) REBUILDPERCENT(1)
.......
        STRUCTURE NAME(DSNDB0M_GBP8K0) INITSIZE(3000)
                  SIZE(5000) PREFLIST(CF1,CF2)
                  DUPLEX(ENABLED)
        STRUCTURE NAME(DSNDB0M_GBP16K0) INITSIZE(3000)
                  SIZE(5000) PREFLIST(CF1,CF2)
                  DUPLEX(ENABLED)
```

Figure 1-28   Example of a CFRM policy definition job

1.28 Format utility for couple data sets


❑ Used to **create** and **format** couple data sets

```
//STEP1    EXEC PGM=IXCL1DSU
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     DEFINEDS SYSPLEX(PLEX1)
              DSN(SYS1.xxxx.CDS01) VOLSER(3380X1)
              MAXSYSTEM(8)
              CATALOG
          DATA TYPE(xxxxxxx)
```

Figure 1-29 Format utility for CDSs

Format utility for CDSs

The CDSs are created and formatted using the IXCL1DSU XCF format utility. You use the IXCL1DSU utility to format *all types* of CDSs for your sysplex.

This utility contains two levels of format control statements:

1. DEFINEDS - The primary format control statement identifies the CDS being formatted.
2. DATA TYPE - The secondary format control statement identifies the type of data to be supported in the CDS: sysplex, Automatic Restart Manager (ARM), CFRM, SFM, WLM, or LOGR data. In particular, note the recommendations about not over-specifying the size or the parameter value in a policy, which could cause degraded performance in a sysplex.

Formatting the sysplex CDS

Use the CDS format utility to create and format the *sysplex* CDSs prior to IPLing a system that is to use the sysplex CDSs. Other types of CDSs do not have to be formatted prior to IPLing the system.

Figure 1-30 shows the use of the format utility to format the *sysplex CDSes*. The values you specify in the MAXSYSTEM, GROUP and MEMBER keywords determine the CDS size.

```
//STEP1    EXEC PGM=IXCL1DSU
//STEPLIB  DD   DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     DEFINEDS SYSPLEX(PLEX1)
              DSN(SYS1.XCF.CDS01) VOLSER(3380X1)
              MAXSYSTEM(8)
              CATALOG
          DATA TYPE(SYSPLEX)
               ITEM NAME(GROUP)  NUMBER(50)
               ITEM NAME(MEMBER) NUMBER(120)
               ITEM NAME(GRS) NUMBER(1)
     DEFINEDS SYSPLEX(PLEX1)
              DSN(SYS1.XCF.CDS02) VOLSER(3380X2)
              MAXSYSTEM(8)
              CATALOG
          DATA TYPE(SYSPLEX)
               ITEM NAME(GROUP)  NUMBER(50)
               ITEM NAME(MEMBER) NUMBER(120)
               ITEM NAME(GRS) NUMBER(1)
/*
```

Figure 1-30 Formatting the sysplex CDS

1.29 Base sysplex

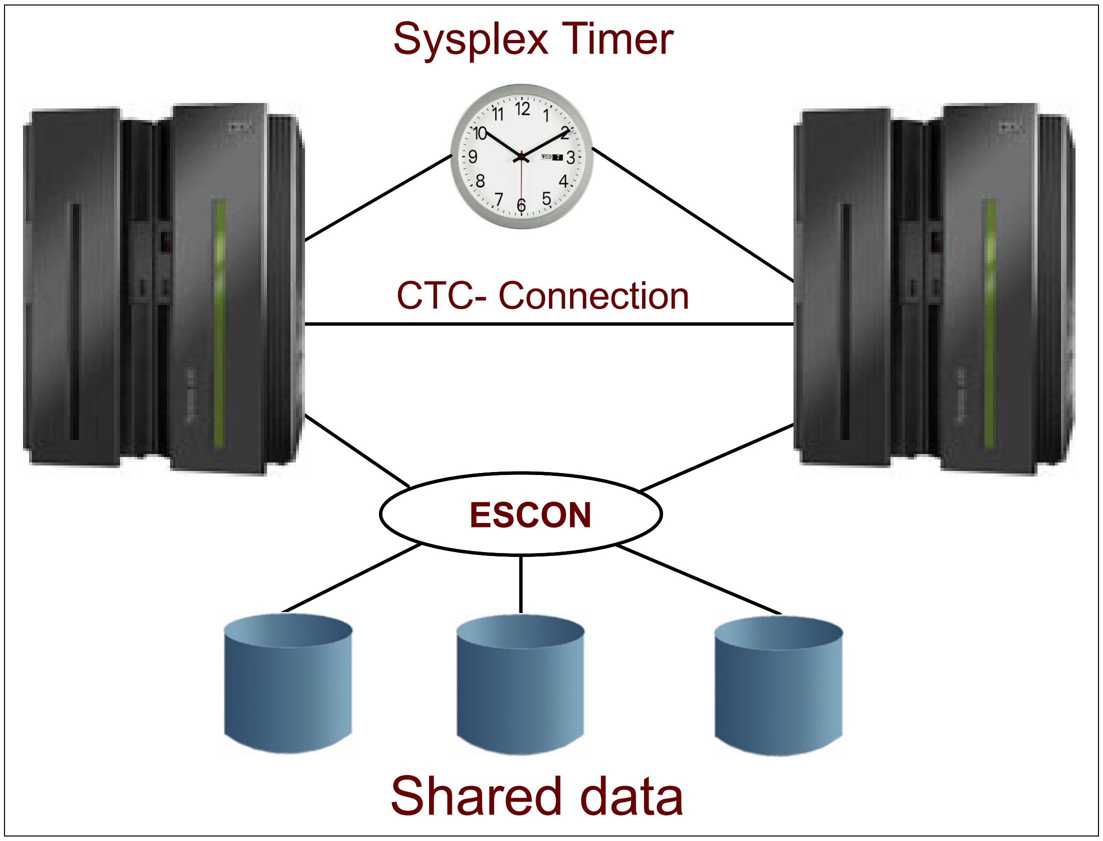


Figure 1-31 Base sysplex

Base sysplex

Recapping this concept, a base sysplex has:

- More than one z/OS image
- Shared DASD environment, meaning that the DASD volumes can be reached through channels by all the images
- XCF uses CTCA links and a sysplex CDS for communication with other XCFs in the sysplex
- External time synchronization between all z/OS images throughout a Sysplex Timer or a Server Time protocol (STP)

Simply stated, base sysplex is a step forward in a loosely coupled configuration.

Full signaling connectivity is required between all images in the sysplex. That is, there must be at least one inbound path (PATHIN) and one outbound path (PATHOUT) between each pair of images in the sysplex. XCF CTCA paths can be dynamically allocated and deallocated using the z/OS system command SETXCF START,PATHIN / PATHOUT. To avoid unplanned system outages due to signaling path failures, we recommend that you define multiple CTCA paths to XCF.

1.30 Parallel Sysplex

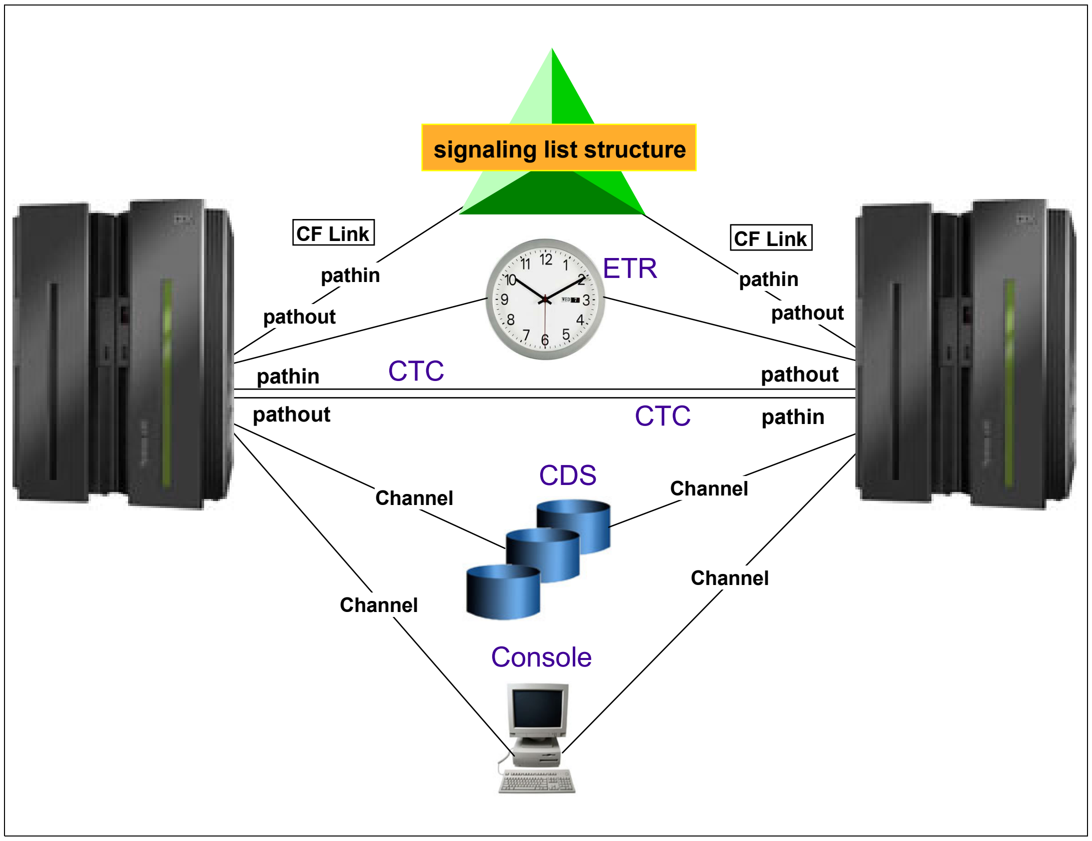


Figure 1-32 Parallel Sysplex

Parallel Sysplex

Parallel Sysplex architecture is generally characterized as a high performance shared data model. Each sysplex member has full read or write access control and globally managed cache coherency with high-performance and near-linear scalability. Specialized hardware and software cluster technology is introduced to address the fundamental performance obstacles that have traditionally plagued data-sharing parallel-processing systems.

Then recapping, Parallel Sysplex configuration has all the base sysplex components plus the coupling facility (CF). The core hardware technologies are embodied in the CF. The CF is used for data sharing together with the CF links, which enables the communication between the z/OS images and the CF. The same CF cannot be accessed by z/OS from different Parallel Sysplexes.

XCF signalling

The members of a sysplex use XCF signaling to communicate with each other.

- In a base sysplex this is done through CTCA connections.
- With Parallel Sysplex, signaling can be established through CF signaling list structures or CTCAs. A combination of both technologies is also possible.

Implementing XCF signaling through coupling facility list structures provides significant advantages in the areas of systems management, performance, recovery providing enhanced availability for sysplex systems. Signaling structures handle inbound and outbound traffic.

Signaling paths defined through CTCA connections must be exclusively defined as either inbound or outbound.

Because a single list structure can be defined for both inbound and outbound traffic, you can use the same COUPLExx PARMLIB member for each system in the sysplex. A CTCA connection, in contrast, cannot be defined to a single system as both inbound and outbound because the device numbers are different. Therefore, with CTCAs, you must specify a unique COUPLExx parmlib member for each system in the sysplex or configure your systems so that they can use the same COUPLExx member by over-specifying the number of devices for signaling paths and tolerating failure messages related to unavailable devices and other configuration errors.

Implementing XCF signaling through CF list structures also enhances the recovery capability of signaling paths and reduces the amount of operator intervention required to run the sysplex.

If XCF signaling is implemented through CF list structures, and if a CF that holds a list structure fails, or if connectivity to a CF that holds a list structure is lost, z/OS can rebuild the list structure in another available CF and reestablish signaling paths.

If the list structure itself fails, z/OS can rebuild it in the same CF or in another available CF and then reestablish signaling paths.

Parallel Sysplex support

The Parallel Sysplex supports up to 32 systems and significantly improves communication and data sharing among those systems. High performance communication and data sharing among a large number of z/OS systems could be technically difficult. But with Parallel Sysplex, high performance data sharing through a coupling technology gives high performance multisystem data sharing capability to authorized applications, such as z/OS subsystems.

Use of the CF by subsystems, such as CICS, IMS, DB2, VSAM, and others, ensures the integrity and consistency of data throughout the entire sysplex. The capability of linking many systems and providing multisystem data sharing makes the sysplex platform ideal for parallel processing, particularly for online transaction processing (OLTP) and decision support.

In short, a Parallel Sysplex builds on the base sysplex capability, and allows you to increase the number of CPCs and z/OS images that can directly share work. The CF enables high-performance, multisystem data sharing across all the systems. In addition, workloads can be dynamically balanced across systems with the help of workload management functions.

1.31 Cross-system extended services (XES)

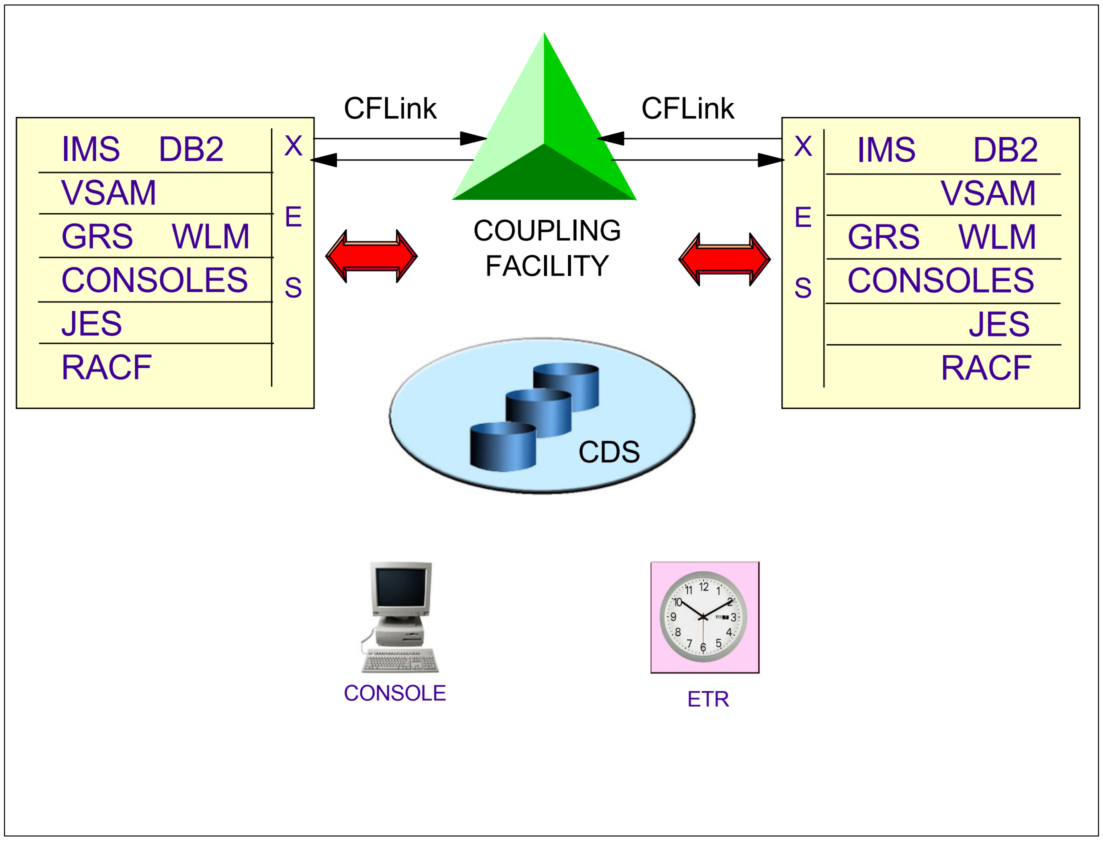


Figure 1-33 Cross-system extended services

Cross-system extended services (XES)

A coupling facility enables parallel processing and improved data sharing for authorized programs running in the sysplex. The cross-system extended services (XES) component of MVS enables applications and subsystems to take advantage of the coupling facility.

XES services

XES is a set of services that allows authorized applications or subsystems running in a sysplex to share data using a coupling facility.

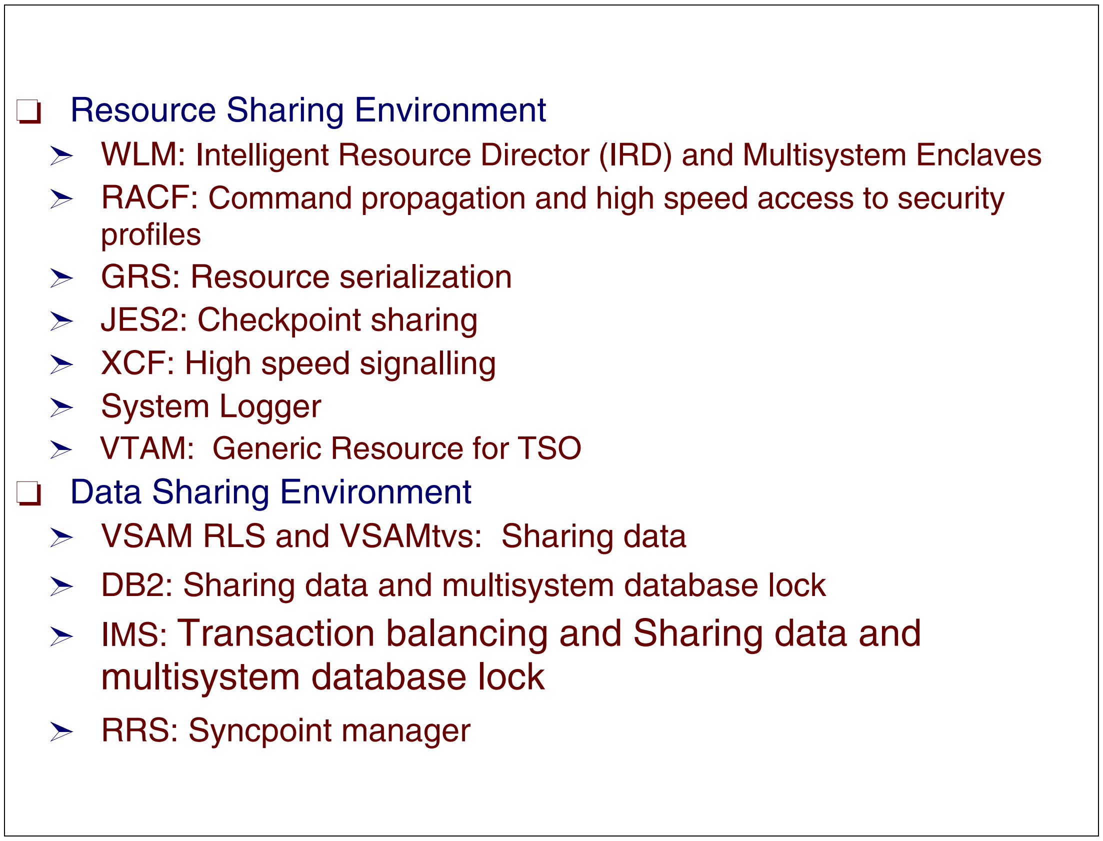
XES allows authorized applications or subsystems running in z/OS systems in a Parallel Sysplex to use the CF structures.

XES provides the sysplex services (through the IXL... macros) that applications and subsystems use to share data held in the CF structures. XES may access one or more CFs in a Parallel Sysplex to:

- Provide high-performance data sharing across the systems in a sysplex
- Maintain the integrity and consistency of shared data
- Maintain the availability of a sysplex

To share data, systems must have connectivity to the CF through CF channels. Systems in the sysplex that are using a CF must also be able to access the coupling facility resource management (CFRM) CDS, where some structure attributes are described.

1.32 Sharing environments



- ❑ Resource Sharing Environment
  - ➢ WLM: Intelligent Resource Director (IRD) and Multisystem Enclaves
  - ➢ RACF: Command propagation and high speed access to security profiles
  - ➢ GRS: Resource serialization
  - ➢ JES2: Checkpoint sharing
  - ➢ XCF: High speed signalling
  - ➢ System Logger
  - ➢ VTAM: Generic Resource for TSO
- ❑ Data Sharing Environment
  - ➢ VSAM RLS and VSAMtvs: Sharing data
  - ➢ DB2: Sharing data and multisystem database lock
  - ➢ IMS: Transaction balancing and Sharing data and multisystem database lock
  - ➢ RRS: Syncpoint manager

Figure 1-34 Sharing environments

Sharing environments

Parallel Sysplex exploitation can be classified according to how CF capabilities are used: to share system resources or to share data among transaction workloads.

Resource sharing environment

In a resource sharing environment, CF capabilities *are not used for production data sharing* such as IMS, DB2, or VSAM RLS. *Some* of the exploiters that use resource sharing to provide functions are:

- WLM: provides unique global performance goals and performance management at the sysplex level.
- RACF: Uses XCF for command propagation and CF for caching RACF data base in a store-through cache model. In the store- through cache model, all data updated (write) in the RACF buffer pool is staged to the CF cache structure and to DASD data set simultaneously. Eventually the next read will come from CF the cache structure. The use of Parallel Sysplex capabilities provides improved performance and scalability.
- JES2: JES2 uses the JES common coupling services (JES XCF) for communicating JES2 member status and other data among the JES2 XCF group members in a multi-access spool (MAS) configuration.
- XCF: Simplicity and even performance when using CF list structures for signaling.
- GRS star configuration: Uses CF structures for enqueues with overall better system performance and scalability compared to previous configuration (called ring).

- System Logger: There are two examples of exploiters using the System Logger for resource sharing. The first one is the operations log (OPERLOG) for console messages and commands, which records and merges messages about programs and system functions from each system in a sysplex. In Parallel Sysplex, OPERLOG can be used as a hardcopy medium instead of SYSLOG.

  The second one is the LOGREC log stream to record hardware failures, selected software errors, and selected system conditions across the sysplex. Using a LOGREC log stream rather than a logrec data set (normally named SYS1.LOGREC) for each system can streamline logrec error recording.
- VTAM: Using TSO generic resource allows for increased availability and workload balancing because all TSO/VTAM application programs can be accessed by the same generic name. The CF structure is used to keep the generic name and the individual application name (Applid). Thus Workload Manager can be used to make efficient use of system resources by selecting a partner session based on balanced load distribution.

Data sharing environment

Data sharing is also known as application-enabled Parallel Sysplex environment and is one in which data management subsystems communicate with each other to share data with total integrity using CF capabilities.

In this environment IMS, DB2, or VSAM/RLS data sharing is implemented and full benefits of the Parallel Sysplex technology are afforded, providing capability for dynamic workload balancing across z/OSs with high performance, improved availability for both planned and unplanned outages, scalable workload growth both horizontally and vertically, etc.

In a data sharing environment, installations can take full advantage of the ability to view their multisystem environment as a single logical resource space able to dynamically assign physical system resources to meet workload goals across the Parallel Sysplex environment.

1.33 Coupling facility structures

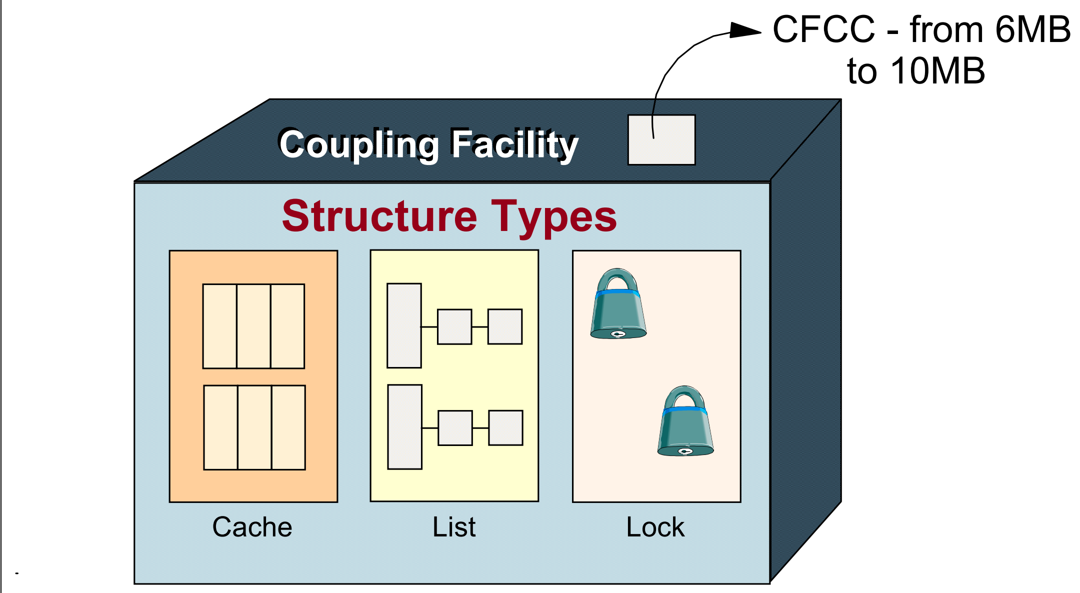


Figure 1-35 Coupling facility structures

Coupling facility structures

Within the CF, central storage is dynamically partitioned into structures. z/OS services (XES) manipulate data within the structures. There are three types of structures:

Cache structure

It has a directory and a data element area. Supplies a mechanism through the directory called buffer invalidation to ensure consistency of buffered data. Then, if the same DB2 page is duplicated in two DB2 local buffer pools, in two different z/OS and one copy is changed, then the other is invalidated because the information kept in the directory. The cache structure can also be used as a high-speed buffer (to avoid I/O operations) for storing shared data with common read/write access.

List structure

Enables authorized applications to share data that is organized in a set of lists, for implementing functions such as shared work queues and shared status information.

Lock structure

A lock structure supplies shared and exclusive locking capability for serialization of shared resources down to a very small unit of data (such as one lock per each DB2 page). Each lock manager control the locks related to the data that the transaction from its z/OS is accessing. One lock manager running in one z/OS can see the locks of other lock managers spread in the other z/OS systems. All the data in a data base is shared which allows updates from any data base manager.

Structure size

A CF structure is created when the first user (exploiter connection) requests connection to the structure. The size and the CF location for the structure allocation depends on:

- Values specified in the CFRM policy: Initial, maximum, smaller initial size, and minimum size.
- A minimum possible size as determined by the CFCC level.

  A coupling facility is a special logical partition that runs the coupling facility control code (CFCC) and provides high-speed caching, list processing, and locking functions in a sysplex.

  Note: You should be aware of your application's CFLEVEL requirements. Different levels of coupling facility control code (CFCC) support different coupling facility functions.

- Structure size specified by the authorized application (exploiter) when using the XES services
- CF storage constraints
- CF storage increment

We highly recommend to use the *CF Structure Sizer* tool (CFSIZER) to plan your structure sizes. The sizing recommendations are always done for the highest available CFCC level. The CFSIZER is provided via the following:

`http://www-1.ibm.com/servers/eserver/zseries/cfsizer/`

1.34 DB2 cross invalidation (1)

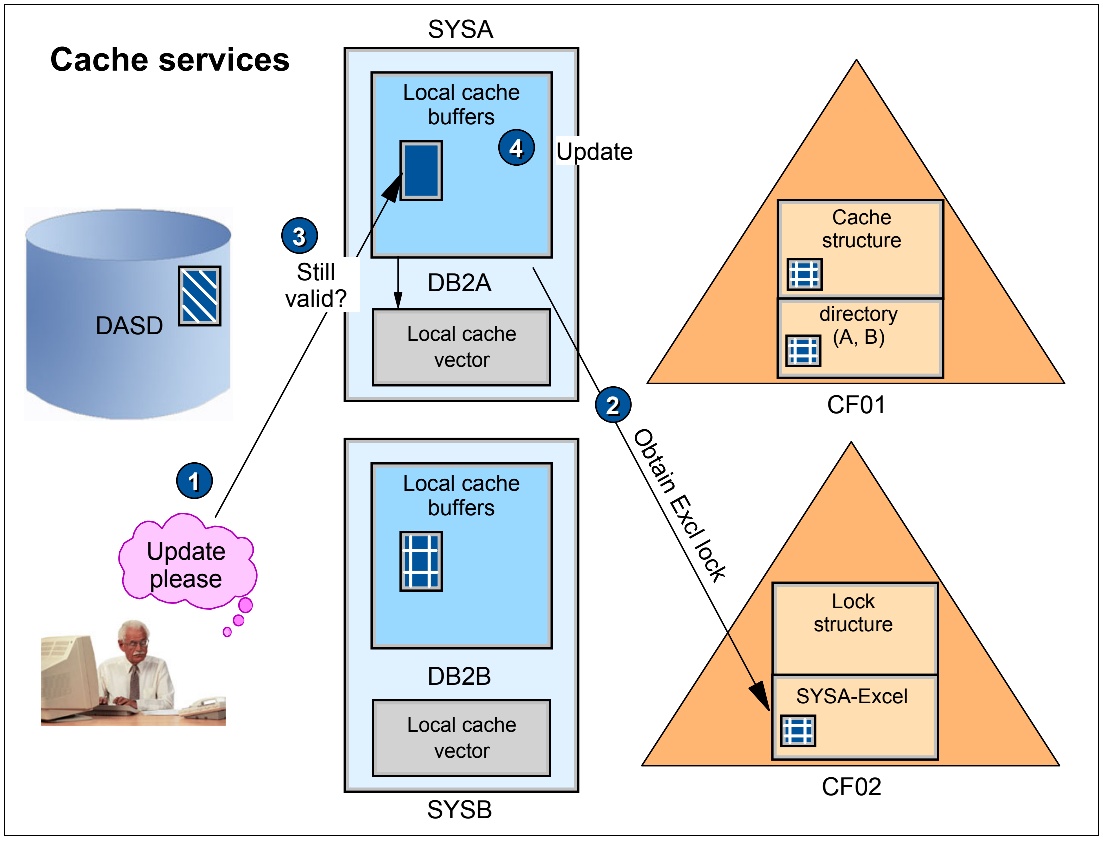


Figure 1-36 DB2 Cross invalidation

DB2 cross invalidation

Let us follow an example on DB2 to see how the CF structures help in keeping data integrity. The scenario has DB2A and DB2B located in SYSA and SYSB, both in the same sysplex. There is a DB2 page with several valid copies: DASD, DB2A local buffer, DB2B local buffer, data element area in a CF cache structure. On the cache structure directory, the CFCC has also the information about the two copies in the local buffer pools.This information is passed to the CFCC by each DB2 when each copy of the page is loaded in the respective local buffer pool.

Now an end user (could be CICS) send a transaction to update that pages contents. To guarantee integrity of a lock (related to the page), an exclusive mode is requested and granted from the lock structure in the CF.

DB2 verifies if the page copy in the local buffer is still valid. All the copies are valid, so the update is done in that copy. This update creates a situation in which all the n-1 copies are invalid because they are not current. This lack of currency affects read integrity and needs to be overcome. To solve the problem, see “DB2 cross invalidation (2)” on page 60.

1.35 DB2 cross invalidation (2)

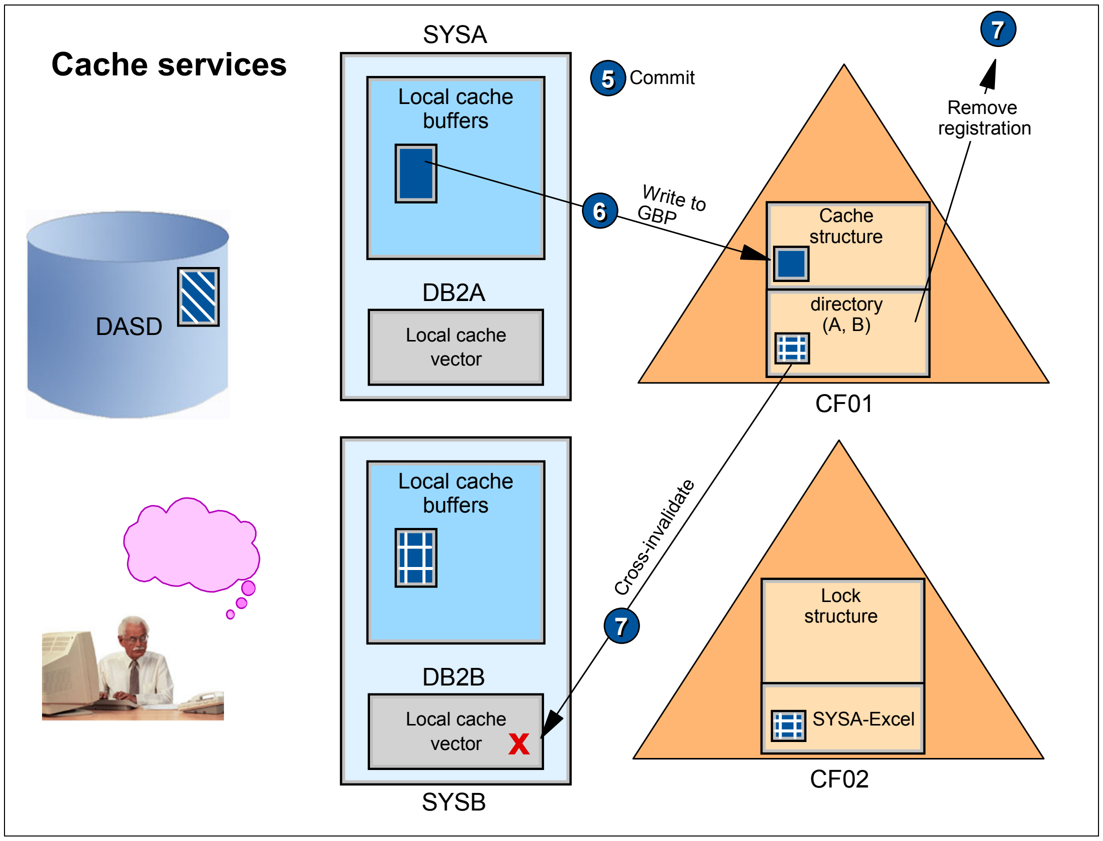


Figure 1-37 DB2 Cross invalidation

DB2 cross invalidation

DB2A informs the CFCC that the page contents was modified and also send a new copy to the data element area in the cache structure.

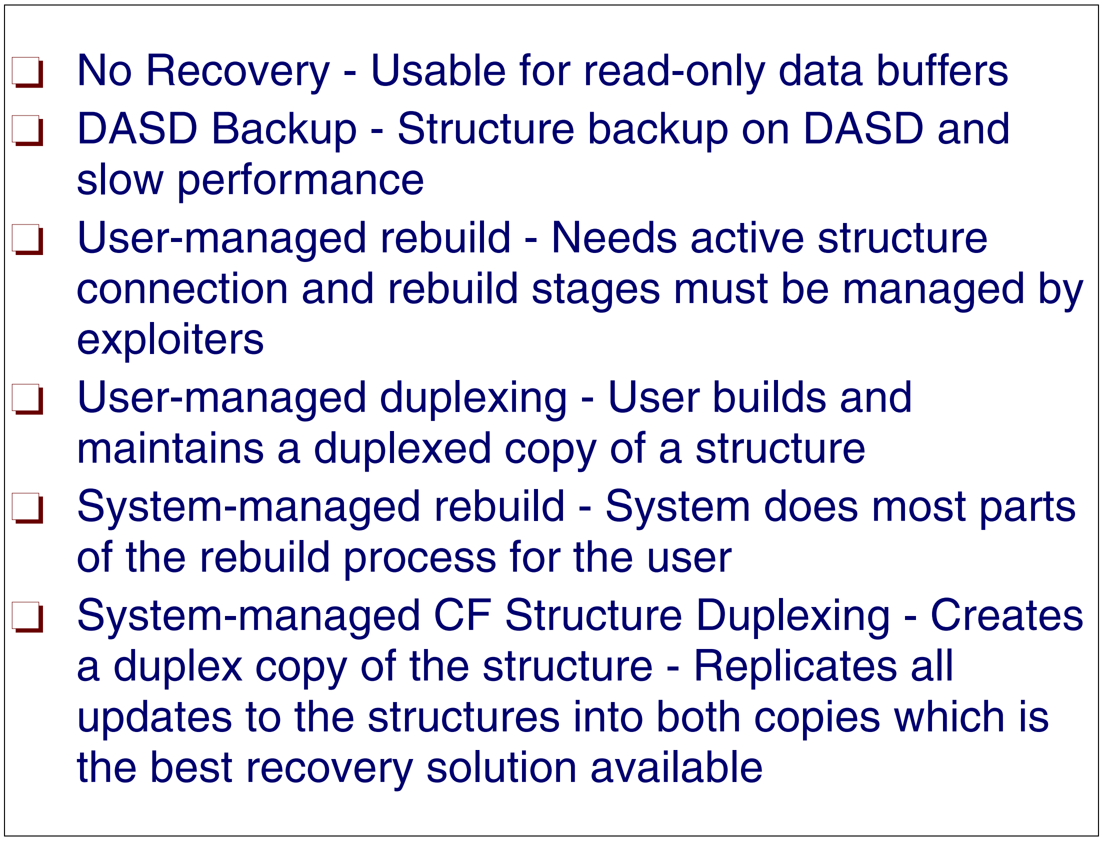
The CFCC by consulting the cache directory learns that DB2 has an valid copy of the page.

Then, asynchronously through the CF link, the CFCC turns on an invalid bit in a local cache vector area accessed by DB2B. There is a bit in every page in the DB2B local buffer pool. Obviously, the directory information about such a copy is erased.

Later, when DB2B needs a valid copy of the page, the testing (and consequent switch off) will tell DB2B that the copy is invalid. In this case, DB2B asks the CFCC for a valid copy that maybe is still in the data element area of the cache structure.

This algorithm is called cross invalidation and it guarantees the read integrity.

1.36 Level of Parallel Sysplex recovery



- No Recovery - Usable for read-only data buffers
- DASD Backup - Structure backup on DASD and slow performance
- User-managed rebuild - Needs active structure connection and rebuild stages must be managed by exploiters
- User-managed duplexing - User builds and maintains a duplexed copy of a structure
- System-managed rebuild - System does most parts of the rebuild process for the user
- System-managed CF Structure Duplexing - Creates a duplex copy of the structure - Replicates all updates to the structures into both copies which is the best recovery solution available

Figure 1-38 Evolution of parallel sysplex recovery

No recovery

Structures with no recovery mechanism contain typically either read-only or scratch-pad like data. A valid read-only copy is kept in DASD, for example. Recovery is not a major issue for these data types.

DASD backup

Some structures recover from hard failures by maintaining a hardened copy of the data on DASD. System Logger's use of a staging data set is an example. The disadvantage is the synchronous write to DASD. The I/O operation is much slower in comparison to a CF structure update.

User-managed rebuild

User-managed rebuild was introduced along with initial CF support. It provides a robust failure recovery capability for CF structure data. z/OS XES coordinates a structure rebuild process with all active connectors to the structure. But it also requires significant amounts of supporting code from the exploiters to handle different complex recovery scenarios.

Structures duplexing

None of the above approaches are ideal. Several have significant performance overheads associated with them during mainline operation. One example is the cost of synchronously hardening data out to DASD in addition to the CF. Some of them compromise availability in a failure scenario by involving a potentially lengthy rebuild or log recovery process, during which the data is unavailable. The log merge and recovery for an non-duplexed DB2 group buffer pool cache is another example.

CF duplexing is designed to address these problems by creating a duplexed copy of the structure prior to any failure and maintaining that duplexed copy during normal use of the structure. This is done by transparently replicating all updates to the structure into both copies. The result is a robust failure recovery capability through failover to the unaffected structure instance. This results in:

- An easily exploited common framework for duplexing the structure data contained in any type of CF structure, with installation control over which structures are/are not duplexed.
- High availability in failure scenarios by providing a rapid failover to the unaffected structure instance of the duplexed pair with very little disruption to the ongoing execution of work by the exploiter and application.

Prior to duplexing, a structure could be recovered in one of these ways:

1. Copying the data from the existing structure instance to a new one. This is fast, but for large structures it can take more time. This option is available only for cache structures and link failure scenarios, not for CF failures. The source structure is lost in case of a CF failure.
2. Recreating the data from in-storage data in the connected systems. This option is slower than option number one.
3. Reading the required data from DASD. This option often involves restarting the affected subsystems and can be really slow.
4. Throw everything away and start with a new empty structure.

For duplexed structures, there is no recovery involved. The loss of a CF or connectivity to a CF no longer requires that the structure be rebuilt. Instead, the system automatically reverts back to simplex mode for the affected structure, dropping the affected copy of the structure in the process. So the recovery time is somewhere from a little faster to significantly faster.

User-managed duplexing

User-managed duplexing support was introduced in OS/390 R3. There are two structures (each one in each CF), the primary and the secondary. The exploiter is in charge of keeping both structures in synchronization and do the switch over to the failure-unaffected structure copy. Both instances of the structure are kept allocated indefinitely. This is only supported for cache structures. DB2’s group buffer pool cache structure technique is one of the exploiters.

System-managed rebuild

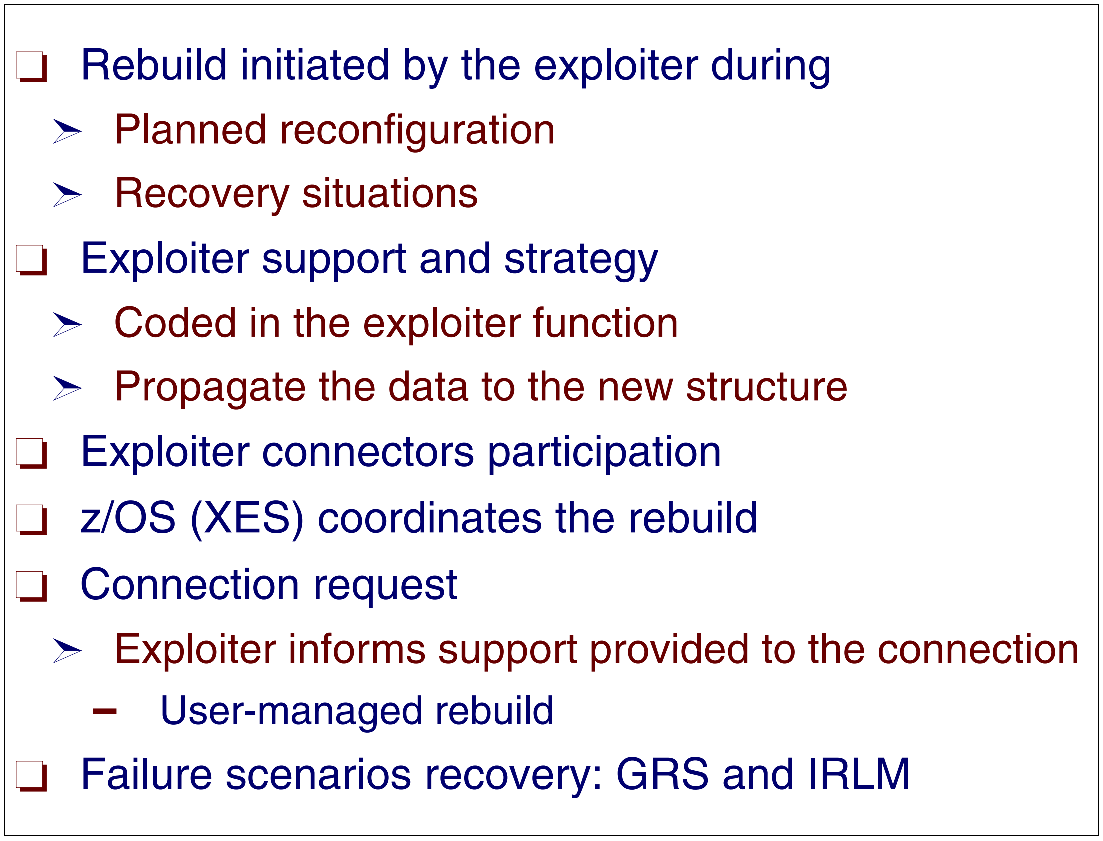
System-managed rebuild was introduced in OS/390 R8. This process allows to manage several aspects of the user-managed process that formerly required explicit support and participation from the connectors (exploiters). XES internally allocates the new structure and propagates the necessary structure data to the new structure. Then it switches over to using the new structure instance. XES is not capable of rebuilding the structure just by itself in failure scenarios but removes most of the difficult rebuild steps from user responsibility.

System-managed CF structure duplexing

In system-managed CF structure duplexing XES is much more involved than in user-managed duplexing, then relieving the exploiter code of a complex logic. However, because it is a generic implementation its performance is not so good as the previous one. IRLM, that is the DB2 lock manager is the main exploiter of such duplexing. For more details refer to the technical white paper: *System-Managed CF Structure Duplexing,* GM13-0103 available at:

```
http://www-1.ibm.com/servers/eserver/zseries/library/techpapers/gm130103.html
```

1.37 User-managed rebuild



- ❑ Rebuild initiated by the exploiter during
  - ➢ Planned reconfiguration
  - ➢ Recovery situations
- ❑ Exploiter support and strategy
  - ➢ Coded in the exploiter function
  - ➢ Propagate the data to the new structure
- ❑ Exploiter connectors participation
- ❑ z/OS (XES) coordinates the rebuild
- ❑ Connection request
  - ➢ Exploiter informs support provided to the connection
    - – User-managed rebuild
- ❑ Failure scenarios recovery: GRS and IRLM

Figure 1-39 User-managed rebuild

User-managed rebuild

Most CF exploiters have implemented structure rebuild protocols to resume service promptly and subsequently allow the business to continue processing with minimal impact to service. User-managed rebuild is used in recovery situations and in planned structure reconfiguration. Because does not require "old" structure to still be available, so can recover from CF failure. In user-managed rebuild each CF exploiter implements its own rebuild strategy to suit its needs. The exploiter instances using structures rebuild their own structures. z/OS XES assists and coordinates the rebuild process and may in fact initiate the rebuild process. In a failure scenario, only the CF exploiter instances can determine whether rebuilding a structure is possible or not.Exploiter understands contents, so can optimize performance

Each set of same instance exploiters has its own rebuild strategy. Certain exploiters require all the data resident in the CF structure to be available to the surviving instances for a successful rebuild of the structure. This situation occurs when required information is spread among all the instances accessing the structure with no one instance having a complete view. The basic principle of these exploiters is: the loss of a single component is rare, the loss of two critical components is extremely rare. Thus these instances implement quick, automatic recovery following the failure of a single component; but accept longer and perhaps a more operator-intensive recovery should a double failure scenario occur.

A structure in other CF exploiters may have access to all the data required (if any) to handle the loss of a connected instance, the structure or an instance and the structure. Again, it depends on how the structure is used and the underlying assumptions each exploiter made with respect to the degree of loss and the probability of the loss.

Data sharing and resource sharing failure scenarios

Let us discuss an example of data sharing exploiter in three failure scenarios. IRLM is a component that provides locking services for DB2 and IMS; an IRLM instance is an address space in a z/OS system in which the IRLM code is running.

1. *The failure of any one IRLM instance*

   All the remaining IRLM instances can obtain information from the IRLM lock structure, which allows them to continue granting DB2 or IMS locks while protecting the resources owned by the failed IRLM instance until such time as the recovery action against the failed IRLM instance is taken.

2. *Loss of access to the IRLM lock structure*

   The active IRLM instances have enough information internally to rebuild and repopulate the IRLM lock structure. This recovery action is done without operator interaction and is completed in a matter of seconds. No one IRLM instance has enough information to rebuild the structure in this scenario. It takes a cooperative effort among all of the IRLMs to rebuild the structure.

3. *Simultaneous loss of IRLM lock structure and the failure of an IRLM instance connected to the lock structure (the double failure scenario)*

   Rebuilding the structure is not possible. The surviving IRLM instances do not have sufficient information to rebuild the lock structure and protect the resources owned by the failed IRLM instances. The surviving IRLM instances abort the rebuild process and disconnect from the lock structure (in a failed persistent state). The database managers using the IRLM are unable to continue because locks cannot be granted or released. Recovery in this situation is neither automatic nor quick demanding data base forward recovery.

IRLM is failure-dependent and CF exploiters in such situations require a failure-independent environment. An example of an almost failure-independent environment is to avoid to locate the CF and the z/OS participating of the same Parallel Sysplex in the same CPC, to decrease the probability of a double failure. Another example could be duplexing the structure.

Now we show an example of a resource-sharing exploiter in the same failure scenarios.

GRS is a z/OS component that serializes access to resources through enqueue and reserve of z/OS services. We must recall that when a JOB ends abnormally (Abend) all the holding ENQs are automatically dequeued. When implemented in star configuration, GRS uses CF lock structures to hold sysplex-wide enqueues. Each GRS instance maintains only a local copy of its own global resources while the GRS CF lock structure has the overall image of all systems.

- *In case of the failure of any number of z/OS images*, enqueues held by the failed images are discarded by the GRS in the remaining systems.
- *In case of loss of access to the GRS lock structure,* the active GRS instances rebuild the lock structure into the alternate CF.
- *In case of the simultaneous loss of the GRS lock structure and the failure of any system images,* the surviving GRS images repopulate the alternate CF structure with enqueues they held at the time of failure.

GRS is a failure-independent CF exploiter and *does not require* a failure-independent environment. However, a failure-independent environment provides faster recovery. The two examples of the same rebuild strategy implementation with different effects on availability. IRLM needs a failure-independent environment because the data locked by the failed instance can compromise data integrity and availability. On the other hand, GRS does not require failure independence because the resources held by the failed system are released when the system dies.

1.38 User-managed duplexing

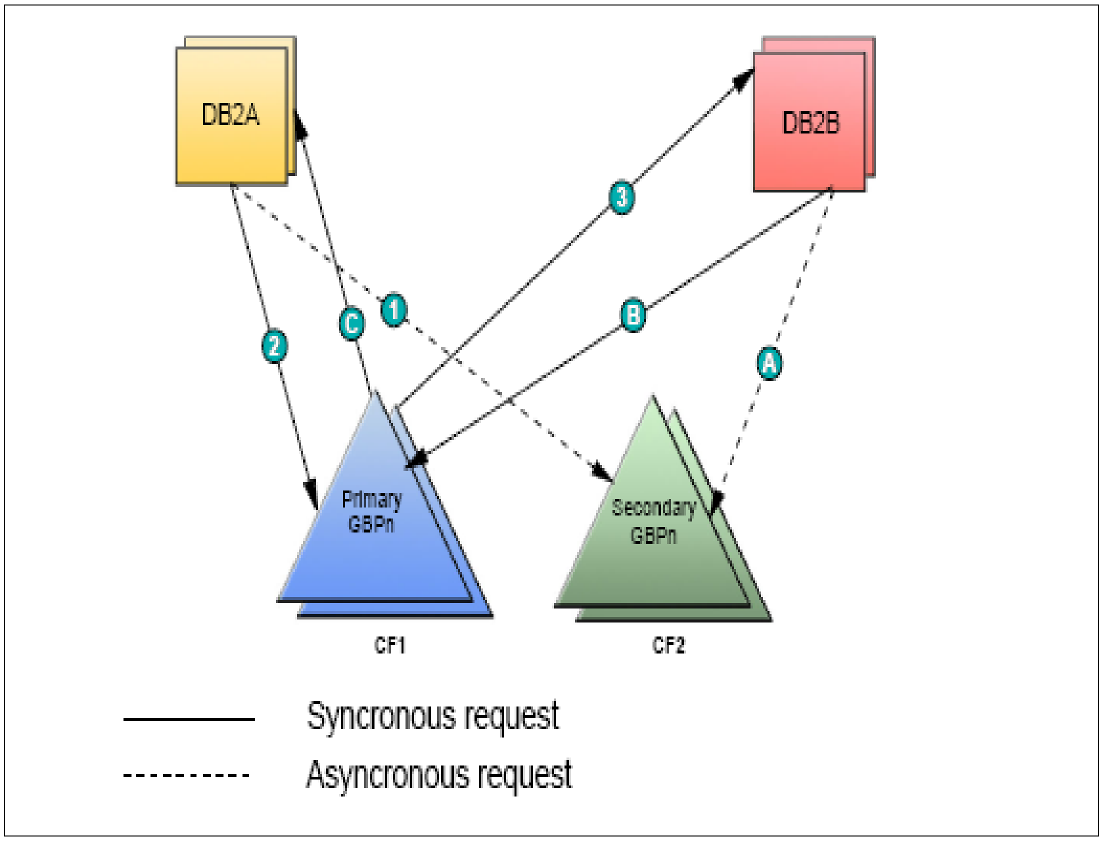


Figure 1-40 User-managed duplexing

User-managed duplexing

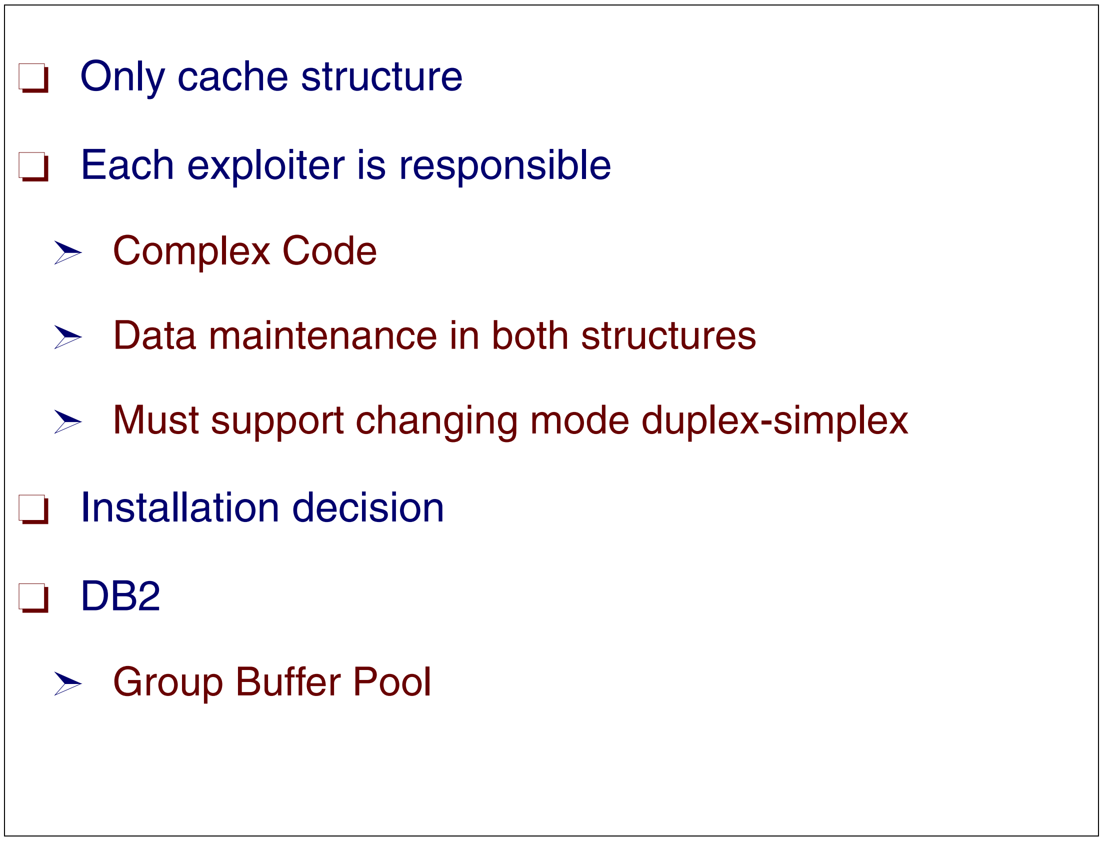
There are two types of rebuild processing, rebuild and duplexing rebuild. The method by which the rebuild processing is accomplished can be either user-managed or system-managed. User-managed duplexing rebuild, a variation of the structure rebuild process, is available only for cache structures. For duplexing to occur,all connectors to the structure must specify not only ALLOWDUPREBLD=YES but also ALLOWREBLD=YES when connecting to the structure.

Structure rebuild and duplexing rebuild provide the framework by which an application can ensure that there is a viable and accurate version of a structure being used by the application.

Structure rebuild allows you to reconstruct the data in a structure when necessary, for example, after a failure. Duplexing rebuild allows you to maintain the data in duplexed structures on an ongoing basis, so that in the event of a failure, the duplexed structure can be switched to easily. Duplexing rebuild is the solution for those applications that are unable or find it difficult to reconstruct their structure data after a failure occurs.

There are two structures (each one in each CF), the primary and the secondary. The exploiter is in charge of keeping both structures in synchronization and do the switch over to the failure-unaffected structure copy. Updates are propagated to both structures (the writes to the secondary are asynchronous) and reads are only from the primary. Both instances of the structure are kept allocated indefinitely. This is only supported for cache structures. DB2's group buffer pool cache structure technique is one of the exploiters.

1.39 User-managed duplexing rebuild



- Only cache structure
- Each exploiter is responsible
  - Complex Code
  - Data maintenance in both structures
  - Must support changing mode duplex-simplex
- Installation decision
- DB2
  - Group Buffer Pool

Figure 1-41 User-managed duplexing rebuild

User-managed duplexing rebuild

User-managed duplexing rebuild was integrated into OS/390 V2R6 and is available *only to CF cache structures*. User-managed duplexing rebuild allows you to allocate another cache structure in a different CF for the purpose of duplexing the data in the structure to achieve better availability and usability. Although called a rebuild process, it is in fact a data duplication process or data synchronization between the two structures. So, in case of failure, the structure in an alternate CF is used.

The CF exploiter must support user-managed duplexing rebuild, since the exploiter is responsible for managing both instances of the structure and, when necessary, can revert to using only one of the structures. The connectors (exploiters) to the structure must participate in the defined protocol to accomplish the type of procedure—rebuild or duplexing rebuild. The connector is responsible for construction of the new instance and maintaining the duplicate data for a duplexed structure.

Simplex/duplex modes

In addition to the types of duplexing methods provided, a coupling facility structures can also be duplexed automatically. System-managed duplexing rebuild provides the capabilities of keeping two instances of a coupling facility structure, one a duplicate of the other. When there are two duplicate structure instances, the structure is considered to be in duplex-mode. If there is only one structure instance, the structure is considered to be in simplex-mode.

Installation decision

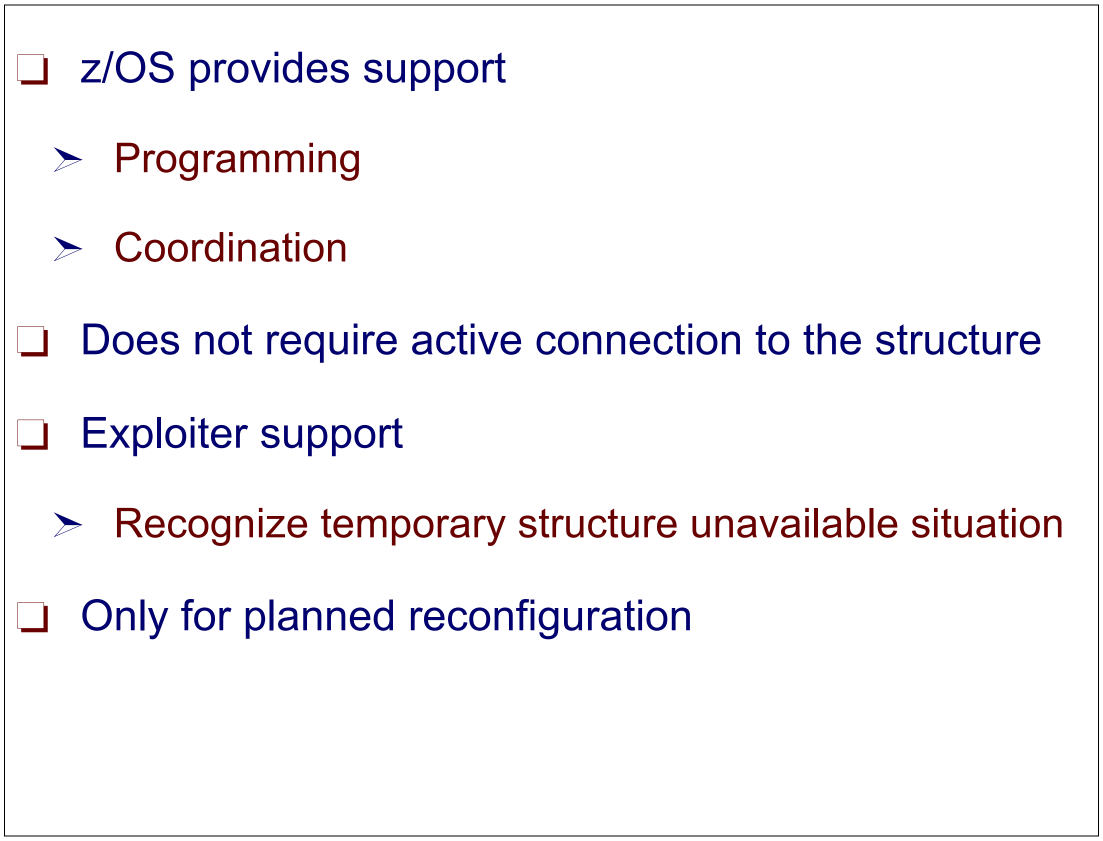
User-managed rebuild requires complex programming on the part of the product that uses the CF structure. The entire rebuild process has to be managed by the product. This includes tasks such as:

- Coordinating activity between all the connectors to stop any access to the structure until the rebuild is complete
- Working with other connectors of the structure to decide who rebuilds which parts of the structure content
- Recovering from unexpected events during the rebuild process
- Handling any errors that may arise during the process

DB2 group buffer pools

At this time, CF structure duplexing rebuild is supported by DB2 for its group buffer pools (GBP) cache structures. GBP is used for cross invalidation and as a cache for DB2 pages (to avoid DASD I/Os). DB2 writes to a primary GBP, it simultaneously writes to the secondary, duplexed GBP. Registering interest and reads are only done to the primary structure. Since both GBP instances are kept in sync with each other in terms of changed data, there are no concerns about the loss of the primary GBP because the data is still available in the alternate CF.

1.40 System-managed rebuild



- ❑ z/OS provides support
  - ➢ Programming
  - ➢ Coordination
- ❑ Does not require active connection to the structure
- ❑ Exploiter support
  - ➢ Recognize temporary structure unavailable situation
- ❑ Only for planned reconfiguration

Figure 1-42 System-managed rebuild

System-managed rebuild

System-managed duplexing rebuild is a process managed by the system that allows a structure to be maintained as a duplexed pair. The process is controlled by CFRM policy definition as well as by the subsystem or exploiter owning the structure. The process can be initiated by operator command (**SETXCF**), programming interface (IXLREBLD), or be MVS-initiated. Note that user-managed duplexing rebuild is controlled and initiated in the same manner as system-managed duplexing rebuild but is managed by the subsystem or exploiter owning the structure and applies only to cache structures.

System-managed rebuild is intended for use in *planned reconfiguration scenarios* and provides structure rebuild with minimal participation from structure connectors (exploiters).

System-managed rebuild provides the protocols to coordinate rebuild processing and provides the support necessary for rebuild. It does not require any active exploiter instance connected to the CF structure being rebuilt. The system-managed method requires that connectors recognize that the structure will become temporarily unavailable for requests, but does not require them to develop protocols to coordinate rebuild.

In the connection request, the exploiter states whether that connection supports system-managed processes or not. The first request to a structure determines the processes supported by all the connections to that structure.

System-managed rebuild *cannot* be used in a failure scenario because the primary structure must be available.

1.41 System-managed CF duplexing

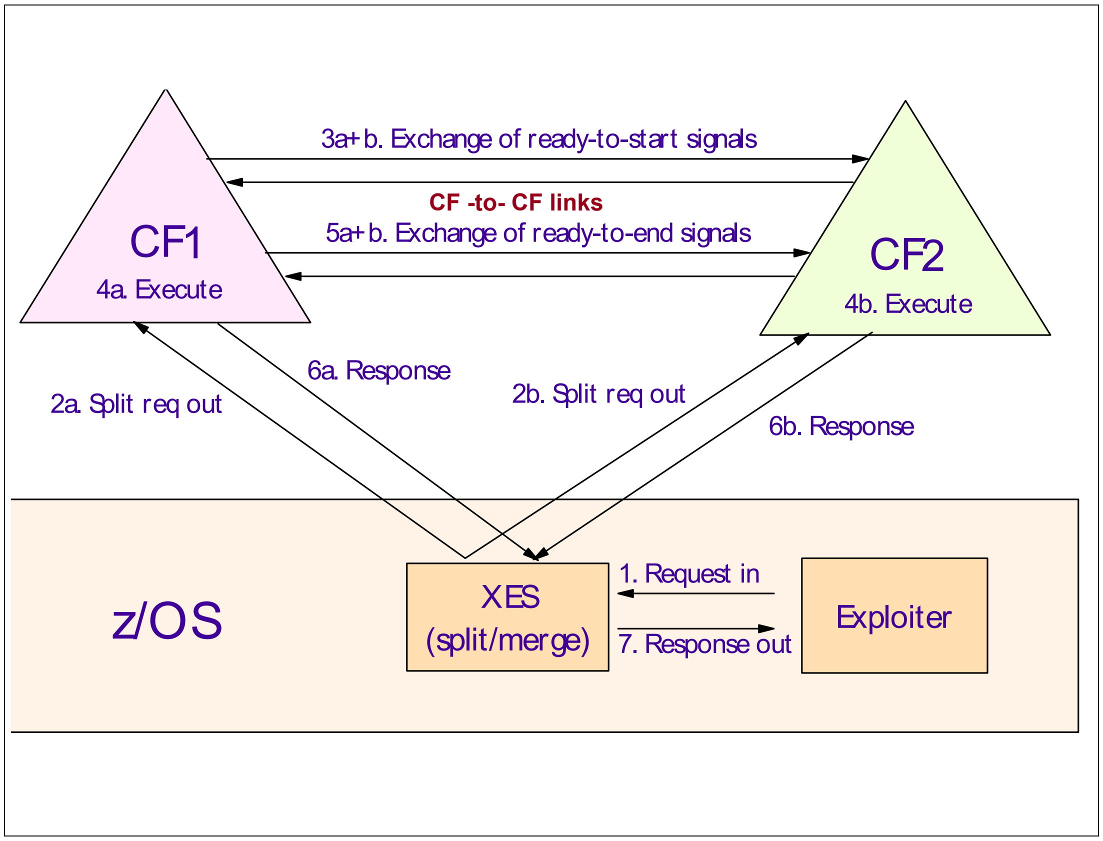


Figure 1-43 System managed duplexing

System-managed duplexing

The following requirements must be considered before system-managed duplexing rebuild can be established in a Parallel Sysplex environment. Figure 1-43 on page 69 shows the additional CF processing involved with system-managed duplexing.

1. XES receives an exploiter *request in.*
2. XES sends the *request out* to CF1 and CF2.
3. CF1 and CF2 exchange signals *ready-to-start.* At CFLEVEL 15 this exchange is dropped in normal situations to improve the performance.
4. Both CFs *execute* the write request.
5. When finished, the CFs exchange signals *ready-to-end.*
6. Each CF sends the *response* to XES.
7. When XES receives the reply from CF1 and CF2 the *response out* is returned to the exploiter.

The decision to enable a particular structure for system-managed duplexing is done at the installation level, through the DUPLEX keyword in the CFRM policy.

System-managed duplexing benefits

System-managed duplexing rebuild provides:

- Availability

Since the data is already in the second CF, a faster recovery is achieved. Any application that provides support for system-managed processes can participate in either system-managed rebuild or system-managed duplexing rebuild.

- Manageability and usability are provided by a consistent procedure to set up and manage structure recovery across multiple exploiters, it can significantly improve operability, resulting in improved availability.
- Enablement makes structures that do not have rebuild capability, like CICS temp storage and MQSeries®, viable in a high-availability environment.
- Reliability is a common framework that makes possible less effort on behalf of the exploiters, resulting in more reliable subsystem code.
- Cost benefits enables the use of non-standalone CFs

System-managed duplexing costs on resources

The system-managed duplexing costs depend on which structures are being duplexed and how they are being invoked by the applications. The structure updates impact:

- z/OS CPU utilization - Processing relating to send and receive two CF messages and response (instead of one).
- Coupling facility CPU utilization - The CF containing the structure now has to process requests that update the duplexed structure. The impact of processing these requests may have already been planned for to handle a CF rebuild situation in a simplex environment. Additional CF usage for both CFs is incurred to handle the CF-to-CF communication to coordinate the updates.
- Coupling facility link usage - Additional traffic on the links due to the additional requests to the secondary structure.
- The CF-to-CF links.
- z/OS CF link subchannel utilization - Will increase since the z/OS-to-CF response times increase due to the CF-to-CF communication.
- CF storage requirements need not increase. Although a new structure is now required on the second CF, this space should have already been planned for to handle the CF rebuild situation.

Costs versus availability installation decision

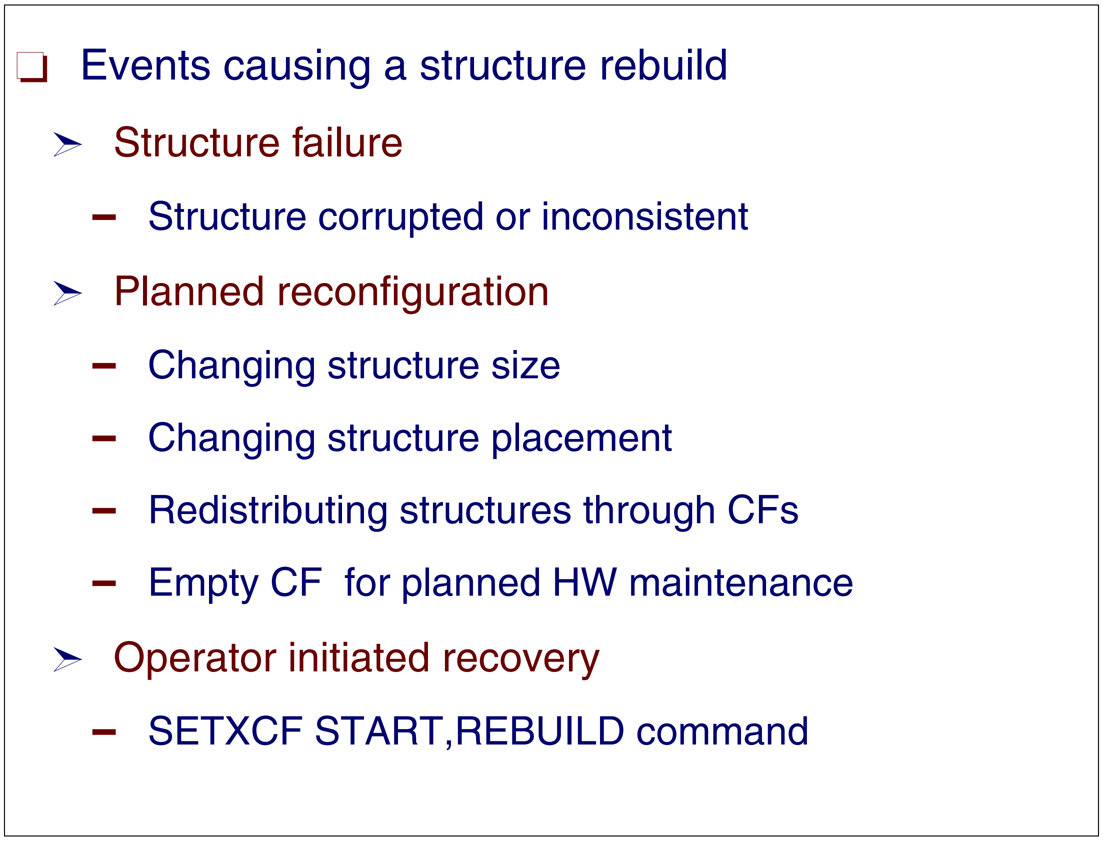
Even in a hardware and software configuration that fully supports system-managed duplexing rebuild, you must consider, on a structure by structure basis, whether the availability benefits to be gained from system-managed duplexing rebuild for a given structure outweigh the additional costs associated with system-managed duplexing for that structure. The availability benefits depend on considerations such as:

- Whether or not the structure exploiter currently supports rebuild
- Whether that rebuild process works in all failure scenarios
- How long that rebuild process generally takes, among others, compared to the rapid failure recovery capability that duplexing provides

The cost benefits depend on considerations such as:

- The hardware configuration cost - Possible additional CF processor and CF link capacity requirements
- Software configuration cost - Possible upgrades to installed software levels
- The coupling efficiency cost of duplexing the structure given its expected duplexing workload

1.42 CF structure rebuild



- Events causing a structure rebuild
  - Structure failure
    - Structure corrupted or inconsistent
  - Planned reconfiguration
    - Changing structure size
    - Changing structure placement
    - Redistributing structures through CFs
    - Empty CF for planned HW maintenance
  - Operator initiated recovery
    - SETXCF START,REBUILD command

Figure 1-44 CF structure rebuild

CF structure rebuild

Here we are recapping the CF structure rebuild function. The reasons for rebuilding can be:

- Structure failures caused by a corruption or inconsistency
- Planned structure reconfiguration as structure size changes. Moving the structure to another CF for load balancing or to empty a CF for HW maintenance.
- Operator-initiated rebuild with SETXCF command:

  ```
  SETXCF START,REBUILD,STRNAME=strname
  ```

  You can use this command to recover a hung structure connector, for instance.

It's important to know that a structure can only be rebuilt if there is an active structure connection and if the structure exploiter allows the rebuild.

The key to high availability in a Parallel Sysplex environment is the ability to quickly and without operator intervention recover from the failure of a single component. Rebuild is a process that involves the construction of a new instance of the CF structure. There are detailed structure rebuild and recovery scenarios in *z/OS MVS Setting Up a Sysplex,* SA22-7625.

A structure is allocated after the first exploiter connection request. The connection request parameter controls whether the connector supports structure alter (change the size dynamically), user-managed processes, or system-managed processes.

1.43 Parallel Sysplex availability

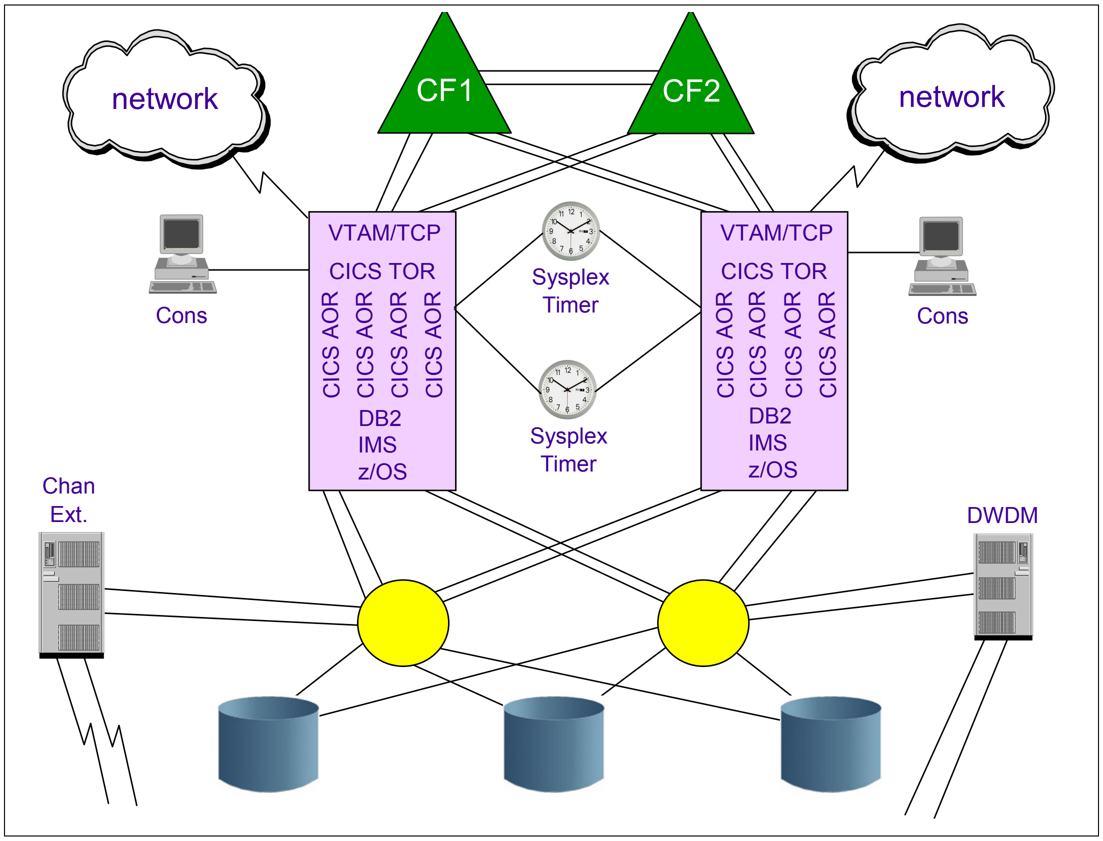


Figure 1-45 Parallel Sysplex availability

Parallel Sysplex availability

To get the availability benefits enabled by a Parallel Sysplex you must make the effort to exploit the capabilities it provides. In general it is possible to size a Parallel Sysplex to any of the availability levels required:

- High availability (HA) - no unplanned disruptions in the service caused by errors
  - Delivers an acceptable or agreed service during scheduled periods.
- Continuous operation (CO) - planned outages do not disrupt the service
  - These outages sometimes are needed to maintain the software and hardware. The redundancy caused by data sharing should not disrupt the service, but may affect performance due to less available capacity.
- Continuous availability (CA) - the 24 hours by seven days in the week (24 by 7).
  - CA combines HA and CO techniques.
  - We may say that 24 by 7 is the goal for installations exploiting all the Parallel Sysplex features and recommendations. The problem here is the occurrence of multiple concurrent failures, as in a disaster situation. Parallel Sysplex is not ready (and nobody else is) for concurrent multiple failures. In certain cases the 24 by 7 still can be delivered for double failures, but no more.

Continuous availability is a balance between costs and benefits. The fundamentals that contribute to achieving your availability goals are the following:

- Quality of technology
  - Reliable components
  - Redundancy in systems and network
  - Capacity to support failures
  - Exploitation of product availability features
  - Resiliency and fast initialization
  - Isolation of function
  - Controlled diversity
  - Security controls
  - Automation
  - End-to-end monitoring tools
- Robust applications and data
  - Applications designed for high availability
  - Redundant data and applications
  - Minimal defects and fast recovery
- Skilled people
  - Competence and experience
  - Awareness and forethought
  - Adequate depth of resources
  - Proactive focus
  - Management support
- Effective processes
  - Service level objectives aligned with business
  - High availability strategy and architecture
  - Predefined recovery and incident procedures
  - Robust testing
  - Documented configuration and procedures
  - Effective incident management
  - Proactive problem prevention
  - Minimal risk associated with change

For more detailed information, refer to: *Achieving the Highest Levels of Parallel Sysplex Availability,* SG24-6061and to the following website for availability information:

```
http://www-1.ibm.com/servers/eserver/zseries/pso/availability.html
```

1.44 CF configuration examples

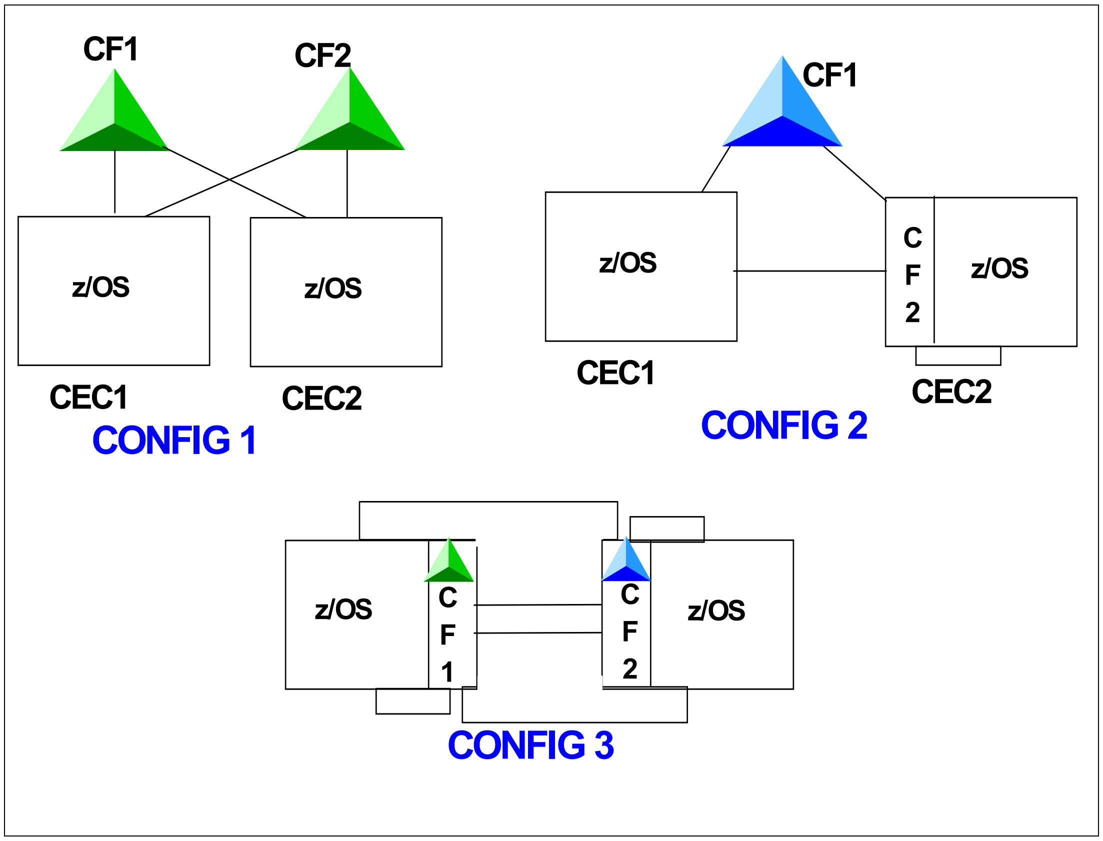


Figure 1-46 CF configuration samples

CF configuration samples

The CF is one component of a highly available Parallel Sysplex. To get high availability for a CF, you must eliminate single points of failure by designing the Parallel Sysplex based on the following factors:

- Number of CF LPs
- Number of ICF processors
- Placement of the CF LPs
  - Standalone or not
  - CF structure duplexing
- CF link technology, number of links and links shared or not
- Placement of CF structures

The robustness of the CF configuration depends on:

- The application or subsystem (exploiters) that utilizes the CF structures
- The location of the structure
- The capability of the CF exploiter of reacting to a double failure recovery scenario, that is, the simultaneous loss of a structure and a coupling exploiter connected to the structure

Applications using CF services may require different failure isolation techniques. Application-enabled environments (the ones exploiting data sharing) have more severe availability characteristics than resource sharing environments. Data sharing configurations exploit the CF cache and lock structures in ways that involve sophisticated recovery mechanisms, and some CF structures require failure independence to minimize recovery times and outage impact.

Configuring the Parallel Sysplex environment for high availability is closely tied to isolating the CF from the LPs on which software exploiters are executing (also isolating CFs from one another) to remove single points of failure.

CF LP on standalone CPCs

The standalone CF provides the most robust CF capability, because the CPC is wholly dedicated to running the CFCC code, that is, all of the processors (PUs), CF links, and memory are for CF use only. The various standalone CF models provide for maximum Parallel Sysplex connectivity. The maximum number of configurable PUs and memory is tied to the physical limitations of the associated machine family (that is, z900 Model 100 9-way, or z990 with 32 CPs).

Additionally, given the physical separation on the CF from production workloads, CFs can be independently upgraded and maintenance applied with minimal planning and no impact to production workloads. Also, a total failure in the standalone CPC causes just a single failure, because just one CF is down.

When choosing such a configuration, each CF LP from the same Parallel Sysplex must run in a different CPC, as shown in CONFIG1 in Figure 1-46 on page 74. That configuration is the most robust and also the most expensive. It is recommended for data sharing production environments without system-managed duplexing.

CF LP on a CPC running z/OSs of the same sysplex

A CF LP running on a CPC having a z/OS image in the *same* Parallel Sysplex introduces a potential for a single point of failure in a *data sharing environment.* This single point of failure is eliminated when using duplexing for those structures demanding a failure-independent environment.

When you choose a configuration such as CONFIG3 in Figure 1-46, and are *not using* system-managed duplexing, and you have an exploiter requiring that its structure be placed in a failure-independent environment, you must ensure that the CF is not in the same configuration as the z/OS systems that access it. For example, placing the CF in a CPC with one or more additional LPs that are running z/OS to access the CF would not provide a failure-independent environment.

Additionally, in this configuration, when the general purpose CPC is disruptively upgraded, any ICF processor in that CPC is also upgraded with no additional fee, while for a standalone CF additional expenses are needed for upgrade. On the other hand, this upgrade causes a CF outage and a z/OS outage at the same time.

Mixed configuration

A balance of availability and performance can be obtained by configuring a single standalone CF containing structures that require failure isolation, with an ICF that contains structures not requiring failure isolation, such as CONFIG2 in Figure 1-46. CF structure duplexing can then be enabled for those structures where the benefit outweighs the cost. This can typically be for those structures that have no fast method of recovery such as CICS Temporary Storage and others.

1.45 Parallel Sysplex exploiters

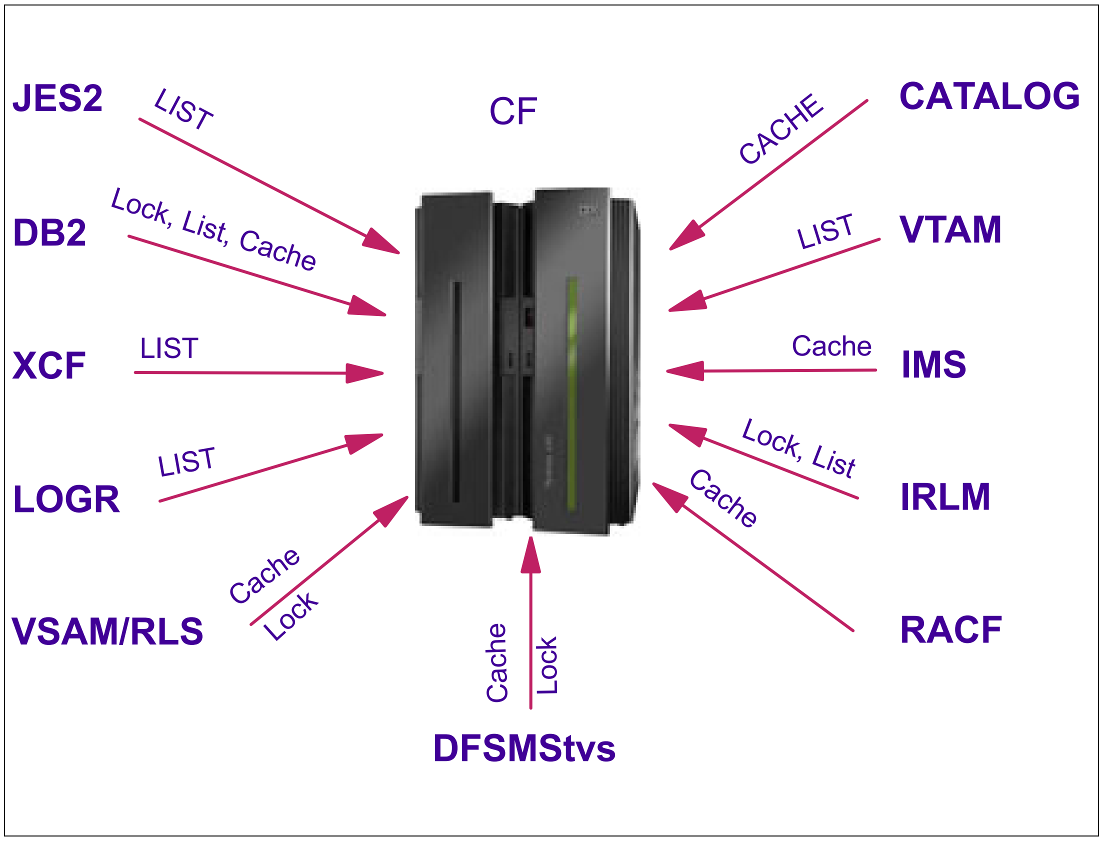


Figure 1-47 Parallel Sysplex exploiters

Parallel Sysplex exploiters

Authorized applications, such as subsystems and z/OS components in the sysplex, can use the CF services to cache data, share queues and status, and access sysplex lock structures in order to implement high-performance data-sharing and rapid recovery from failures. The subsystems and components transparently provide the data sharing and recovery benefits to their customer applications.

Some IBM data management systems that use the CF include database managers and a data access method, as follows:

- Information Management System Database Manager (IMS DB) is the IBM strategic hierarchical database manager. It is used for numerous applications that depend on its high performance, availability, and reliability. A hierarchical database has data organized in the form of a hierarchy (pyramid). Data at each level of the hierarchy is related to, and in some way dependent upon, data at the higher level of the hierarchy.

  IMS database managers on different z/OS systems can access data at the same time. By using the CF in a sysplex, IMS DB can efficiently provide data sharing for more than two z/OS systems and thereby extends the benefits of IMS DB data sharing. IMS DB uses the CF to centrally keep track of when shared data is changed. IRLM is still used to manage data locking, but does not notify synchronously each IMS DB of every change. IMS DB does not need to know about changed data until it is ready to use that data.

- DATABASE 2 (DB2) is the IBM strategic relational database manager. A relational database has the data organized in tables with rows and columns.

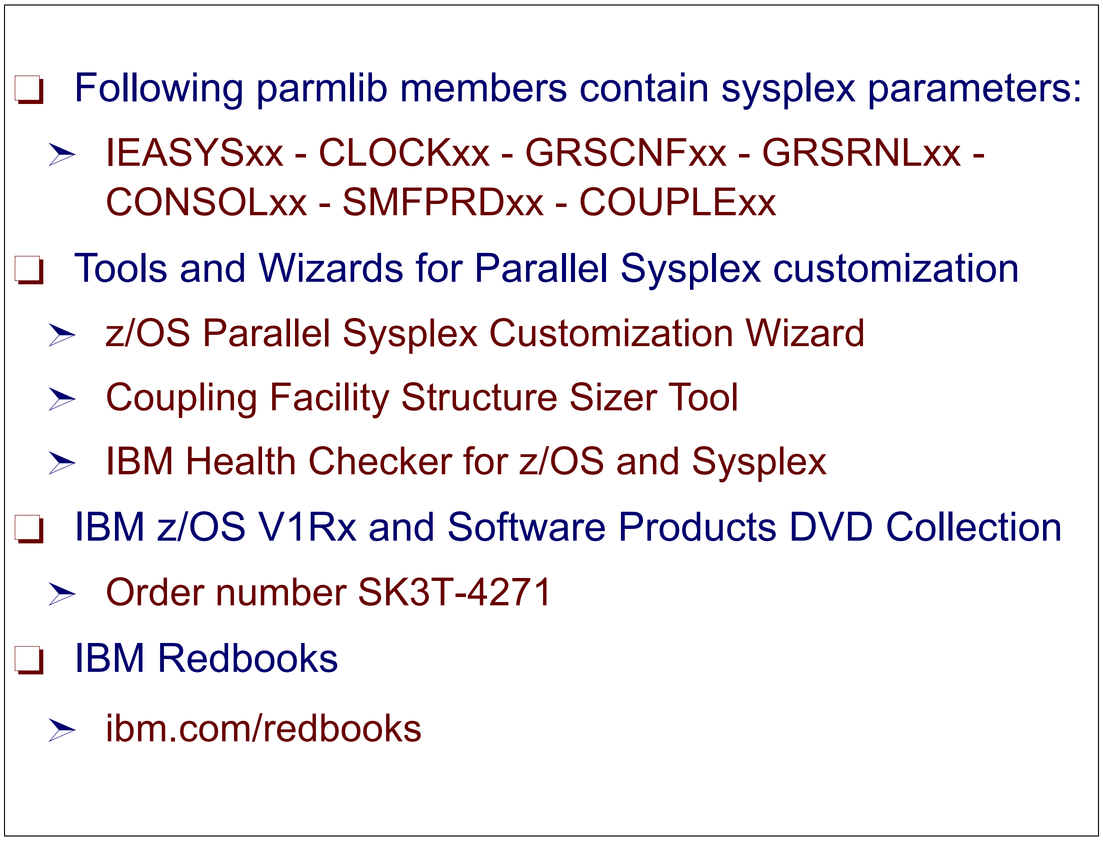
DB2 data-sharing support allows multiple DB2 subsystems within a sysplex to concurrently access and update shared databases. DB2 data sharing uses the CF to efficiently lock, to ensure consistency, and to buffer shared data. Similar to IMS, DB2 serializes data access across the sysplex through locking. DB2 uses CF cache structures to manage the consistency of the shared data when located in local buffers. DB2 cache structures are also used to buffer shared data within a sysplex for improved sysplex performance.

- Virtual Storage Access Method (VSAM), a component of DFSMSdfp, is an access method rather than a database manager. It is an access method that gives CICS and other application programs access to data stored in VSAM data sets.

  VSAM supports an data set accessing mode called record-level sharing (RLS). RLS uses the CF to provide sysplex-wide data sharing for CICS and the other applications that use the accessing mode. By controlling access to data at the record level, VSAM enables CICS application programs running in different CICS address spaces, called CICS regions, and in different z/OS images, to share VSAM data with complete integrity. The CF provides the high performance data-sharing capability necessary to handle the requests from multiple CICS regions.
- DFSMStvs, a transactional VSAM access method, provides a level of data sharing with built-in transactional recovery for VSAM recoverable files that is comparable to the data sharing and transactional recovery support provided by DB2 and IMS databases. In other words, DFSMStvs provides a log. The objective of DFSMStvs is to provide transactional recovery directly within VSAM. It builds on the locking, data caching and buffer coherency functions provided by VSAM RLS, using the CF hardware to provide a shared data storage hierarchy for VSAM data. It adds the logging and two-phase commit and back out protocols required for full transactional recovery capability and sharing.
- Enhanced catalog sharing (ECS). Catalog sharing requires that all changes to the catalog be communicated to all z/OSs to ensure data integrity and data accessibility. To maintain the integrity of the catalog, each system uses a combination of device locking and multiple I/Os to access a “shared record” in DASD for each of the catalogs. ECS enhances catalog sharing in a Parallel Sysplex cluster through the use of the CF. ECS provides a new catalog sharing method that moves the contents of the “shared record” to a cache structure in the CF. This allows z/OS images to access this information at CF speeds, eliminating most of the GRS and I/O activity required to access shared catalogs.

In addition to data management systems, there are other exploiters of the CF, such as Resource Access Control Facility (RACF) or the Security Server element of z/OS, and JES2. Transaction management systems (as CICS or IMS/DC) also exploit the CF to enhance parallelism.

1.46 Defining the sysplex



- Following parmlib members contain sysplex parameters:
  - IEASYSxx - CLOCKxx - GRSCNFxx - GRSRNLxx - CONSOLxx - SMFPRDxx - COUPLExx
- Tools and Wizards for Parallel Sysplex customization
  - z/OS Parallel Sysplex Customization Wizard
  - Coupling Facility Structure Sizer Tool
  - IBM Health Checker for z/OS and Sysplex
- IBM z/OS V1Rx and Software Products DVD Collection
  - Order number SK3T-4271
- IBM Redbooks
  - ibm.com/redbooks

Figure 1-48 Defining the sysplex

Defining the sysplex

The following SYS1.PARMLIB members contain sysplex parameters. Most of them are discussed in the subsequent chapters.

- IEASYSxx
- CLOCKxx
- GRSCNFxx
- GRSRNLxx
- CONSOLxx
- SMFPRDxx
- COUPLExx

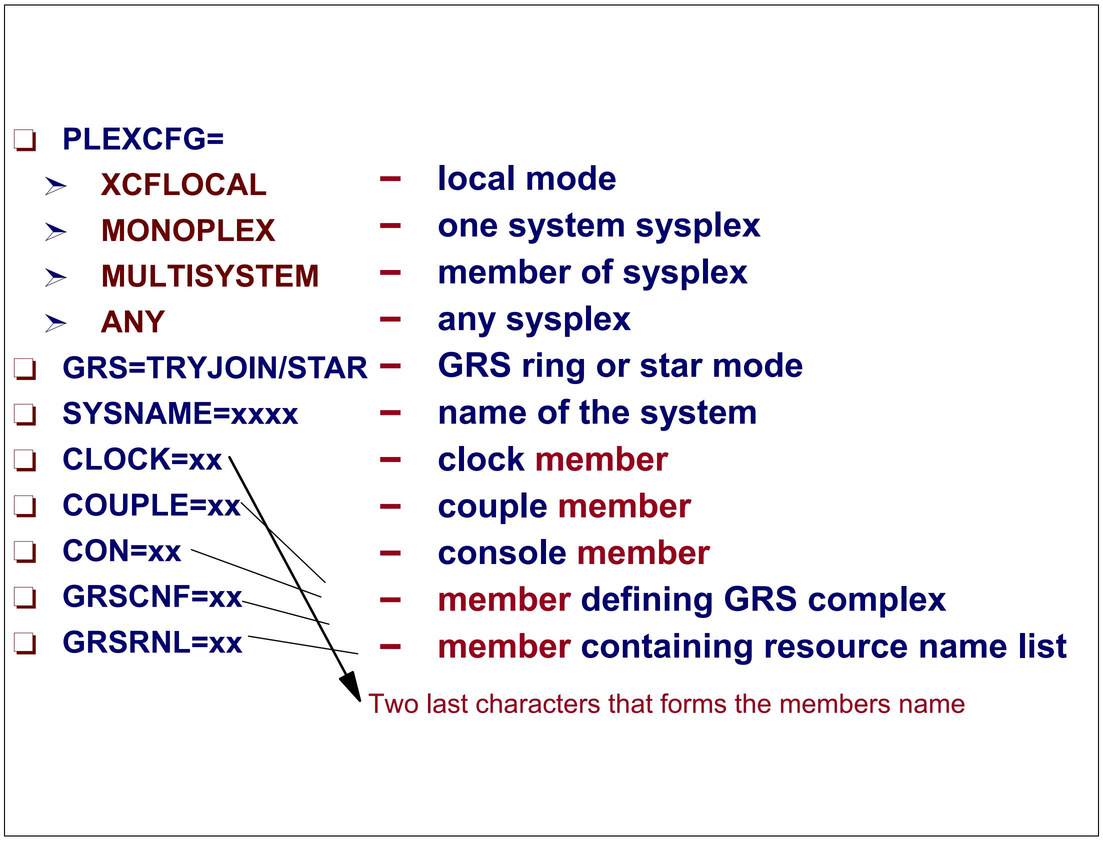
Tools and wizards for Parallel Sysplex customization

There are several tools, wizards, and tech papers linked from the IBM Parallel Sysplex URL:

`http://www-1.ibm.com/servers/eserver/zseries/pso/`

- z/OS Parallel Sysplex Customization Wizard
- Coupling Facility Structure Sizer Tool
- IBM Health Checker for z/OS and sysplex

1.47 IEASYSxx PARMLIB definitions for sysplex



- PLEXCFG=
  - XCFLOCAL – local mode
  - MONOPLEX – one system sysplex
  - MULTISYSTEM – member of sysplex
  - ANY – any sysplex
- GRS=TRYJOIN/STAR – GRS ring or star mode
- SYSNAME=xxxx – name of the system
- CLOCK=xx – clock member
- COUPLE=xx – couple member
- CON=xx – console member
- GRSCNF=xx – member defining GRS complex
- GRSRNL=xx – member containing resource name list

Two last characters that forms the members name

Figure 1-49 IEASYSxx PARMLIB definitions for sysplex

IEASYSxx PARMLIB definitions for sysplex

The values you specify in the SYS1.PARMLIB data set largely control the characteristics of z/OS systems in a sysplex. Many of the values that represent the fundamental decisions you make about the systems in your sysplex are in SYS1.PARMLIB.

IEASYSxx is the system parameter list that holds values that will be active in the z/OS just after the initialization (IPL) of z/OS. However, all those parameters may dynamically altered along the running of the system. Also in IEASYSxx you point to other PARMLIB members and those you need to consider, when setting up a z/OS system to run in a sysplex, are shown in Figure 1-49 and are discussed in the following pages.

SYSNAME parameter

You can specify the z/OS system name on the SYSNAME parameter in the IEASYSxx PARMLIB member. That name can be overridden by a SYSNAME value specified either in the IEASYMxx PARMLIB member or in response to the IEA101A Specify System Parameters message at console during IPL process.

You can specify system symbols in almost all parameter values in IEASYSxx. Symbols are variables that depending on the z/OS system may assume certain values. These symbols allow a same SYS1.PARMLIB be shared between several z/OSs. The IEASYMxx PARMLIB member provides a single place to define system symbols for all z/OSs in a sysplex.

1.48 IEASYSxx PLEXCFG parameter

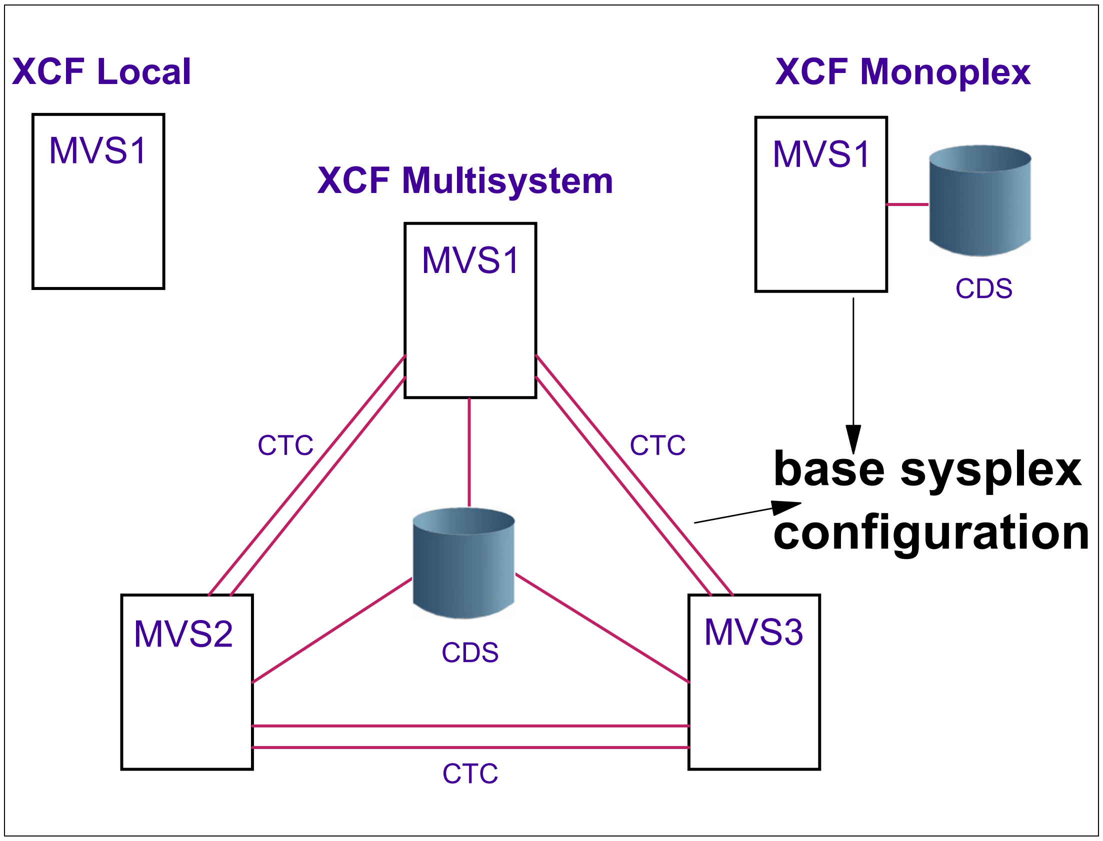


Figure 1-50 IEASYSxx PLEXCFG parameter

IEASYSxx PLEXCFG parameter

The PLEXCFG IEASYSxx parmlib member parameter restricts the type of sysplex configuration into which the system is allowed to IPL. The option for a Parallel Sysplex is:

MULTISYSTEM PLEXCFG=MULTISYSTEM indicates that this IPLing z/OS is a part of a sysplex consisting of one or more z/OS systems that reside on one or more CPCs. The sysplex couple data set (CDS) must be shared by all systems.

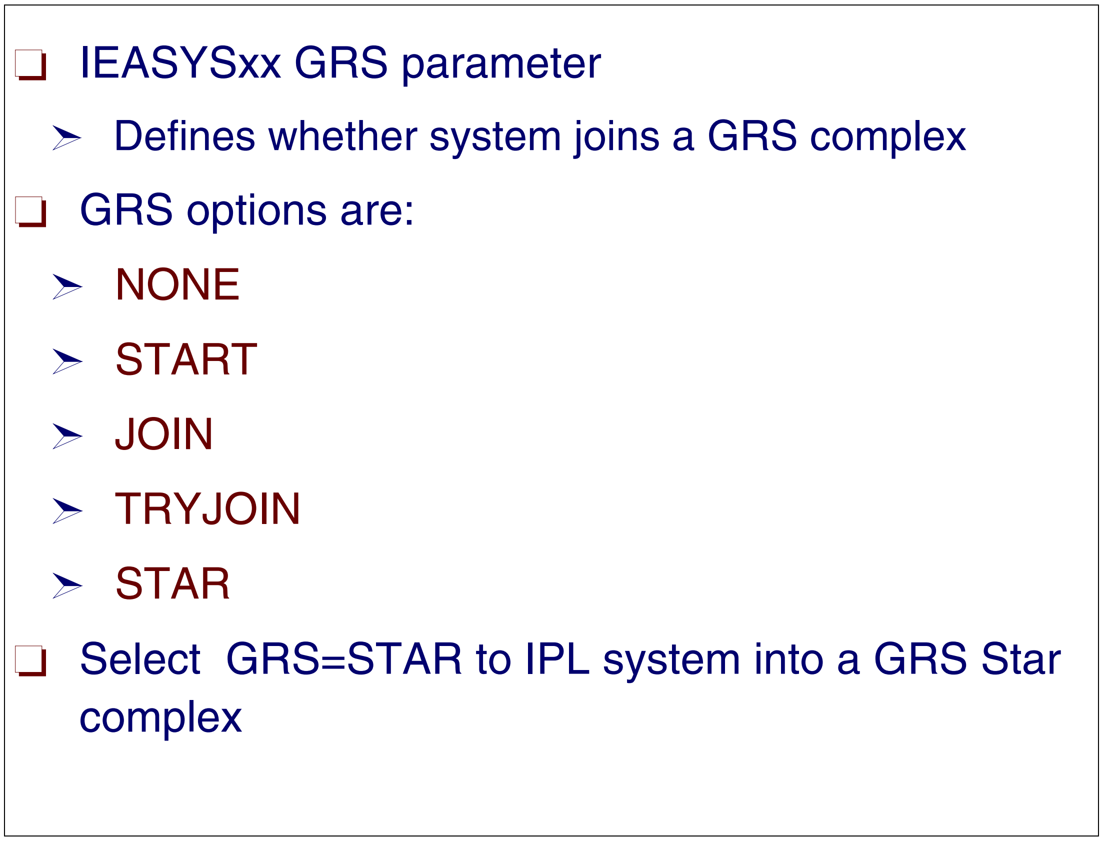
You must specify a COUPLExx parmlib member that identifies the same sysplex couple data sets for all systems in the sysplex (on the COUPLE statement) and signaling paths, if applicable, between systems (on the PATHIN and PATHOUT statements). You must also specify in the CLOCKxx parmlib member whether you are using a Sysplex Timer that is real (ETRMODE=YES) or simulated (SIMETRID specification).

Use MULTISYSTEM when you plan to IPL two or more MVS systems into a multisystem sysplex and exploit full XCF coupling services. GRS=NONE is not valid with PLEXCFG=MULTISYSTEM.

The options available are as follows:

```
PLEXCFG={XCFLOCAL   }
        {MONOPLEX   }
        {MULTISYSTEM}
        {ANY        }
```

1.49 IEASYSxx GRS parameter



- IEASYSxx GRS parameter
  - Defines whether system joins a GRS complex
- GRS options are:
  - NONE
  - START
  - JOIN
  - TRYJOIN
  - STAR
- Select GRS=STAR to IPL system into a GRS Star complex

Figure 1-51 IEASYSxx GRS parameter

IEASYSxx GRS parameter

The GRS IEASYSxx parmlib member parameter indicates whether the IPLing z/OS is to join a global ENQ resource serialization complex.

In a multisystem sysplex, every system in the sysplex must be in the same global resource serialization (GRS) complex. This allows global serialization of resources in the sysplex. To initialize each z/OS system in the multisystem sysplex, you must use the GRS component of z/OS.

GRS options

Every z/OS a sysplex is a member of the same GRS complex. GRS is required in a sysplex because components and products that use sysplex services need to access a sysplex-wide serialization mechanism. You can set up either of the following types of complex for global resource serialization:

- GRS=STAR

  In a GRS star complex, all of the z/OS systems (GRS) must be in the same Parallel Sysplex and connected to a CF lock structure in a *star* configuration via CF links.

  Star is the recommended configuration.

  If PLEXCFG=MULTISYSTEM the system starts or joins an existing sysplex and a global resource serialization star complex. XCF coupling services are used in the sysplex and in the complex.

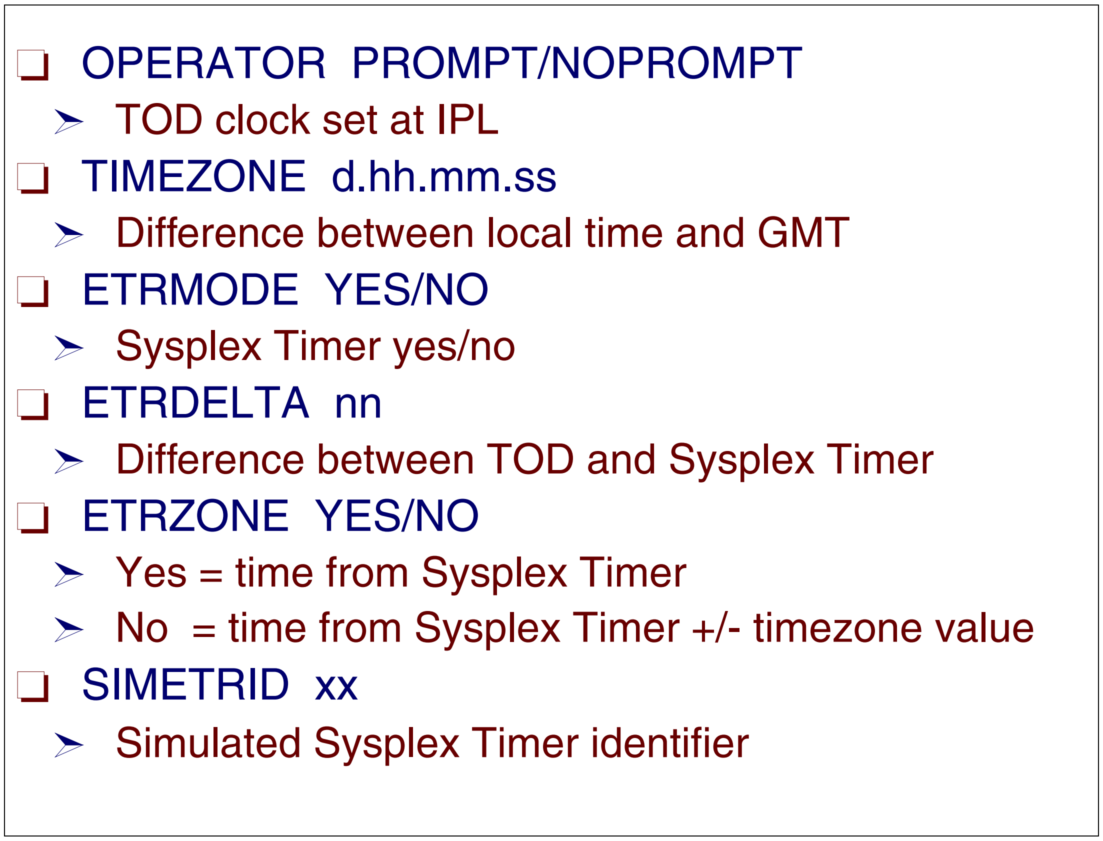
- GRSRNLxx

  This parameter specifies the suffix of the resource name lists (located in SYS1.PARMLIB) to be used to control the scope of the ENQ serialization of resources in the sysplex. This list describes if an ENW has a global or a local scope.

  Use GRSRNL=EXCLUDE that indicates that all global enqueues are to be treated as local enqueues and you plan to use one software product to implement the ENW serialization.

All the other options of the GRS parameter are out of date and because of that, they are not described here.

1.50 CLOCKxx parmlib member



- OPERATOR PROMPT/NOPROMPT
  - TOD clock set at IPL
- TIMEZONE d.hh.mm.ss
  - Difference between local time and GMT
- ETRMODE YES/NO
  - Sysplex Timer yes/no
- ETRDELTA nn
  - Difference between TOD and Sysplex Timer
- ETRZONE YES/NO
  - Yes = time from Sysplex Timer
  - No = time from Sysplex Timer +/- timezone value
- SIMETRID xx
  - Simulated Sysplex Timer identifier

Figure 1-52 CLOCKxx parmlib member

CLOCKxx parmlib member

The CLOCK=xx parameter in the IEASYSxx PARMLIB member specifies the suffix of the CLOCKxx PARMLIB member used during system IPL. CLOCKxx indicates how the time-of-day (TOD) is to be set on the IPLing z/OS. TOD synchronizations is a must because as it operates, z/OS obtains time stamps from the TOD clock and uses these time stamps to:

- Identify the sequence of events in the system
- Determine the duration of an activity
- Record the time on online reports or printed output
- Record the time in online logs used for recovery purposes

There are two techniques to guarantee this TOD sync. One through Sysplex Timer and other through Server Time Protocol.

The CLOCKxx defines which technique is used and also informs things like Daylight Saving and TIMEZONE, that is the deviation from Greewhich.

Using CLOCKxx parmlib member

CLOCKKxx performs the following functions:

- Prompts the operator to initialize the time of day (TOD) clock during system initialization.
- Specifies the difference between the local time and Coordinated Universal Time (UTC).

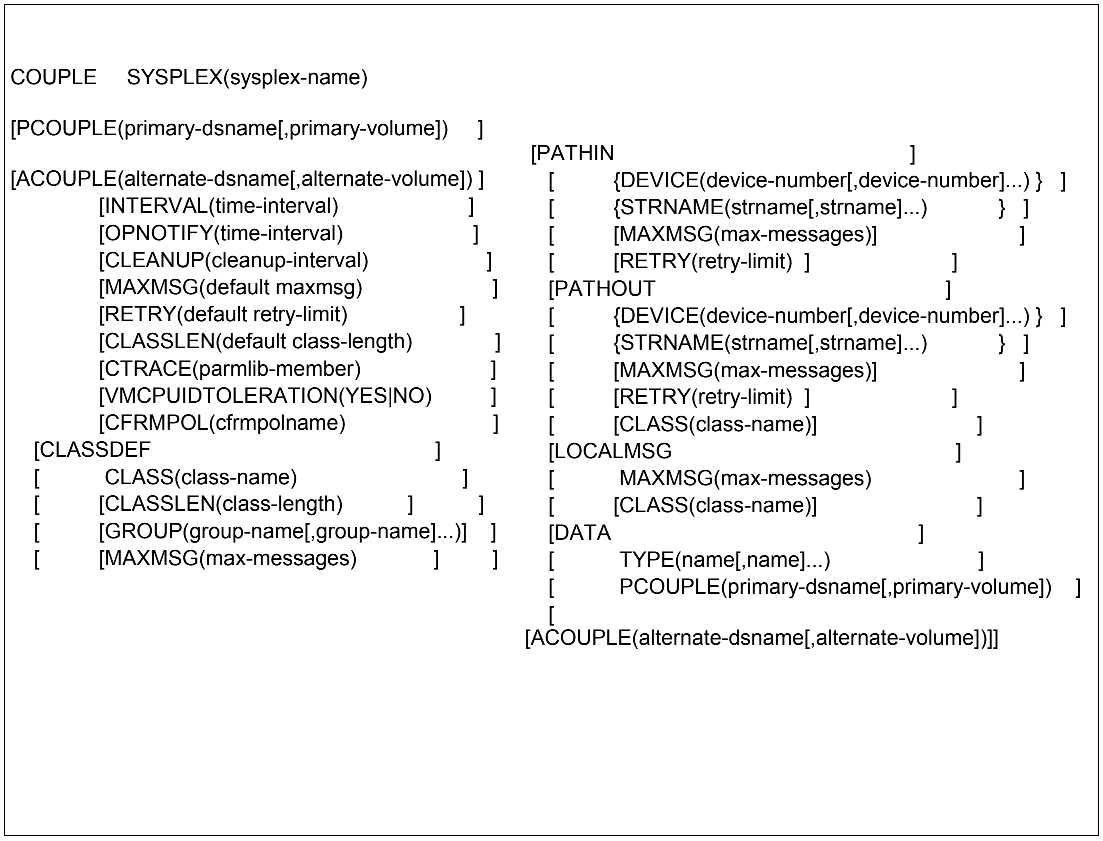
- Controls the utilization of the IBM Sysplex Timer (9037), which is an external time reference (ETR). Having all systems in your complex attached and synchronized to a Sysplex Timer ensures accurate sequencing and serialization of events.
- Provides the means of specifying that the Server time Protocol (STP)
- architecture is to be used in the sysplex. STP defines the method by
- which multiple servers maintain time synchronization.

The CLOCKxx member for a system that is a member of a multisystem sysplex must contain a specification of ETRMODE YES, STPMODE YES, or both. The system then uses the Sysplex Timer or STP to synchronize itself with the other members of the sysplex. The system uses a synchronized time stamp to provide appropriate sequencing and serialization of events within the sysplex.

Note: If all MVS images in the sysplex will run in LPARs or under VM on a single physical processor, you can specify SIMETRID instead of ETRMODE YES or STPMODE YES.

For more information about CLOCKxx and the Sysplex Timer, see *z/OS MVS Setting Up a Sysplex,* SA22-7625.

1.51 COUPLExx PARMLIB member



```
COUPLE    SYSPLEX(sysplex-name)

[PCOUPLE(primary-dsname[,primary-volume])    ]

[ACOUPLE(alternate-dsname[,alternate-volume]) ]
          [INTERVAL(time-interval)              ]
          [OPNOTIFY(time-interval)              ]
          [CLEANUP(cleanup-interval)            ]
          [MAXMSG(default maxmsg)               ]
          [RETRY(default retry-limit)           ]
          [CLASSLEN(default class-length)       ]
          [CTRACE(parmlib-member)               ]
          [VMCPUIDTOLERATION(YES|NO)            ]
          [CFRMPOL(cfrmpolname)                 ]
  [CLASSDEF                                ]
  [        CLASS(class-name)                ]
  [       [CLASSLEN(class-length)       ]   ]
  [       [GROUP(group-name[,group-name]...)]  ]
  [       [MAXMSG(max-messages)         ]   ]

[PATHIN                                  ]
  [       {DEVICE(device-number[,device-number]...) }  ]
  [       {STRNAME(strname[,strname]...)          }  ]
  [       [MAXMSG(max-messages)]                  ]
  [       [RETRY(retry-limit) ]                ]
  [PATHOUT                                ]
  [       {DEVICE(device-number[,device-number]...) }  ]
  [       {STRNAME(strname[,strname]...)          }  ]
  [       [MAXMSG(max-messages)]                  ]
  [       [RETRY(retry-limit) ]                ]
  [       [CLASS(class-name)]                    ]
  [LOCALMSG                               ]
  [        MAXMSG(max-messages)                   ]
  [       [CLASS(class-name)]                    ]
  [DATA                                 ]
  [        TYPE(name[,name]...)                  ]
  [        PCOUPLE(primary-dsname[,primary-volume])   ]
  [
[ACOUPLE(alternate-dsname[,alternate-volume])]]
```

Figure 1-53 COUPLExx PARMLIB member

COUPLExx PARMLIB member

The COUPLE=xx parameter in the IEASYSxx PARMLIB member specifies the COUPLExx PARMLIB member to be used. It has options to be activated after the z/OS IPL related to XCF. Just browsing the COUPLExx keywords:

SYSPLEX	SYSPLEX contains the name of the sysplex. The sysplex name allows a system to become part of the named sysplex. Specify the same sysplex-name for each z/OS system in the sysplex. Sysplex-name must match the sysplex name specified on the CDSs when they were formatted. The sysplex name is also the substitution text for the &SYSPLEX system symbol.
PCOUPLE	PCOUPLE contains the names of the primary and sysplex CDSs.
ACOUPLE	ACOUPLE contains the names of the alternate sysplex CDSs.
INTERVAL	INTERVAL can contain values that reflect recovery-related decisions for the sysplex. INTERVAL specifies the failure detection interval at which XCF on another system is to initiate system failure processing for this system because XCF on this system has not updated its status within the specified time.
OPNOTIFY	OPNOTIFY can contain values that reflect recovery-related decisions for the sysplex. OPNOTIFY specifies the amount of elapsed time at which XCF on another system is to notify the operator that this system has not updated

its status (heart beat). This value must be greater than or equal to the value specified on the INTERVAL keyword.

CLEANUP CLEANUP can contain values that reflect recovery-related decisions for the sysplex. CLEANUP specifies how many seconds the system waits between notifying members that this system is terminating and loading a non-restartable wait state. This is the amount of time members of the sysplex have to perform cleanup processing.

PATHIN/PATHOUT

PATHIN and PATHOUT describe the XCF signaling paths for inbound/outbound message traffic. More than one PATHIN/PATHOUT statement can be specified. The PATHIN/PATHOUT statement is not required for a single-system sysplex.

- DEVICE specifies the device number(s) of a signaling path used to receive/send messages sent from another system in the sysplex.
- STRNAME specifies the name of one or more CF list structures that are to be used to establish XCF signaling paths.

Note: Either the STRNAME keyword or the DEVICE keyword is required.

```
 COUPLE SYSPLEX(&SYSPLEX.)
       PCOUPLE(SYS1.XCF.CDS02)
       ACOUPLE(SYS1.XCF.CDS03)
       INTERVAL(85)
       OPNOTIFY(95)
       CLEANUP(30)
    /* MAXMSG(500)    */
       RETRY(10)
    /* CLASSLEN(1024) */

  /* DEFINITIONS FOR CFRM POLICY */
   DATA TYPE(CFRM)
       PCOUPLE(SYS1.XCF.CFRM04)
       ACOUPLE(SYS1.XCF.CFRM05)
........
/* LOCAL XCF MESSAGE TRAFFIC */
  LOCALMSG MAXMSG(512) CLASS(DEFAULT)

   PATHIN  DEVICE(4EE0,4F00,4F10,4F20)
   PATHOUT DEVICE(5EE0,5F00,5F10,5F20)

   PATHOUT STRNAME(IXC_DEFAULT_1,IXC_DEFAULT_2)
   PATHIN  STRNAME(IXC_DEFAULT_1,IXC_DEFAULT_2)

   CLASSDEF CLASS(BIG) CLASSLEN(62464) MAXMSG(4096) GROUP(UNDESIG)

   PATHIN  STRNAME(IXC_DEFAULT_3)
   PATHOUT STRNAME(IXC_DEFAULT_3) CLASS(BIG) MAXMSG(4096)

   PATHIN  DEVICE(4EE9,4F09,4F19,4F29)

   PATHOUT DEVICE(5EE9,5F09,5F19,5F29) CLASS(BIG) MAXMSG(4096)
........
```

Figure 1-54 COUPLExx member sample

1.52 Consoles in a sysplex

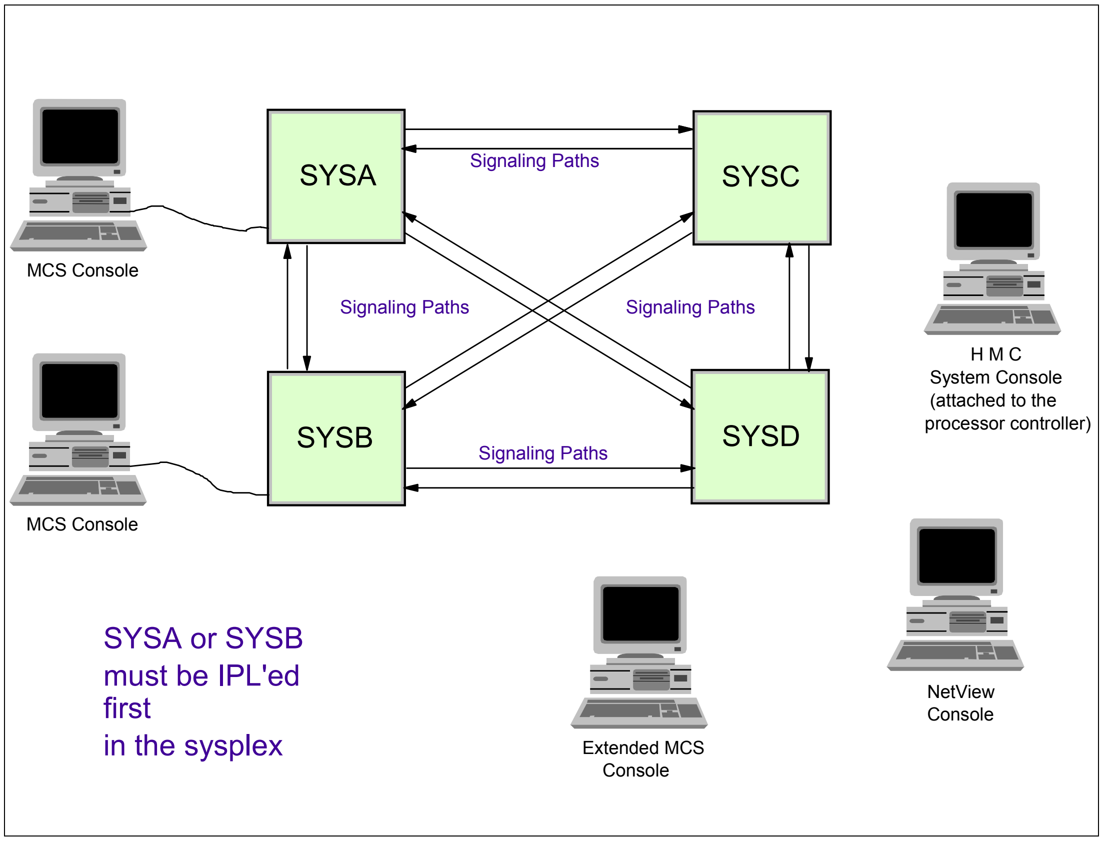


Figure 1-55 Consoles in a sysplex

Consoles in a sysplex

Traditionally, operators on a z/OS image receive messages from programs and enter commands to programs through consoles. These programs can be a z/OS component, or a subsystem or a customer application. Some of the messages imply that the operator should enter a reply. The traffic of messages and commands is managed by multisystem console support (MCS), a z/OS component. With MCS, a sysplex comprised of many z/OS images can be operated from a single console, giving the operator a single point of control for all images (as a cockpit in an airplane). To get more os consoles concepts, please refer to chapter 5 in this volume. In a sysplex, MCS consoles can:

- Be physically attached to any system, but just one at time
- Receive messages from any system in the sysplex, the messages are sent from one z/OS to other z/OS through XCF services.
- Route commands to be executed in any system in the sysplex, the commands are sent from one z/OS to other z/OS through XCF services.

Therefore, the following considerations apply when defining MCS consoles in this environment:

- There is no requirement that each system have consoles physically attached
- A sysplex, which can be up to 32 systems, can be operated from a single console (not recommended)
- One of the consoles must have master command authority, to issue any command
- All the commands before being executed are granted access (or not) by RACF or any other product in charge of the system security.

There are four types of operator consoles in a sysplex: MCS consoles, SMCS consoles, extended MCS consoles (EMCS), and integrated (system) consoles.

MCS consoles

MCS consoles are display devices that are attached to a z/OS system and provide the basic communication between operator and z/OS. MCS consoles must be locally channel-attached to non-SNA 3x74 or IBM 2074 control units; there is no MCS console support for any SNA-attached devices. You can define a maximum of 99 MCS consoles, including any subsystem allocatable consoles for a z/OS system. In a Parallel Sysplex, the limit is also 99 consoles for the entire Parallel Sysplex, which means that you may have to consider this in your configuration planning. One possible way to alleviate this restriction is through the use of extended MCS consoles

SMCS consoles

SMCS consoles are display devices connected to a SNA network through Secure Way Communication Server. The z/OS operator must logon his/her userid and password in order to access the z/OS.

Extended MCS consoles

Extended MCS consoles are defined and activated by authorized programs acting as operators. An extended MCS console is actually a program that acts as a console. It is used to issue z/OS commands and to receive command responses, unsolicited message traffic, and the hardcopy message set. There are two ways to use extended MCS consoles:

- Interactively, through IBM products that have support such as: TSO/E, SDSF, and NetView®
- Through user-written application programs

Generally speaking, an extended MCS console is used for almost any of the functions that are performed from an MCS console. It can also be controlled in a manner that is similar to an MCS console.

System (or integrated) consoles

This term refers to the interface provided by the Hardware Management Console (HMC) on an IBM System z servers. It is referred to as SYSCONS and does not have a device number. There are three system functions that may use this interface:

- Nucleus Initialization Program (NIP)
- Disabled Console Communication Facility (DCCF)
- Multiple Console Support (MCS)

The system console is automatically defined during z/OS initialization.

1.53 Multisystem consoles in a sysplex

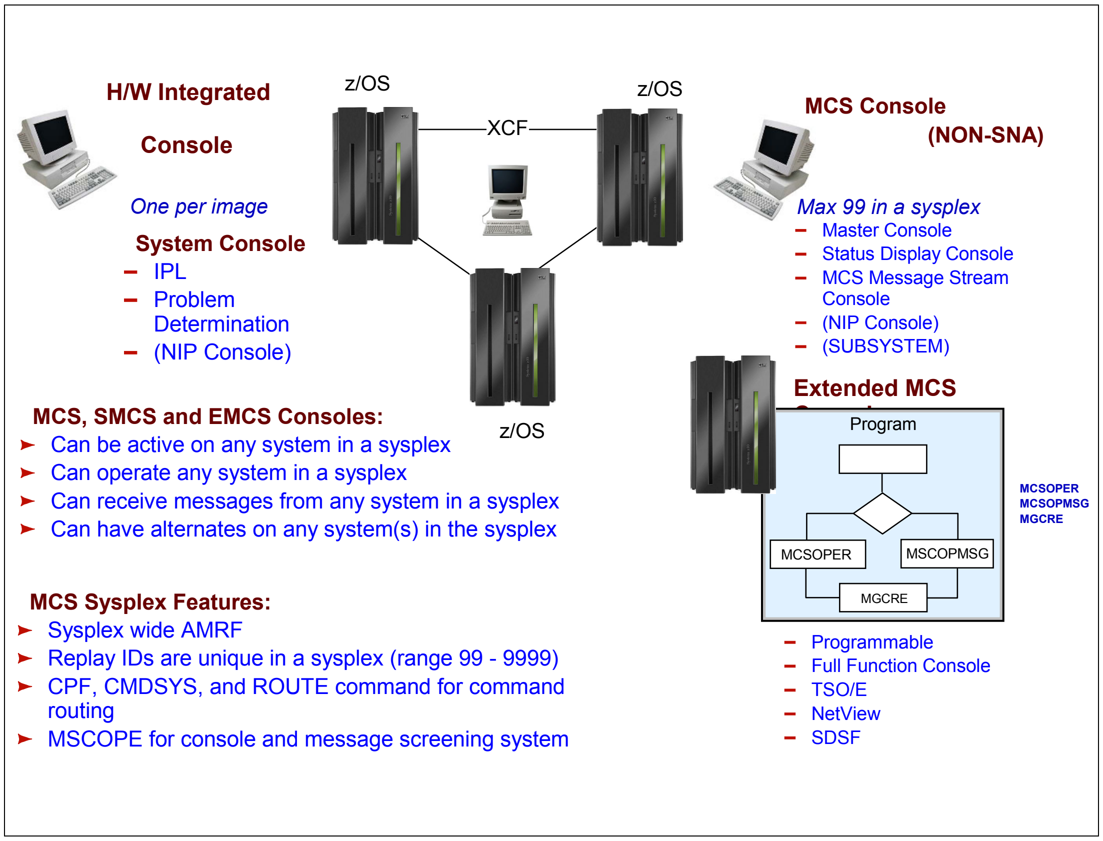

H/W Integrated
Console
One per image
System Console
– IPL
– Problem Determination
– (NIP Console)
z/OS
XCF
z/OS
z/OS
MCS Console (NON-SNA)
Max 99 in a sysplex
– Master Console
– Status Display Console
– MCS Message Stream Console
– (NIP Console)
– (SUBSYSTEM)
Extended MCS
Program
MCSOPER
MSCOPMSG
MGCRE
MCSOPER
MCSOPMSG
MGCRE
– Programmable
– Full Function Console
– TSO/E
– NetView
– SDSF
MCS, SMCS and EMCS Consoles:
➤ Can be active on any system in a sysplex
➤ Can operate any system in a sysplex
➤ Can receive messages from any system in a sysplex
➤ Can have alternates on any system(s) in the sysplex
MCS Sysplex Features:
➤ Sysplex wide AMRF
➤ Replay IDs are unique in a sysplex (range 99 - 9999)
➤ CPF, CMDSYS, and ROUTE command for command routing
➤ MSCOPE for console and message screening system

Figure 1-56 Multisystem consoles in a sysplex

Multisystem consoles in a sysplex

In a Parallel Sysplex implementation, there is no requirement that you have an MCS console on every system in the Parallel Sysplex. Using command and message routing capabilities, from one MCS, SMCS or extended MCS console, it is possible to control multiple z/OS in the Parallel Sysplex. Although MCS consoles are not required on all systems, you should plan the configuration carefully to ensure that there is an adequate number to handle the message traffic and to provide a valid configuration, regardless of the number of z/OSs in the Parallel Sysplex at a time.

If there is neither an MCS console nor an integrated system console on a system, that system probably cannot be the first to be IPLed into a sysplex; or a wait state would result. Alternate consoles must also be considered across the entire sysplex, especially for the sysplex master console. You should plan the console configuration so that there is always an alternate to the sysplex master console available at all times. If you do not do so, unnecessary operator action is required, and messages may be lost or sent to the hardcopy log, that is a log in the JES2 spool data set of all console traffic to be viewed by the auditors, if necessary.

A given computing operations environment can have a variety of different types of operators and different types of consoles. This chapter focuses primarily on the z/OS console operator's tasks, and how the z/OS operator's job might be different in a sysplex.

A z/OS console operator must start, run, and stop the z/OS operating system. That involves controlling z/OS system software, and the hardware it runs on, including processors, channel paths, and I/O devices. The operator might deal with messages and problems from many sources including z/OS, JES, DFSMS/z/OS, and other subsystems and products such as CICS, IMS, and VTAM, to name just a few. Some of the major tasks of operating the z/OS system include:

- Starting the system
- Controlling the system
- Controlling jobs
- Controlling system information recording
- Responding to failures
- Changing the configuration
- Providing problem determination data
- Quiescing the system
- Stopping the system

An operator in a single system environment is using multiple consoles to interact with the system.

In the sysplex environment the major tasks of operating the systems do not change very much. The potential exists that the number of consoles increase and the operator tasks become more repetitive and complex. This is caused by the fact that a sysplex has multiple z/OS images, each image with a set of subsystems and applications. Another effect is that this results in a flood of messages making system monitoring difficult.

The goal is to reduce the number of consoles that operators have to deal with. Suppress and reduce messages with automation software where possible. Use automation scripts to detect and react on standard malfunctions and system replies. Use graphical user interfaces (GUIs) to simplify the environment if possible.

Figure 1-56 on page 89 shows the three types of consoles:

Consoles in a sysplex

In a sysplex, a console can be active on any system in a sysplex and can provide sysplex-wide control. MCS uses XCF services for command and message transportation between systems and thus provides a single system image for the operators. MCS multisystem support features:

- Sysplex-wide action message retention facility (ARMF)
- Sysplex-wide unique reply IDs
- Sysplex-wide command routing through:
  - ROUTE operator command
  - Command prefix facility (CPF)
  - CMDSYS setting for a console (through the CONSOLE statement in the CONSOLxx PARMLIB member or the CONTROL V,CMDSYS= operator command)

1.54 Sysplex operation and management

```
ROUTE sysname,text

RO sysname,text

RO [T=nnn,]{ *ALL                        }[,L={a      }]
            {sysgrpname                  }     {cc     }
            {*OTHER                      }     {cca    }
            {(sysname,[sysgrpname...])}       {name  }
                                               {namea}
```

Figure 1-57 Sysplex operation and management

Sysplex operation and management

The following sections explain how the goals of single system image, single point of control, and minimal human intervention help to simplify the operator's job in a sysplex.

Single system image and single point of control

Single system image allows the operator, for certain tasks, to interact with multiple images of a product as though they were one image. For example, the operator can issue a single command to all z/OS systems in the sysplex, instead of repeating the command for each system, using the z/OS ROUTE command.

Single point of control allows the operator to interact with a suite of products from a single workstation, without in some cases knowing which products are performing which functions.

Single point of control does not necessarily imply that you would have one single workstation from which all tasks by all people in the computing center would be done. But you could set things up so that each operator can accomplish a set of tasks from a single workstation, thereby reducing the number of consoles the operator has to deal with. If that workstation is also a graphical workstation where tasks can be performed by using a mouse to select ("click on") choices or functions to be performed, you have also reduced the complexity of performing the tasks.

One of the assumptions of single point of control is that you can receive messages from all systems on a single system. This does not happen automatically. We recommend that you set up most of your z/OS MCS consoles to receive messages from all systems, for the following reasons:

- You might need messages from multiple systems for diagnosing problems.
- Consolidating messages reduces the number of consoles you need.

Entering operator commands

z/OS operator commands can provide key data for problem analysis. The commands can be entered from the following consoles:

- The multiple console support (MCS) console
- The system console, a z/OS console emulated in the HMC
- The Hardware Management Console to interface with hardware
- The NetView console, an example of an EMCS console

The commands can also be issued by NetView automation CLISTs and by programs that use the extended MCS support.

Some commands have a sysplex scope, independent of any routing command or automation, while others can be routed to systems in the sysplex with the ROUTE command. For example, an operator can reply to message (WTOR) from any console in the sysplex, even if the console is not directly attached to the system that issued the message. Another example is certain forms of DISPLAY XCF and SETXCF commands.

If a problem management tool indicates a failure or an specific message, the operator can use DISPLAY commands that determine the detailed status of the failing system, job, application, device, system component, and so on. Based on the status, the operator has many commands to provide control and initiate actions; for example, the operator can enter commands to recover or isolate a failing resource.

ROUTE command

The operator can use the z/OS ROUTE command to direct a command to all systems in the sysplex (ROUTE *ALL), or a subset of the systems in the sysplex (ROUTE system_group_name). When issuing a command to multiple systems, the operator can receive a consolidated response to one console, rather than receiving responses from each system that must be located and correlated. RO is the short form of the command.

The syntax of the ROUTE command is shown in Figure 1-57 on page 91 and the meanings of the ROUTE parameters are as follows:

sysname	The system name (1 to 8 characters) that will receive and process the command
text	The system command and specific operands of the command being routed
T=	This value is optional and indicates the maximum number of seconds z/OS waits for responses from each system before aggregating the responses. T is *not* valid when you specify just *one* sysname.
ALL	Specifies that the command is to be routed to all systems in the sysplex.
OTHER	Sends the command to all systems in a sysplex except the system on which the command is entered.
sysgrpname	Routes the command to a subset of systems in the sysplex.
L	Is optional and specifies the display area, console, or both, to display the command responses.

1.55 Displaying CF information

```
D CF
IXL150I  13.58.49  DISPLAY CF 131
COUPLING FACILITY 002084.IBM.02.000000026A3A
                  PARTITION: 0D  CPCID: 00
                  CONTROL UNIT ID: FFF8
NAMED CF2
COUPLING FACILITY SPACE UTILIZATION
 ALLOCATED SPACE                 DUMP SPACE UTILIZATION
  STRUCTURES:      78848 K        STRUCTURE DUMP TABLES:        0 K
  DUMP SPACE:       2048 K                  TABLE COUNT:        0
 FREE SPACE:      902400 K       FREE DUMP SPACE:            2048 K
TOTAL SPACE:      983296 K      TOTAL DUMP SPACE:            2048 K
                               MAX REQUESTED DUMP SPACE:        0 K
   VOLATILE:         YES         STORAGE INCREMENT SIZE:      256 K
    CFLEVEL:          13
    CFCC RELEASE 13.00, SERVICE LEVEL 00.02
    BUILT ON 02/13/2004 AT 14:33:00

    CF REQUEST TIME ORDERING: REQUIRED AND ENABLED

COUPLING FACILITY SPACE CONFIGURATION
                          IN USE          FREE         TOTAL
CONTROL SPACE:           80896 K      902400 K      983296 K
NON-CONTROL SPACE:           0 K           0 K           0 K
```

Figure 1-58 Displaying CF information

Displaying CF information

The command has several optional parameters, as follows:

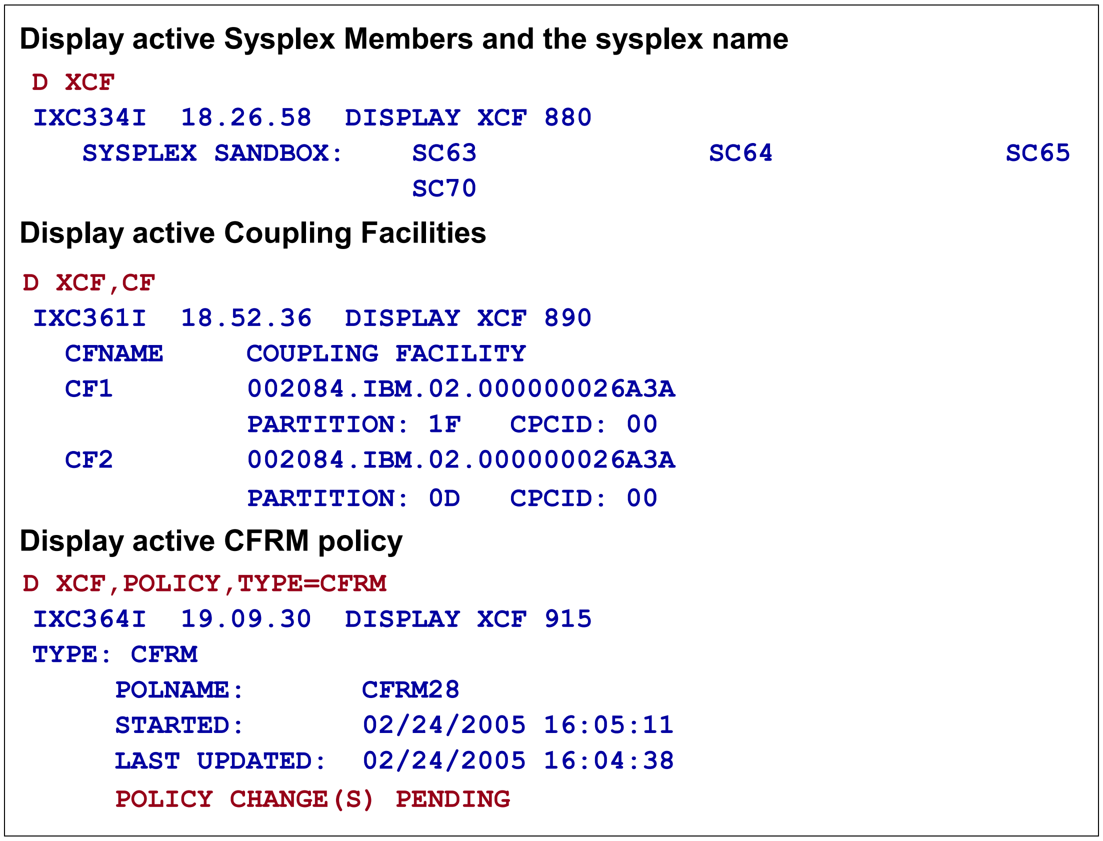
```
D CF[,CFNAME={(cfname[,cfname]...)]
```

Use the **DISPLAY CF** command to display storage and attachment information about coupling facilities attached to the system on which the command is processed.

The **DISPLAY CF** command has local scope—it displays the CF hardware definition such as model and serial number, partition number, and so forth. In addition you will see the local system connections such as channels and paths.

When specified without further parameters, as in Figure 1-58, the system displays information about all coupling facilities that are attached. The output of the command in Figure 1-58 is only a partial paste.

1.56 Display XCF information (1)



Display active Sysplex Members and the sysplex name

```
 D XCF
 IXC334I  18.26.58  DISPLAY XCF 880
    SYSPLEX SANDBOX:     SC63              SC64                SC65
                         SC70
```

Display active Coupling Facilities

```
D XCF,CF
 IXC361I  18.52.36  DISPLAY XCF 890
   CFNAME     COUPLING FACILITY
   CF1        002084.IBM.02.000000026A3A
              PARTITION: 1F   CPCID: 00
   CF2        002084.IBM.02.000000026A3A
              PARTITION: 0D   CPCID: 00
```

Display active CFRM policy

```
D XCF,POLICY,TYPE=CFRM
 IXC364I  19.09.30  DISPLAY XCF 915
 TYPE: CFRM
      POLNAME:       CFRM28
      STARTED:       02/24/2005 16:05:11
      LAST UPDATED:  02/24/2005 16:04:38
      POLICY CHANGE(S) PENDING
```

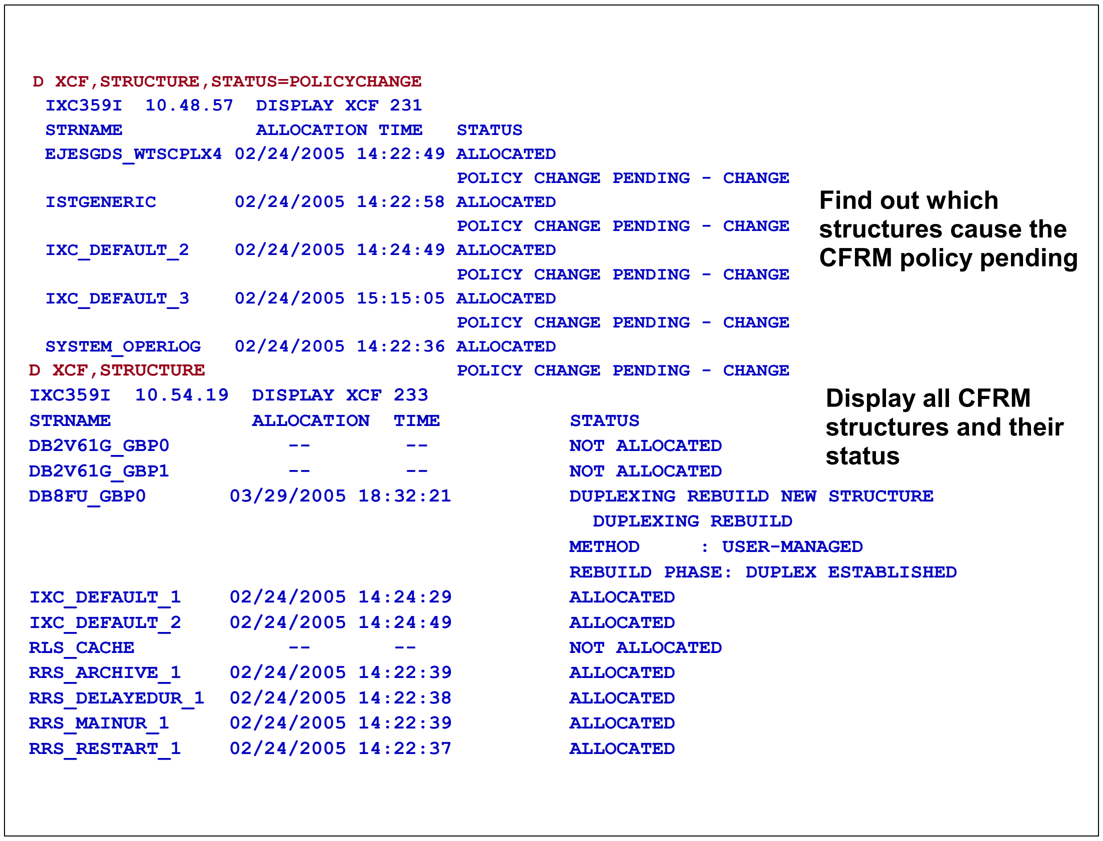
Figure 1-59 Display XCF information (1)

Display XCF information (1)

Use the `DISPLAY XCF` command to display cross-system coupling information in the sysplex, as follows:.

- `D XCF` - displays active sysplex member names SCxx in the sysplex *sandbox*.
- `D XCF,CF` - displays active coupling facilities defined in the CFRM policy.
- `D XCF,POLICY,TYPE=CFRM` - displays active CFRM policy CFRM28 with start and update information. In our example the policy is in POLICY CHANGE(S) PENDING status.

1.57 Display XCF information (2)



```
D XCF,STRUCTURE,STATUS=POLICYCHANGE
 IXC359I  10.48.57  DISPLAY XCF 231
 STRNAME           ALLOCATION TIME   STATUS
 EJESGDS_WTSCPLX4 02/24/2005 14:22:49 ALLOCATED
                                      POLICY CHANGE PENDING - CHANGE
 ISTGENERIC       02/24/2005 14:22:58 ALLOCATED
                                      POLICY CHANGE PENDING - CHANGE
 IXC_DEFAULT_2    02/24/2005 14:24:49 ALLOCATED
                                      POLICY CHANGE PENDING - CHANGE
 IXC_DEFAULT_3    02/24/2005 15:15:05 ALLOCATED
                                      POLICY CHANGE PENDING - CHANGE
 SYSTEM_OPERLOG   02/24/2005 14:22:36 ALLOCATED
D XCF,STRUCTURE                       POLICY CHANGE PENDING - CHANGE
IXC359I  10.54.19  DISPLAY XCF 233
STRNAME            ALLOCATION  TIME           STATUS
DB2V61G_GBP0          --         --           NOT ALLOCATED
DB2V61G_GBP1          --         --           NOT ALLOCATED
DB8FU_GBP0        03/29/2005 18:32:21         DUPLEXING REBUILD NEW STRUCTURE
                                                DUPLEXING REBUILD
                                              METHOD      : USER-MANAGED
                                              REBUILD PHASE: DUPLEX ESTABLISHED
IXC_DEFAULT_1     02/24/2005 14:24:29         ALLOCATED
IXC_DEFAULT_2     02/24/2005 14:24:49         ALLOCATED
RLS_CACHE             --         --           NOT ALLOCATED
RRS_ARCHIVE_1     02/24/2005 14:22:39         ALLOCATED
RRS_DELAYEDUR_1   02/24/2005 14:22:38         ALLOCATED
RRS_MAINUR_1      02/24/2005 14:22:39         ALLOCATED
RRS_RESTART_1     02/24/2005 14:22:37         ALLOCATED
```

Find out which structures cause the CFRM policy pending
Display all CFRM structures and their status

Figure 1-60 Display XCF information (2)

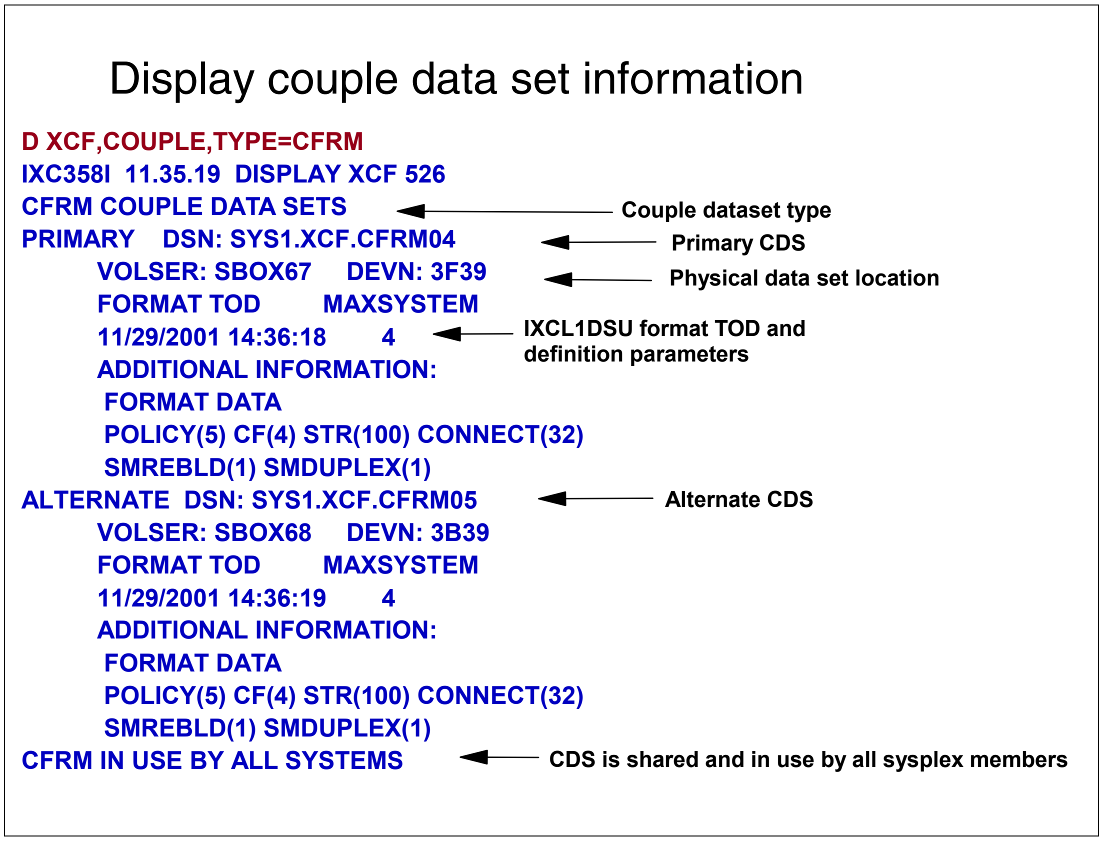
Display XCF information (2)

The commands shown in Figure 1-60are as follows:

- The activation of a coupling facility resource management policy has caused pending policy changes to some coupling facility structures. The changes are pending the deallocation of the structure in a coupling facility.
  - **`D XCF,STRUCTURE,STATUS=POLICYCHANGE`** - displays all structures in CFRM POLICY CHANGE PENDING state. A SETXCF START,REBUILD,STRNAME=strname command should resolve this status.
- The CFRM policy contains the maximum structure size and can contain a smaller initial size also. The initial structure size defined in the CFRM policy (or the maximum size if an initial size is not specified) is used as the attempted allocation size unless it is overridden by a structure size specified on the IXLCONN macro.

  The CFRM policy can also optionally designate a minimum structure size. The MINSIZE value specifies the minimum bound for the structure size on all allocation requests except those resulting from system-managed rebuild.
  - **`D XCF,STRUCTURE`** - displays all structures defined in the CFRM policy. Note that Figure 1-60 shows only a partial display.

1.58 Display XCF information (3)



Display couple data set information

```
D XCF,COUPLE,TYPE=CFRM
IXC358I  11.35.19  DISPLAY XCF 526
CFRM COUPLE DATA SETS
PRIMARY    DSN: SYS1.XCF.CFRM04
        VOLSER: SBOX67     DEVN: 3F39
        FORMAT TOD         MAXSYSTEM
        11/29/2001 14:36:18       4
        ADDITIONAL INFORMATION:
         FORMAT DATA
         POLICY(5) CF(4) STR(100) CONNECT(32)
         SMREBLD(1) SMDUPLEX(1)
ALTERNATE  DSN: SYS1.XCF.CFRM05
        VOLSER: SBOX68     DEVN: 3B39
        FORMAT TOD         MAXSYSTEM
        11/29/2001 14:36:19       4
        ADDITIONAL INFORMATION:
         FORMAT DATA
         POLICY(5) CF(4) STR(100) CONNECT(32)
         SMREBLD(1) SMDUPLEX(1)
CFRM IN USE BY ALL SYSTEMS
```

Couple dataset type
Primary CDS
Physical data set location
IXCL1DSU format TOD and definition parameters
Alternate CDS
CDS is shared and in use by all sysplex members

Figure 1-61 Display XCF information (3)

Display XCF information (3)

This command displays information about a CFRM couple data set (CDS) in use in the sysplex. The information includes:

- Physical attributes of the CFRM couple data set (name, volume serial number, device address, and time the data set was formatted)
- Maximum number of systems that the primary CFRM couple data set can support
- Names of the systems using the CFRM couple data set.

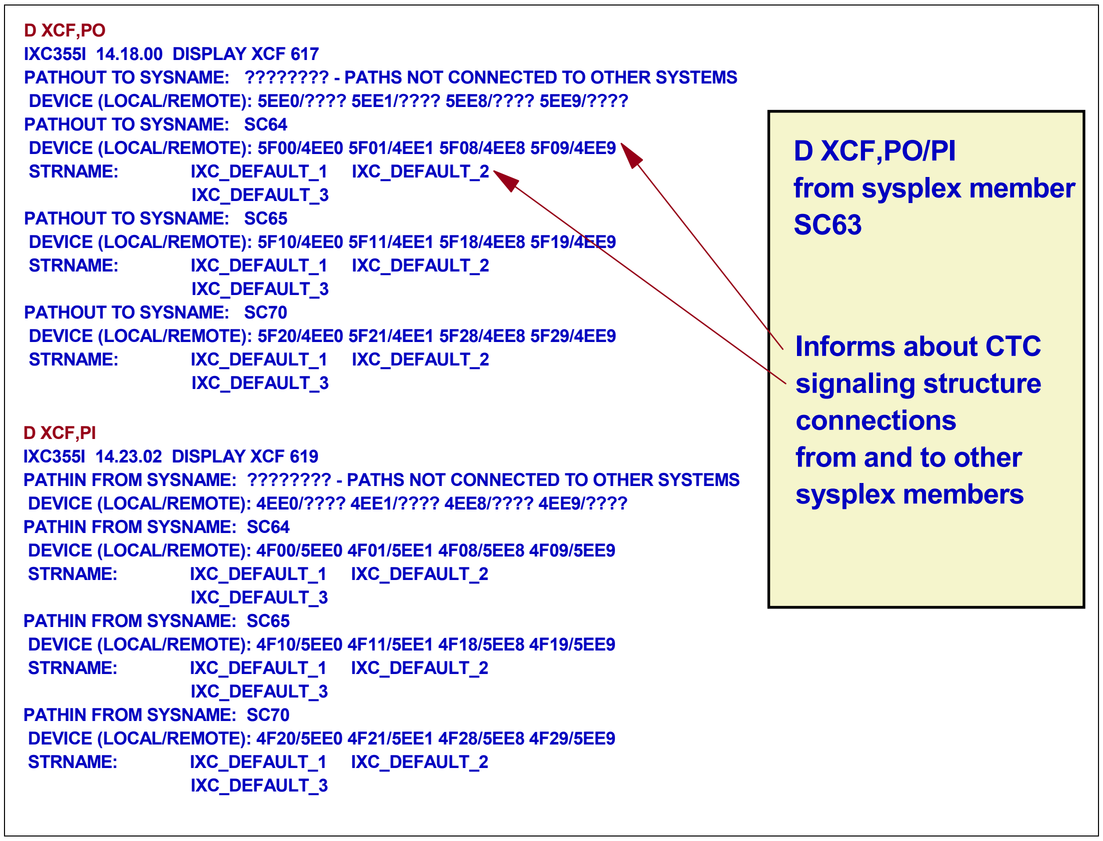
In Figure 1-61 the `D XCF,COUPLE,TYPE=CFRM` shows all relevant information about a specific CDS named CFRM.

Type=name

Here *name* specifies the name of the service using the CDS for which information is to be displayed. The name may be up to eight characters long. It may contain characters A-Z and 0-9 and the characters $, @, and # and must start with a letter . Supported names are:

- SYSPLEX for sysplex (XCF) types
- ARM for automatic restart management
- CFRM for coupling facility resource management
- SFM for sysplex failure management
- LOGR for the System Logger
- WLM for workload management

1.59 Display XCF signaling paths (1)



```
D XCF,PO
IXC355I  14.18.00  DISPLAY XCF 617
PATHOUT TO SYSNAME:   ???????? - PATHS NOT CONNECTED TO OTHER SYSTEMS
 DEVICE (LOCAL/REMOTE): 5EE0/???? 5EE1/???? 5EE8/???? 5EE9/????
PATHOUT TO SYSNAME:   SC64
 DEVICE (LOCAL/REMOTE): 5F00/4EE0 5F01/4EE1 5F08/4EE8 5F09/4EE9
 STRNAME:              IXC_DEFAULT_1    IXC_DEFAULT_2
                       IXC_DEFAULT_3
PATHOUT TO SYSNAME:   SC65
 DEVICE (LOCAL/REMOTE): 5F10/4EE0 5F11/4EE1 5F18/4EE8 5F19/4EE9
 STRNAME:              IXC_DEFAULT_1    IXC_DEFAULT_2
                       IXC_DEFAULT_3
PATHOUT TO SYSNAME:   SC70
 DEVICE (LOCAL/REMOTE): 5F20/4EE0 5F21/4EE1 5F28/4EE8 5F29/4EE9
 STRNAME:              IXC_DEFAULT_1    IXC_DEFAULT_2
                       IXC_DEFAULT_3

D XCF,PI
IXC355I  14.23.02  DISPLAY XCF 619
PATHIN FROM SYSNAME:  ???????? - PATHS NOT CONNECTED TO OTHER SYSTEMS
 DEVICE (LOCAL/REMOTE): 4EE0/???? 4EE1/???? 4EE8/???? 4EE9/????
PATHIN FROM SYSNAME:  SC64
 DEVICE (LOCAL/REMOTE): 4F00/5EE0 4F01/5EE1 4F08/5EE8 4F09/5EE9
 STRNAME:              IXC_DEFAULT_1    IXC_DEFAULT_2
                       IXC_DEFAULT_3
PATHIN FROM SYSNAME:  SC65
 DEVICE (LOCAL/REMOTE): 4F10/5EE0 4F11/5EE1 4F18/5EE8 4F19/5EE9
 STRNAME:              IXC_DEFAULT_1    IXC_DEFAULT_2
                       IXC_DEFAULT_3
PATHIN FROM SYSNAME:  SC70
 DEVICE (LOCAL/REMOTE): 4F20/5EE0 4F21/5EE1 4F28/5EE8 4F29/5EE9
 STRNAME:              IXC_DEFAULT_1    IXC_DEFAULT_2
                       IXC_DEFAULT_3
```

D XCF,PO/PI from sysplex member SC63
Informs about CTC signaling structure connections from and to other sysplex members

Figure 1-62 Display XCF signaling pathes (1)

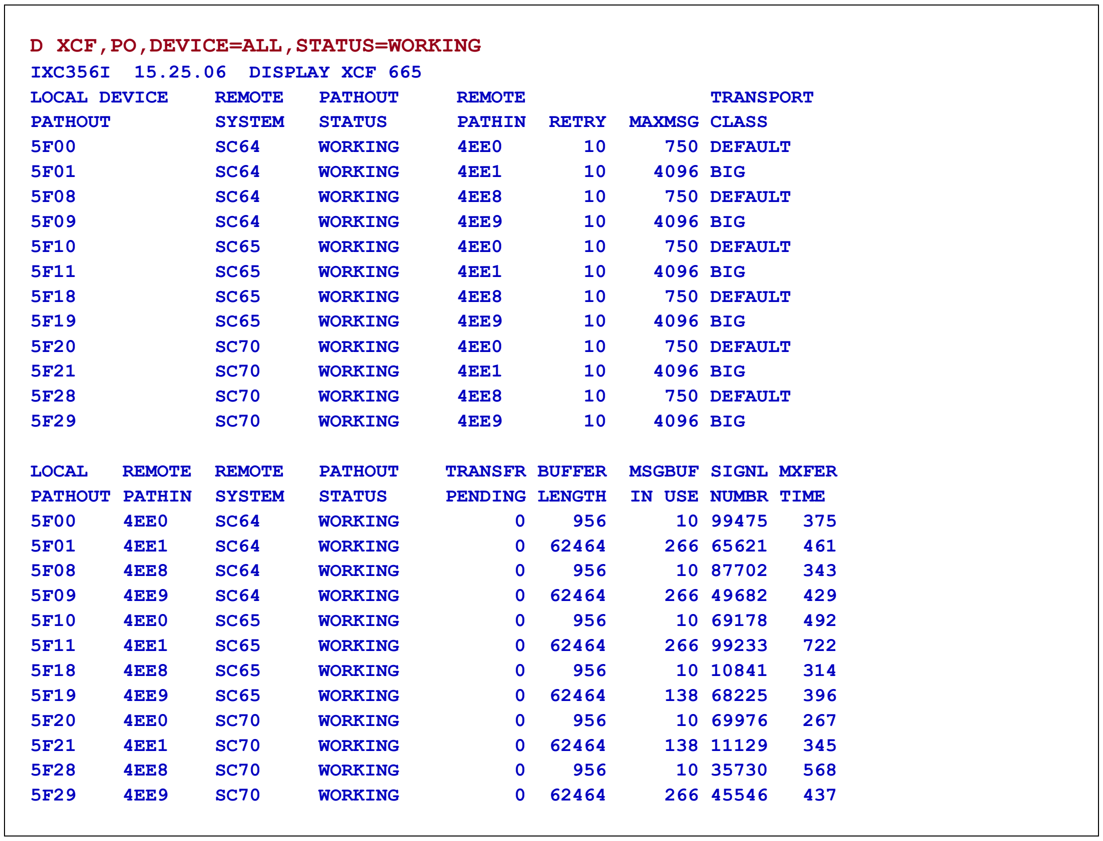
Display XCF signaling paths (1)

`D XCF,PI` or **`D XCF,PO`** is useful for seeing all outbound and inbound connections from a sysplex member. Remember that these connections are defined in the COUPLExx member. In our example, Figure 1-62, you see signaling list structures and CTC connections making up our XCF communication paths.

Message IXC355I displays the device number of one or more outbound signaling paths that XCF can use, and information about outbound XCF signaling paths to this system. The display provides information for only those devices and structures that are defined to the system where this command is executed. The path summary response identifies each outbound path and, if known, the system name and device address of its associated inbound path. If specified without further qualification, summary information about all outbound XCF signalling paths is displayed. Use of the **`DEVICE,STRNAME`** or **`CLASS`** keyword requests that detail information be displayed.

If there are no outbound paths to this system, the system displays message IXC356I.

1.60 Display XCF signaling paths (2)



```
D XCF,PO,DEVICE=ALL,STATUS=WORKING
IXC356I  15.25.06  DISPLAY XCF 665
LOCAL DEVICE     REMOTE    PATHOUT     REMOTE                 TRANSPORT
PATHOUT          SYSTEM    STATUS      PATHIN   RETRY  MAXMSG CLASS
5F00             SC64      WORKING     4EE0        10     750 DEFAULT
5F01             SC64      WORKING     4EE1        10    4096 BIG
5F08             SC64      WORKING     4EE8        10     750 DEFAULT
5F09             SC64      WORKING     4EE9        10    4096 BIG
5F10             SC65      WORKING     4EE0        10     750 DEFAULT
5F11             SC65      WORKING     4EE1        10    4096 BIG
5F18             SC65      WORKING     4EE8        10     750 DEFAULT
5F19             SC65      WORKING     4EE9        10    4096 BIG
5F20             SC70      WORKING     4EE0        10     750 DEFAULT
5F21             SC70      WORKING     4EE1        10    4096 BIG
5F28             SC70      WORKING     4EE8        10     750 DEFAULT
5F29             SC70      WORKING     4EE9        10    4096 BIG

LOCAL    REMOTE  REMOTE    PATHOUT    TRANSFR BUFFER  MSGBUF SIGNL MXFER
PATHOUT  PATHIN  SYSTEM    STATUS     PENDING LENGTH  IN USE NUMBR TIME
5F00     4EE0    SC64      WORKING          0    956      10 99475   375
5F01     4EE1    SC64      WORKING          0  62464     266 65621   461
5F08     4EE8    SC64      WORKING          0    956      10 87702   343
5F09     4EE9    SC64      WORKING          0  62464     266 49682   429
5F10     4EE0    SC65      WORKING          0    956      10 69178   492
5F11     4EE1    SC65      WORKING          0  62464     266 99233   722
5F18     4EE8    SC65      WORKING          0    956      10 10841   314
5F19     4EE9    SC65      WORKING          0  62464     138 68225   396
5F20     4EE0    SC70      WORKING          0    956      10 69976   267
5F21     4EE1    SC70      WORKING          0  62464     138 11129   345
5F28     4EE8    SC70      WORKING          0    956      10 35730   568
5F29     4EE9    SC70      WORKING          0  62464     266 45546   437
```

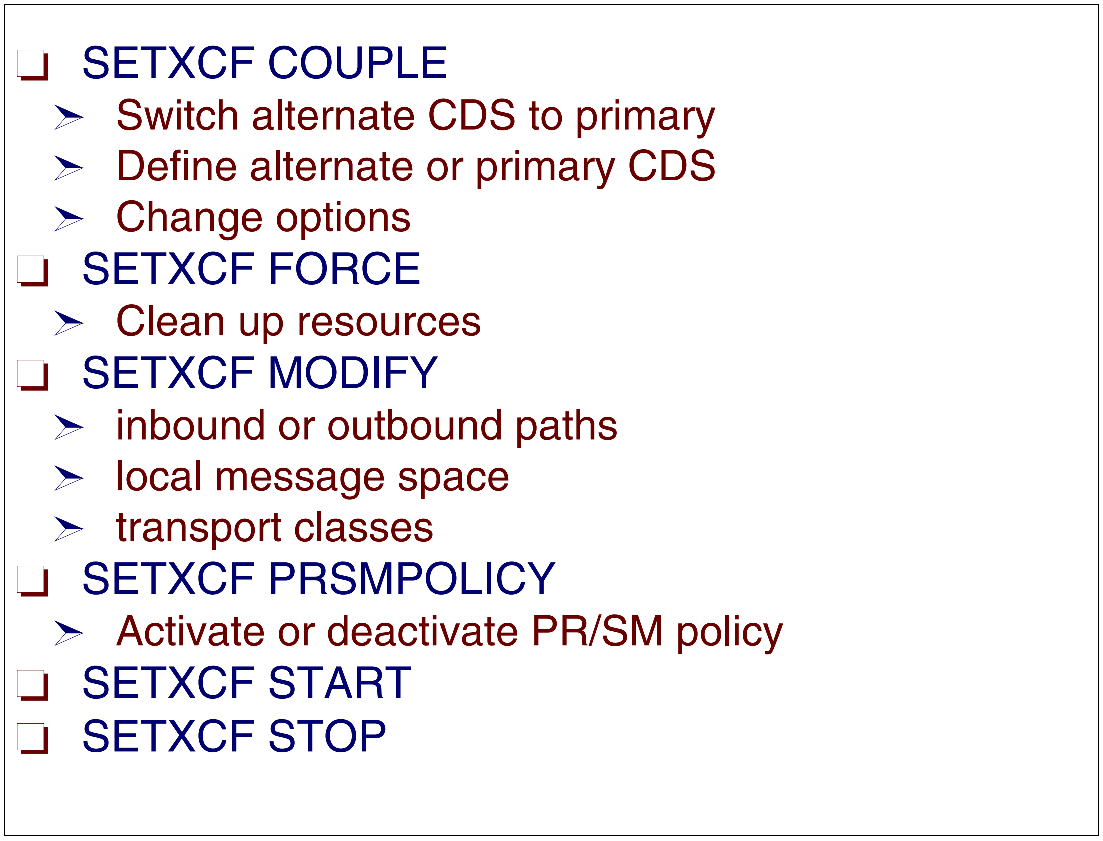
Figure 1-63 Display XCF signalling paths (2)

Display XCF signaling paths (2)

The **D XCF,PO,DEVICE=ALL,STATUS=WORKING** command shows, in list format, the outbound CTC devices and their inbound device number partners. It also gives an overview of the message classes and buffer sizes assigned to the CTC devices.

The second part of the display shows statistics if there is any transfer problem. There is no `TRANSFER PENDING` and the `MXFER TIME` in microseconds is below the recommended threshold value of 2000 or 2 milliseconds. `MXFER` is the mean transfer time for up to the last 64 signals received within the last minute. Values above 2 milliseconds may indicate that there is not enough CTC capacity. In this case RMF-based measurements should be done.

1.61 SETXCF command



- SETXCF COUPLE
  - Switch alternate CDS to primary
  - Define alternate or primary CDS
  - Change options
- SETXCF FORCE
  - Clean up resources
- SETXCF MODIFY
  - inbound or outbound paths
  - local message space
  - transport classes
- SETXCF PRSMPOLICY
  - Activate or deactivate PR/SM policy
- SETXCF START
- SETXCF STOP

Figure 1-64 The SETXCF command

SETXCF command

The `SETXCF` command is used to control the sysplex environment. It has some variations, according to the action requested, as explained below.

SETXCF COUPLE

The `SETXCF COUPLE` command is used to:

- Switch a current alternate CDS to a primary CDS. The switch can be for either sysplex CDSs or other types of CDSs.
- Specify a primary non-sysplex CDS, such as CFRM, SFM, or WLM.
- Specify an alternate CDS.
- Change options specified in the COUPLExx parmlib member.

SETXCF FORCE

The `SETXCF FORCE` command is used to clean up resources related to structures in a CF. The resources can be either structures actively in use in the sysplex or dumps associated with structures pending deallocation.

SETXCF MODIFY

The `SETXCF MODIFY` command is used to change current XCF parameters. The system changes only those parameters explicitly provided on the `SETXCF MODIFY` command; all other parameters associated with the resource remain the same. You can use this to modify:

- Inbound paths
- Outbound paths
- Local message space
- Transport classes

SETXCF PFRSMPOLICY

The `SETXCF PRSMPOLICY` command is used either to activate an XCF PR/SM policy, or to deactivate a current active XCF PR/SM policy.

SETXCF START

The `SETXCF START` command is used to:

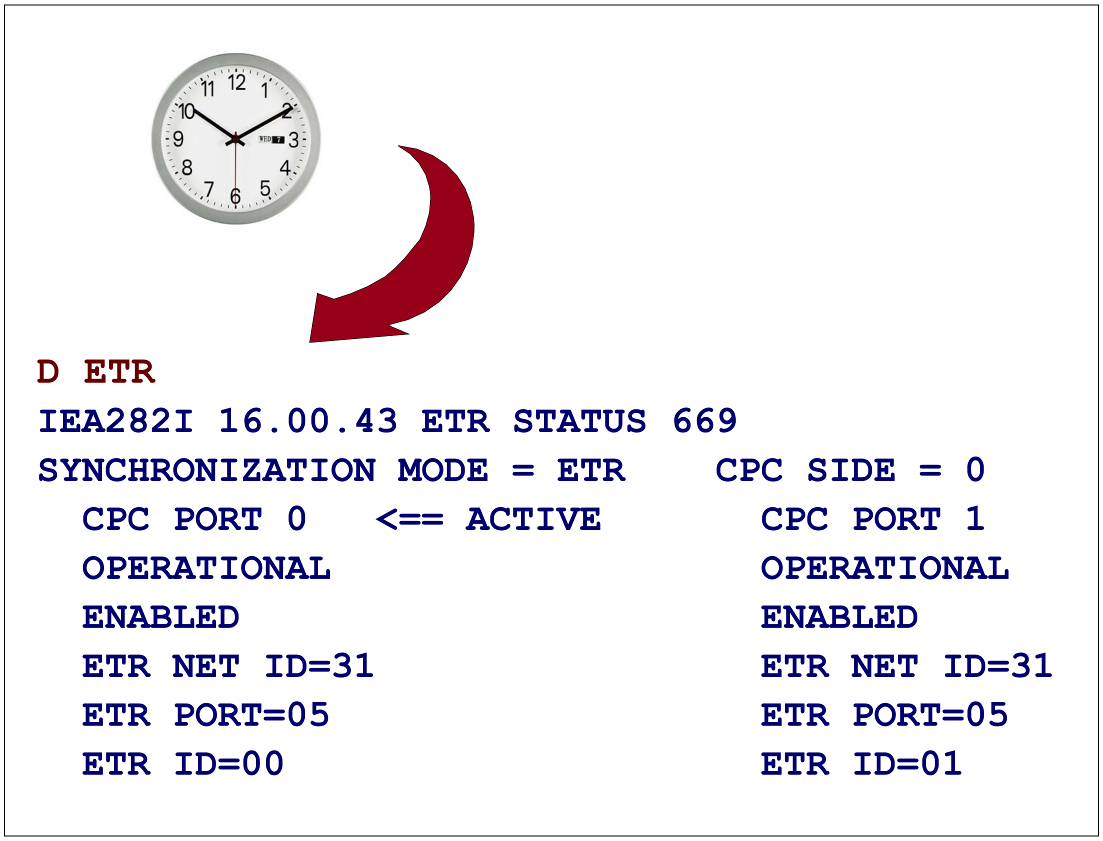
- Start new inbound signaling paths or restart inoperative inbound signaling paths.
- Start outbound signaling paths or restart inoperative outbound signaling paths.
- Define transport classes.
- Start using a new administrative policy as an active policy.
- Start rebuilding one or more CF structures either in the same CF or in another CF.
- Start populating a CF that has been newly brought into service or returned to service in a sysplex with structures selected from the set of those defined in the active CFRM policy. The structures selected are those that list the CF to be populated as higher in the structure's preference list than the CF in which the structure already is allocated.
- Start user-managed duplexing of one or more structures in a CF into another CF.
- Start altering the size of a CF structure.

SETXCF STOP

The `SETXCF STOP` command is used to:

- Stop one or more inbound signaling paths.
- Stop one or more outbound signaling paths.
- Delete the definition of a transport class.
- Stop using an administrative policy.
- Stop rebuilding one or more CF structures.
- Stop populating a CF that had been newly brought into service in a sysplex with structures selected from the set of those defined in the active CFRM policy.
- Stop user-managed duplexing of one or more structures in a CF and specify the structure that is to remain in use.
- Stop altering a CF structure.

1.62 Managing the external timer



```
D ETR
IEA282I 16.00.43 ETR STATUS 669
SYNCHRONIZATION MODE = ETR    CPC SIDE = 0
  CPC PORT 0   <== ACTIVE       CPC PORT 1
  OPERATIONAL                   OPERATIONAL
  ENABLED                       ENABLED
  ETR NET ID=31                 ETR NET ID=31
  ETR PORT=05                   ETR PORT=05
  ETR ID=00                     ETR ID=01
```

Figure 1-65 Managing external timer

Managing the external timer

To manage the external timer reference (ETR) you can use the following z/OS commands:

- **`DISPLAY ETR`** displays the current and status, in detail, of each ETR port, giving the ETR network ID, ETR port number, and the ETR ID. The complete syntax is:

  `D ETR,DATA`

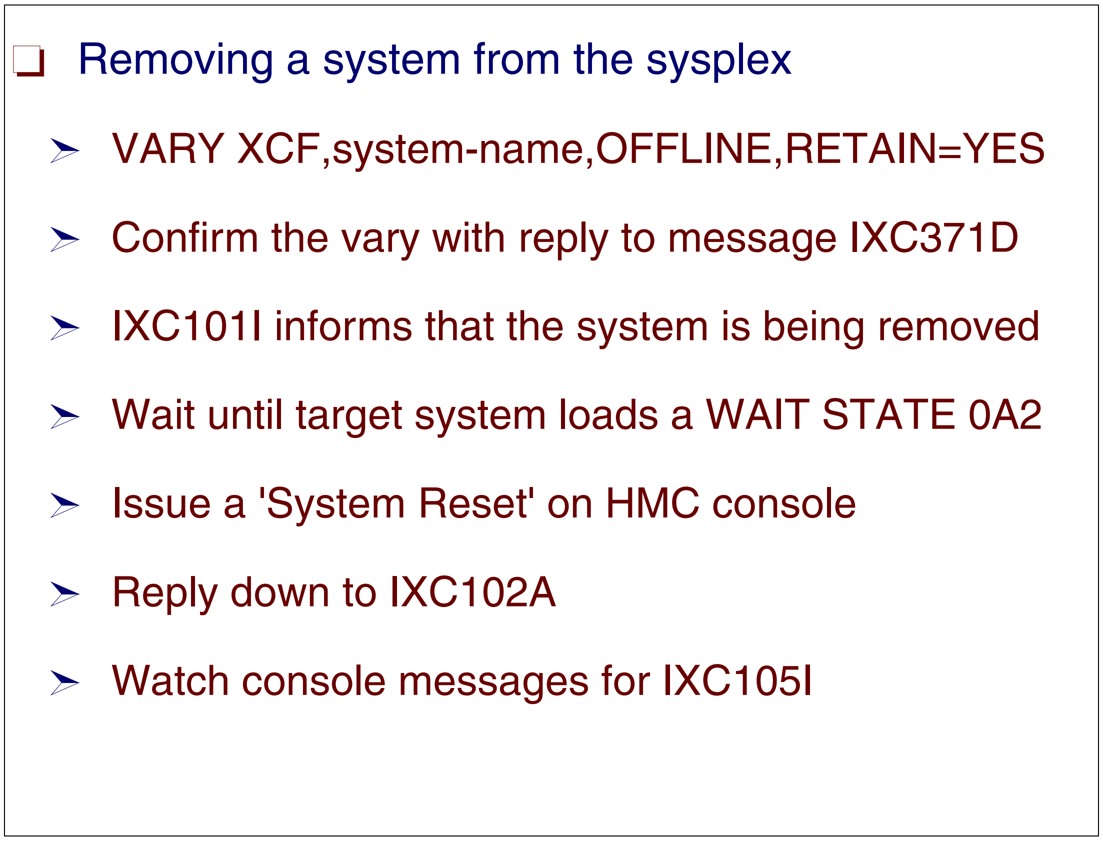
  `DATA` is the default, so you can use just **`D ETR`**.

- **`SETETR PORT=nn`** can be used to enable ETR ports that have been disabled. An ETR port disabled by a hardware problem can be enabled after the problem has been corrected. **`PORT=nn`** specifies the number of the ETR port to be enabled. The valid values for n are 0 and 1.

- **`MODE`** is used to control the actions of recovery management when certain types of machine check interruptions occur. The actions you can control are recording/monitoring or suppressing status for each type of machine check interruption on the logrec system data set.

`MODE AD`	The AD parameter defines machine checks indicating the ETR attachment is to be monitored in the specified mode.
`MODE SC`	The SC parameter defines machine checks indicating the ETR synchronization checks are to be monitored in the specified mode.

1.63 Removing a system from the sysplex


❑ Removing a system from the sysplex

- VARY XCF,system-name,OFFLINE,RETAIN=YES
- Confirm the vary with reply to message IXC371D
- IXC101I informs that the system is being removed
- Wait until target system loads a WAIT STATE 0A2
- Issue a 'System Reset' on HMC console
- Reply down to IXC102A
- Watch console messages for IXC105I

Figure 1-66 Removing a system from the sysplex

Removing a system from the sysplex

Removing a system from the sysplex means that:

- All XCF group members in the sysplex know that the system is being removed, so they can perform any necessary cleanup processing.
- All I/O to sysplex-shared resources is completed, to ensure data integrity.
- All signaling paths are configured to a desired state (retained or not).

Commands to remove a system

Use the **VARY XCF** command to remove a system from the sysplex. You can remove the system temporarily or permanently. To temporarily remove a system from the sysplex, issue:

```
VARY XCF,system-name,OFFLINE,RETAIN=YES
```

With **RETAIN=YES** (the default), MVS on each remaining system in the sysplex retains the definition of the devices for the signaling paths that connected to the removed system. Therefore, the removed system can be re-IPLed into the sysplex or another system can be added in its place, and MVS automatically starts the signaling paths. Note that the last system removed from the sysplex, remains defined to the sysplex couple data set. To permanently remove a system from the sysplex, issue:

```
VARY XCF,system-name,OFFLINE,RETAIN=NO
```

Reply to message IXC371D, which requests confirmation of the **VARY XCF** command. Message IXC101I then informs you that the system is being removed.

When the target system is in a wait state (issues message IXC220W with the wait state code 0A2), issue a *system reset*. The reset *must* be done after the wait state, to ensure the integrity of I/O to sysplex-shared I/O resources.

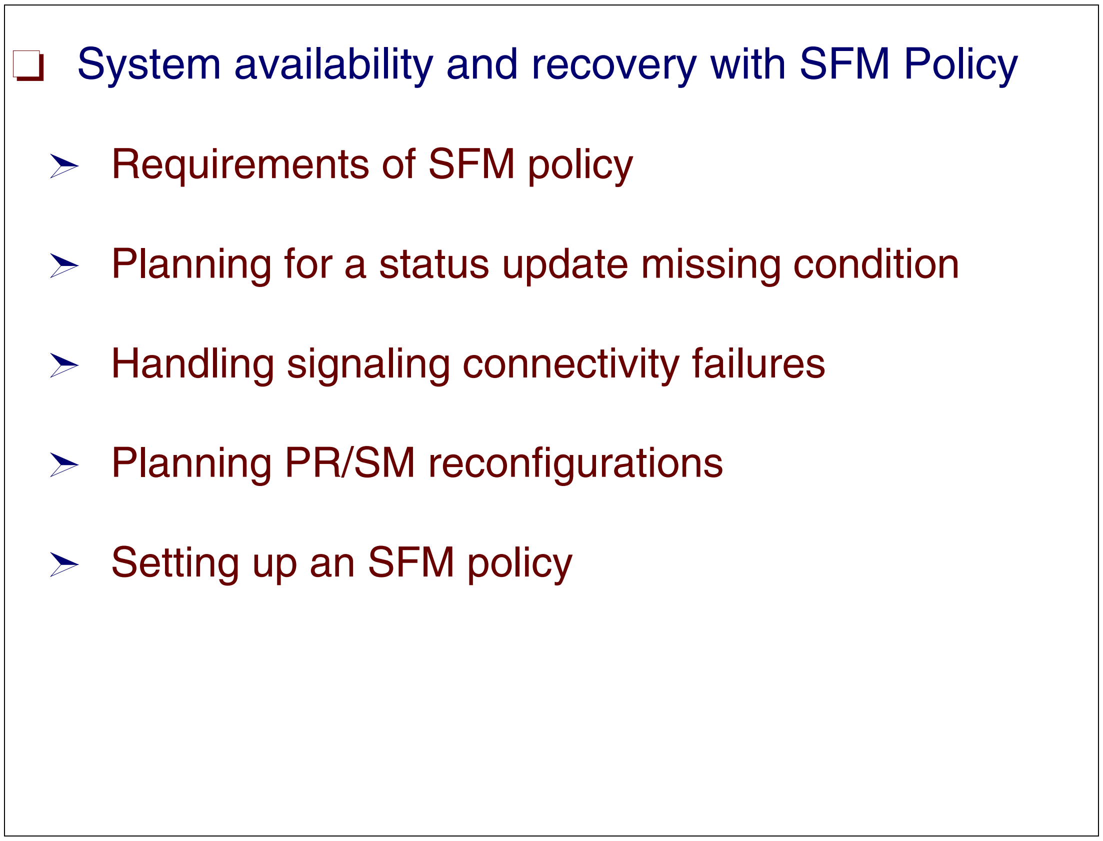
Reply **`DOWN`** to message IXC102A to continue removing the system from the sysplex.

After you reply, when the removal of the system is complete, message IXC105I is issued.

For more information about removing a z/OS system from a sysplex refer to the IBM Parallel Sysplex home page at:

`http://www.ibm.com/s390/pso/removing.html`

1.64 Sysplex failure management (SFM)



- System availability and recovery with SFM Policy
  - Requirements of SFM policy
  - Planning for a status update missing condition
  - Handling signaling connectivity failures
  - Planning PR/SM reconfigurations
  - Setting up an SFM policy

Figure 1-67 Sysplex failure management (SFM)

Sysplex failure management (SFM)

Sysplex failure management (SFM) allows you to define a sysplex-wide policy that specifies the actions that MVS is to take when certain failures occur in the sysplex. A number of situations might occur during the operation of a sysplex when one or more systems must be removed so that the remaining sysplex members can continue to do work. The goal of SFM is to allow these reconfiguration decisions to be made and carried out with little or no operator involvement.

Overview and requirements of SFM policy

If the sysplex includes a coupling facility, the full range of failure management capabilities that SFM offers is available to the sysplex. For SFM to handle signaling connectivity failures without operator intervention or to isolate a failing system, a coupling facility must be configured in the sysplex.

An SFM policy includes the following statements:

- Policy statement
- System statements
- Reconfiguration statements

Planning for a status update missing condition

If any system loses access to the SFM couple data set, the policy becomes inactive in the sysplex. If that system regains access to the SFM couple data set, SFM automatically becomes active again in the sysplex.

For a sysplex to take advantage of an SFM policy, the policy must be active on all systems in the sysplex. That is:

- All systems must be running a supported z/OS operating system.
- An SFM policy must be started in the SFM couple data set.
- All systems must have connectivity to the SFM couple data set.

Handling signaling connectivity failures

All systems in the sysplex must have signaling paths to and from every other system at all times. Loss of signaling connectivity between sysplex systems can result in one or more systems being removed from the sysplex so that the systems that remain in the sysplex retain full signaling connectivity to one another. SFM can eliminate operator intervention when signaling connectivity between two or more systems is lost.

Planning PR/SM reconfigurations

Loss of connectivity to a coupling facility can occur because of a failure of the coupling facility attachment or because of certain types of failures of the coupling facility itself. MVS provides the capability to initiate a rebuild of one or more structures in the coupling facility to which connectivity has been lost, using the CFRM policy and optionally, the SFM policy.

Setting up an SFM policy

The administrative data utility, IXCMIAPU allows you to associate the definitions with a policy name and to place the policy in a pre-formatted SFM couple data set.

To implement an SFM policy, you need to:

- Format an SFM couple data set and ensure that it is available to all systems in the sysplex.
- Define the SFM policy.
- Start the SFM policy in the SFM couple data set.

1.65 Parallel Sysplex complex

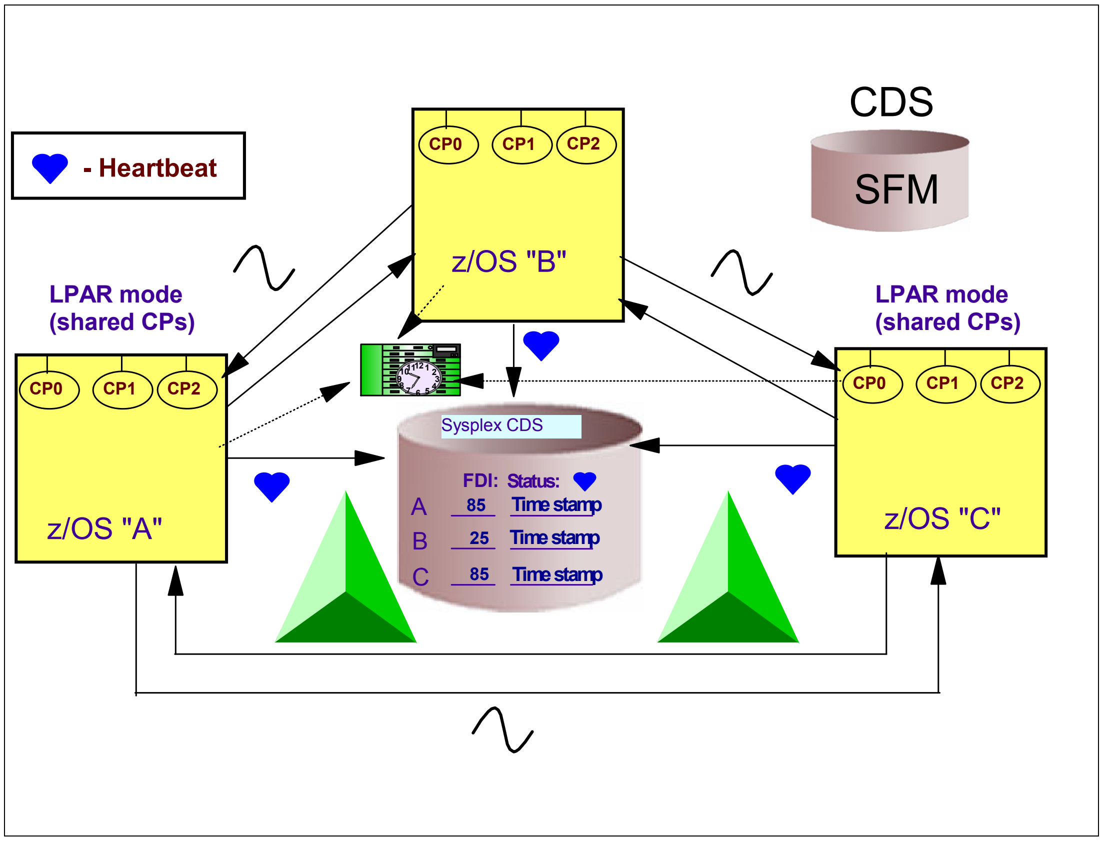


Figure 1-68 Parallel Sysplex complex

Parallel Sysplex complex

Until all systems are IPLed and join the sysplex, a mixed complex exists; that is, one or more of the systems in the global resource serialization complex are not part of the sysplex.

In multisystem sysplex mode, you need:

- A formatted primary sysplex couple data set shared by all systems in the sysplex.
- Signaling connectivity between all systems in the sysplex.
- The same Sysplex Timer for all systems in a sysplex that includes more than one CPC.

Sysplex status monitoring

Each system in the sysplex periodically updates its own status and monitors the status of other systems in the sysplex. The status of the systems is maintained in a couple data set (CDS) on DASD. A status update missing condition occurs when a system in the sysplex does not update its status information in either the primary or alternate couple data set within the failure detection interval, specified on the INTERVAL keyword in COUPLExx parmlib member, and appears dormant.

SFM allows you to specify how a system is to respond to this condition. System isolation allows a system to be removed from the sysplex as a result of the status update missing condition, without operator intervention, thus ensuring that the data integrity in the sysplex is preserved. Specifically, system isolation uses special channel subsystem microcode in the target CPC to cut off the target LP from all I/O and coupling facility accesses. This results in the target LP loading a non-restartable wait state, thus ensuring that the system is unable to corrupt shared resources.

SFM couple data set (CDS)

Sample JCL to run the format utility for formatting couple data sets for the sysplex failure management service is shipped in SYS1.SAMPLIB member IXCSFMF.

For an SFM couple data set, (DATA TYPE(SFM), valid data names are POLICY, SYSTEM, and RECONFIG.

ITEM NAME(POLICY) NUMBER( )	Specifies the number of administrative policies that can be defined. (Default=9, Minimum=1, Maximum=50)
ITEM NAME(SYSTEM) NUMBER( )	Specifies the number of systems for which actions and weights can be defined. This should be the maximum number of systems that will be in the sysplex that the policy will govern. Note that the number specified does not need to include those systems identified by NAME(*), for which policy default values are applied. (Default=8, Minimum=0, Maximum=32)
ITEM NAME(RECONFIG) NUMBER( )	Specifies the number of reconfigurations involving PR/SM partitions that can be specified. Default=0, Minimum=0, Maximum=50)

Failure detection interval (FDI)

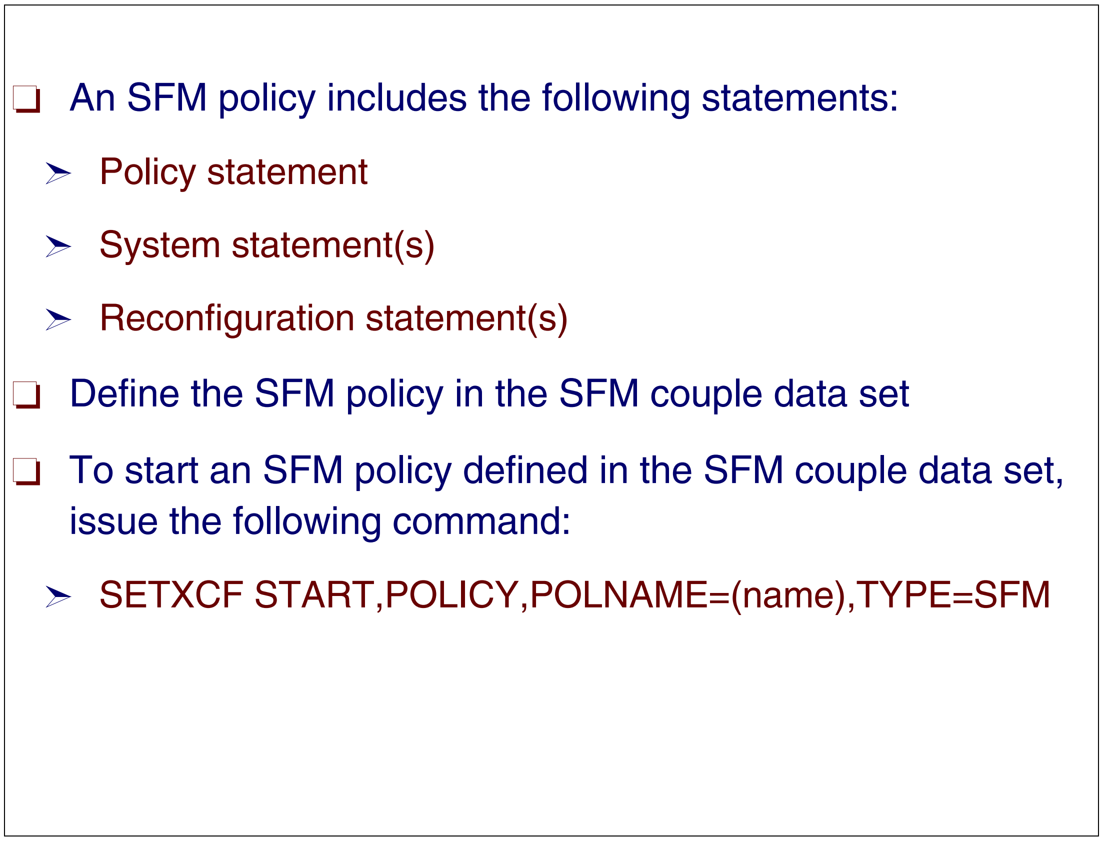
The failure detection interval (FDI) is the amount of time that a system can appear to be unresponsive before XCF will take action to remove the system from the sysplex. Internally we refer to this as the effective FDI, externally it is often designated by the word INTERVAL (referring to the INTERVAL parameter in COUPLExx parmlib member and on the `SETXCF COUPLE` command).

Note: It is recommended that the user let the system default the effective FDI to the SpinFDI by not specifying the INTERVAL keyword. The INTERVAL keyword allows customers to specify an effective FDI that is larger than the Spin FDI. When specified, the INTERVAL value should be at least as large as the SpinFDI to give the system enough time to resolve a spin loop timeout before it gets removed from the sysplex, but no so large that the rest of the sysplex suffers sympathy sickness.

Coupling facility

If the sysplex includes a coupling facility, the full range of failure management capabilities that SFM offers is available to the sysplex. For SFM to handle signaling connectivity failures without operator intervention or to isolate a failing system, a coupling facility must be configured in the sysplex.

1.66 Requirements of SFM policy



- ❑ An SFM policy includes the following statements:
  - ➢ Policy statement
  - ➢ System statement(s)
  - ➢ Reconfiguration statement(s)
- ❑ Define the SFM policy in the SFM couple data set
- ❑ To start an SFM policy defined in the SFM couple data set, issue the following command:
  - ➢ SETXCF START,POLICY,POLNAME=(name),TYPE=SFM

Figure 1-69 Requirements of SFM policy

Requirements of SFM policy

The SFM policy includes all the function available through XCFPOLxx parmlib member. If a system is connected to a couple data set with a started SFM policy, all XCFPOLxx parmlib member specifications on that system are deactivated, regardless of whether the SFM policy is active in the sysplex.

Because the SFM policy provides function beyond that provided by the XCF PR/SM policy, it is generally recommended that you use the SFM policy to manage failures in your sysplex. However, in cases where you cannot activate an SFM policy, activating an XCF PR/SM policy can be useful.

SFM allows you to define a sysplex-wide policy that specifies the actions that MVS is to take when certain failures occur in the sysplex. A number of situations might occur during the operation of a sysplex when one or more systems need to be removed so that the remaining sysplex members can continue to do work. The goal of SFM is to allow these reconfiguration decisions to be made and carried out with little or no operator involvement.

SFM policy

For a sysplex to take advantage of an SFM policy, the policy must be active on all systems in the sysplex. That is:

- An SFM policy must be started in the SFM couple data set.
- All systems must have connectivity to the SFM couple data set.

If any system loses access to the SFM couple data set, the policy becomes inactive in the sysplex. If that system regains access to the SFM couple data set, SFM automatically becomes active again in the sysplex.

Note: Similarly, if a system joins the sysplex where an SFM policy is active, the policy is disabled for the entire sysplex. When that system is removed from the sysplex, SFM automatically becomes active again in the sysplex.

Specifying an SFM policy in the CDS

The administrative data utility, IXCMIAPU allows you to associate the definitions with a policy name and to place the policy in a pre-formatted SFM couple data set.

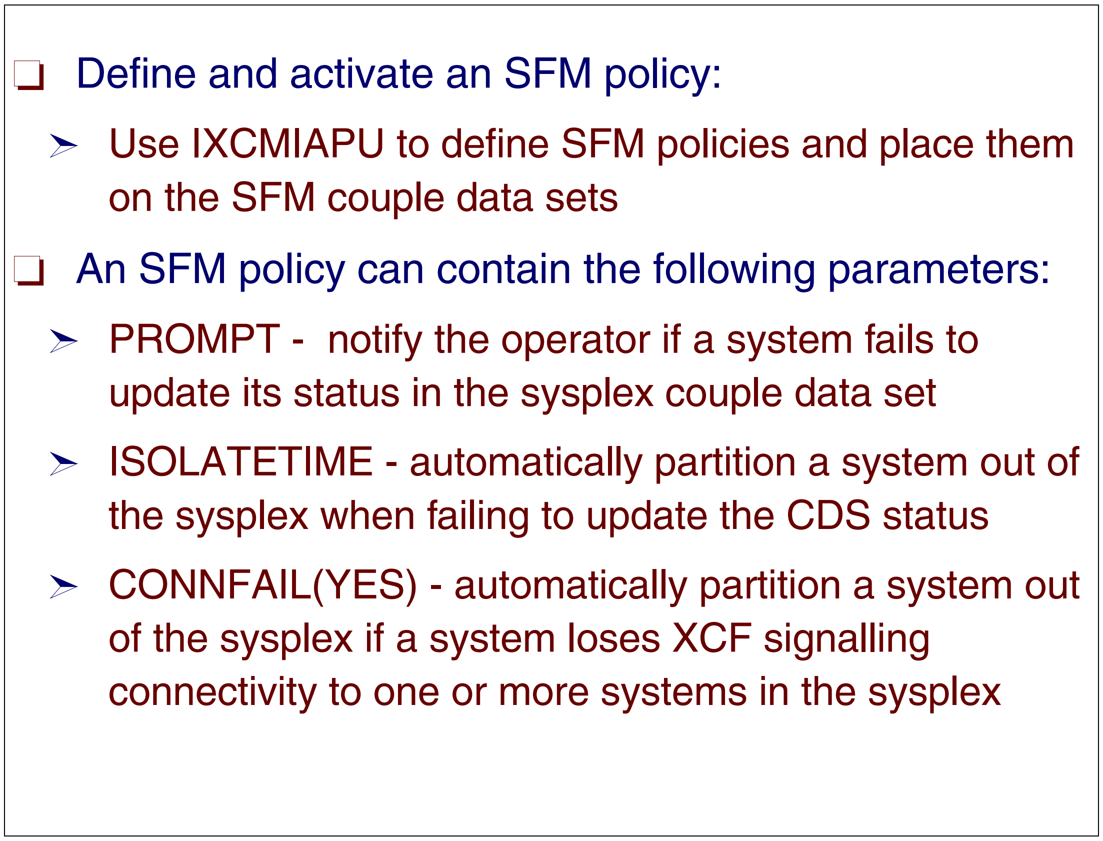
To start an SFM policy (POLICY1, for example) that is defined in the SFM couple data set, issue the following command:

```
SETXCF START,POLICY,POLNAME=POLICY1,TYPE=SFM
```

Note: The SFM policy includes all the function available through XCFPOLxx parmlib member. However, if a system is connected to a couple data set with a started SFM policy, all XCFPOLxx parmlib member specifications on that system are deactivated, regardless of whether the SFM policy is active in the sysplex.

Because the SFM policy provides function beyond that provided by the XCF PR/SM policy, it is generally recommended that you use the SFM policy to manage failures in your sysplex. However, in cases where you cannot activate an SFM policy, activating an XCF PR/SM policy can be useful.

1.67 SFM implementation



- Define and activate an SFM policy:
  - Use IXCMIAPU to define SFM policies and place them on the SFM couple data sets
- An SFM policy can contain the following parameters:
  - PROMPT - notify the operator if a system fails to update its status in the sysplex couple data set
  - ISOLATETIME - automatically partition a system out of the sysplex when failing to update the CDS status
  - CONNFAIL(YES) - automatically partition a system out of the sysplex if a system loses XCF signalling connectivity to one or more systems in the sysplex

Figure 1-70 SFM implementation

SFM implementation

Use the IXCMIAPU utility to define SFM policies and place them on the SFM couple data sets. An SFM policy can contain the following parameters:

- Specify PROMPT to notify the operator if a system fails to update its status in the sysplex couple data set (this results in the same recovery action as when SFM is inactive). The COUPLExx(OPNOTIFY) parameter controls when the operator is prompted.

  Note: You cannot specify SSUMLIMIT if you specify the PROMPT parameter.

- Specify ISOLATETIME to automatically partition a system out of the sysplex if a system fails to update the sysplex couple data set with status.
- Specify CONNFAIL(YES) to automatically partition a system out of the sysplex if a system loses XCF signalling connectivity to one or more systems in the sysplex.
- DEACTTIME or RESETTIME can be used to automate recovery actions for system failures if a coupling facility is not available. ISOLATETIME is recommended instead of DEACTTIME or RESETTIME.

Note: PROMPT, ISOLATETIME, DEACTTIME and RESETTIME are mutually exclusive parameters. Use ISOLATETIME instead of RESETTIME/DEACTTIME because:

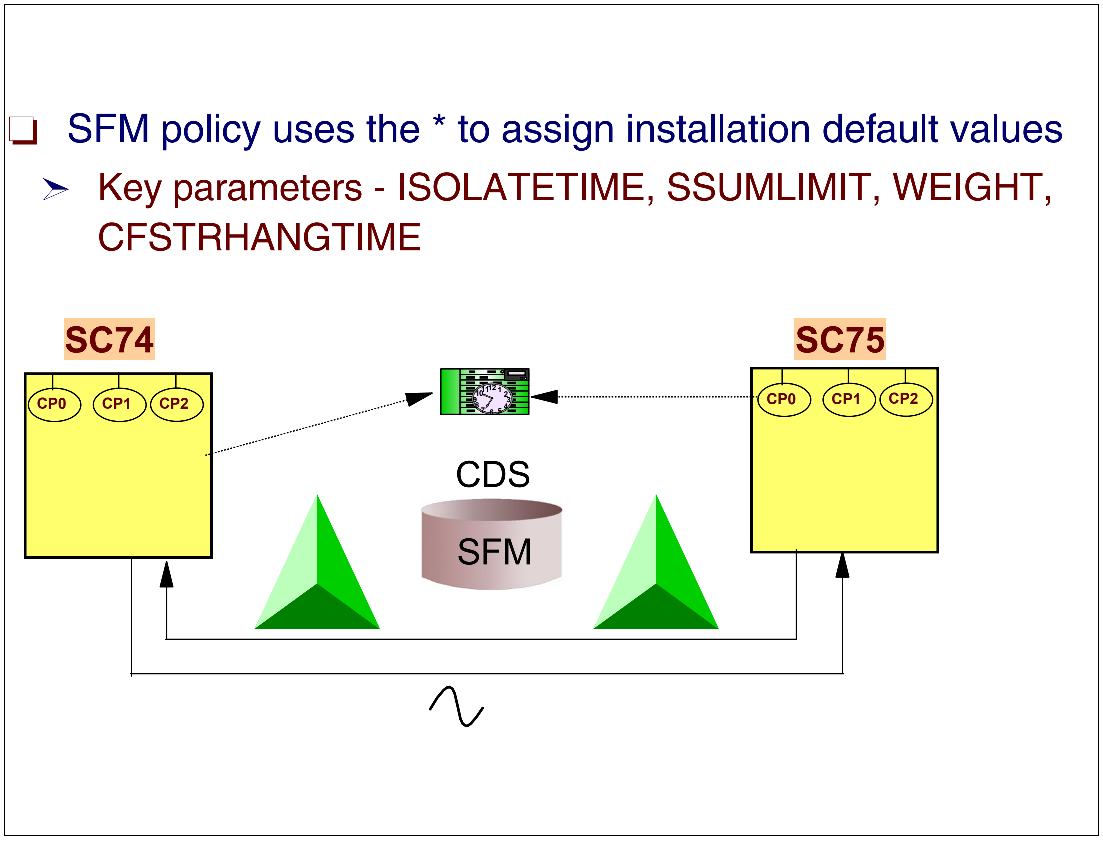
- ISOLATETIME is sysplex-wide. RESETTIME and DEACTTIME are effective only within a processor.
- ISOLATETIME quiesces I/O. RESETTIME and DEACTTIME imply a hard I/O reset.

Policy considerations

Consider the following guidelines when defining a policy:

- Assign weights to each system to reflect the relative importance of each system in the sysplex.
- The RECONFIG parameter can be used to reconfigure storage resources to a “backup” LPAR in the event of a production MVS failure.
- Use the `SETXCF` command to start and stop SFM policies.
- SFM accomplishes automatic removal of systems from the sysplex by performing a system isolation function (also known as fencing) for the system being removed. Fencing requires a coupling facility.
- When a system is shut down (planned), or if a system fails (unplanned), that system must be removed from the sysplex as soon as possible to prevent delays on other systems in the sysplex. Systems that are removed from the sysplex must not persist outside the sysplex (they must be fenced, system-reset, or disabled in some other manner such as powered off).
- SFM is a sysplex-wide function. Like CFRM, the SFM policy need only be started on one system in the sysplex to be active sysplex-wide.
- Only one SFM policy can be active at a time.
- Do not specify a DSN on the administrative data utility JCL.
- All active systems require connectivity to SFM couple data sets for the SFM policy to remain active.
- If SFM becomes inactive, PROMPT is reinstated as the recovery action initiated on a system failure.
- When PROMPT is specified or defaulted to, COUPLExx(OPNOTIFY) controls when IXC402D is issued.

1.68 SFM policy parameters



- SFM policy uses the * to assign installation default values
  - Key parameters - ISOLATETIME, SSUMLIMIT, WEIGHT, CFSTRHANGTIME

Figure 1-71 SFM policy parameters

SFM policy parameters

The following example shows the use of the * to assign installation default values in an SFM policy. There are many more parameters, but these are somewhat important.

With z/OS V1R12, the following parameters were changed to the values shown in Figure 1-72:

```
WEIGHT(25) - ISOLATETIME(0) - SSUMLIMIT(150) - CFSTRHANGTIME(300)
```

System SC74

In this example, system SC74 uses the following parameter values:

- It requires the ISOLATETIME value of 100 seconds and accepts the system default of SSUMLIMIT(NONE). The SSUMLIMIT(150) specified on the SYSTEM NAME(*) statement does not apply to SC74, because the SYSTEM NAME(SC74) statement specifies an indeterminate status action of ISOLATETIME(100).
- It uses policy default WEIGHT value of 10 and CFSTRHANGTIME value of 300 established by the SYSTEM NAME(*) statement.
- It uses the system defaults for all attributes not specified on either the SYSTEM NAME(SC74) or the SYSTEM NAME(*) statement, for example, MEMSTALLTIME(NO).

System SC75

System SC75 uses the following parameter values:

- It requires a WEIGHT value of 25.
- It uses the policy default combination of ISOLATETIME(0), SSUMLIMIT(150), and the policy default CFSTRHANGTIME value of 300 established by the SYSTEM NAME(*) statement.

It uses the system defaults for all attributes not specified on either the SYSTEM NAME(SC75) or the SYSTEM NAME(*) statement, for example, MEMSTALLTIME(NO).

All other systems

All other systems use the following parameter values:

- The policy default combination of ISOLATETIME(0) SSUMLIMIT(150), the policy default WEIGHT value of 10, and the policy default CFSTRHANGTIME value of 300 established by the SYSTEM NAME(*) statement.
- The system defaults for all other attributes, for example, MEMSTALLTIME(NO).

```
DEFINE POLICY NAME(POLICY1) ...

   SYSTEM NAME(SC74)
      ISOLATETIME(100)

   SYSTEM NAME(SC75)
      WEIGHT(25)

   SYSTEM NAME(*)
      WEIGHT(10)
      ISOLATETIME(0) SSUMLIMIT(150)
      CFSTRHANGTIME(300)
```

Figure 1-72 SFM policy parameters

Installation considerations

ISOLATETIME(0) is the default when none of the DEACTTIME, RESETTIME, PROMPT, or ISOLATETIME parameters is specified. IBM suggests using ISOLATETIME(0) to allow SFM to isolate and partition a failed system without operator intervention and without undue delay.

2

System Logger

In a Parallel Sysplex, each system has its own Syslog and Logrec data sets. With multiple systems, you want to merge the logs from all systems so that you have a comprehensive time stamp or a sequence number order view of all log data within the Parallel Sysplex.

Also, multisystem applications like CICS and DB2 have their own logs. You can use System Logger services to accumulate log records and merge them from multiple instances of an application as CICS or DB2, including merging data from different systems across a sysplex.

z/OS provides the System Logger, which is a set of services that allow you to manage log data across the systems in the Parallel Sysplex. System Logger has its own required couple data set, LOGR, that contains information (a policy) on how to manage System Logger resources.

In this chapter we present a general overview of the System Logger, as follows:

- System Logger terminology
- Log stream recording media
- z/OS System Logger benefits
- Settings for System Logger
- Creating and formatting the LOGR couple data set
- LOGR policy definition
- Managing log data
- Log streams in a coupling facility
- DASD only log streams
- System Logger services
- System Logger exploiters

Note: More details about System Logger are in:

- *Systems Programmer's Guide to z/OS System Logger*, SG24-6898
- *z/OS MVS Setting Up a Sysplex,* SA22-7625

2.1 System Logger terminology

- Log record
- Log block
- Interim storage
- Log stream
- Staging data sets
- Offload and offload data sets

Figure 2-1 System Logger terminology

Log record

Log record is a single record with a time stamp (or a sequence number) on it produced by the System Logger exploiter. It could be something like a line of syslog (the log of the z/OS consoles), for example. Or it could be a before-image created by a data manager before it updates a record in a database, and also an after-image.

Log block

The log data can be sent to System Logger every time a new log record is created, or log records can be written to a buffer and subsequently sent to System Logger. In either case, the piece of data that is sent to System Logger is called a log block. It is transparent to System Logger whether the log block contains a single log record or many log records.

Interim storage

Interim storage is the primary storage used to hold log data that has not yet been hard coded on a DASD offload data set. The interim storage medium can be defined to:

- A coupling facility list structure. To prevent data loss conditions the log data is duplexed to a data space, although log streams residing in a CF structure might optionally be duplexed to a staging data set.
- A staging data set. It can be used as interim storage on DASD for a DASD-only log stream configuration, or to duplex log data in the coupling facility that has not yet been offloaded to offload data sets.

Log stream

A log stream is an application-specific collection of one or more log blocks written using System Logger services. The data is written to and read from the log stream by one or more instances of the application associated with the log stream. A log stream can be used for such purposes as a transaction log, a log for recreating databases, a recovery log, or other logs needed by applications. If the application has multiple instances on different z/OS images writing log blocks to the same log stream, the result is a sysplex-wide log stream.

There are two types of log streams, depending on the storage medium System Logger uses to hold interim log data:

- Coupling facility log streams, using CF list structures
- DASD-only log streams

An installation can use either coupling facility log streams, DASD-only log streams, or a combination of both types of log streams.

Staging data sets

Staging data sets are required for DASD-only log streams, and are optional for CF structure-based log streams. The naming convention for staging data sets differs based on the type of log stream:

- For a log stream residing in a CF structure, the naming convention is:

  `<EHLQ or HLQ>.<streamname>.<system name>`

- For DASD-only log streams, the naming convention is:

  `<EHLQ or HLQ>.<streamname>.<sysplex name>`

Offload action and offload data sets

As applications write data to a log stream, the interim storage defined for log data begins to fill, eventually reaching or exceeding its high threshold. Offloading is the process of moving valid, and not yet deleted, data from interim storage to offload data sets, respecting the time stamp or a sequence number order.

Offload data sets are VSAM linear data sets, sometimes also referred to as log data sets or log stream data sets. The naming convention for offload data sets is:

`<EHLQ or HLQ>.<streamname>.<seq#>`

LOGR couple data set

The LOGR couple data set (CDS) holds the System Logger policy information and information about all defined log streams, and must be accessible by all the z/OS System Loggers in the sysplex.

Defining the LOGR CDS and the System Logger policy information is one of the first tasks you must complete when preparing to use a System Logger environment.

2.2 Log stream recording media

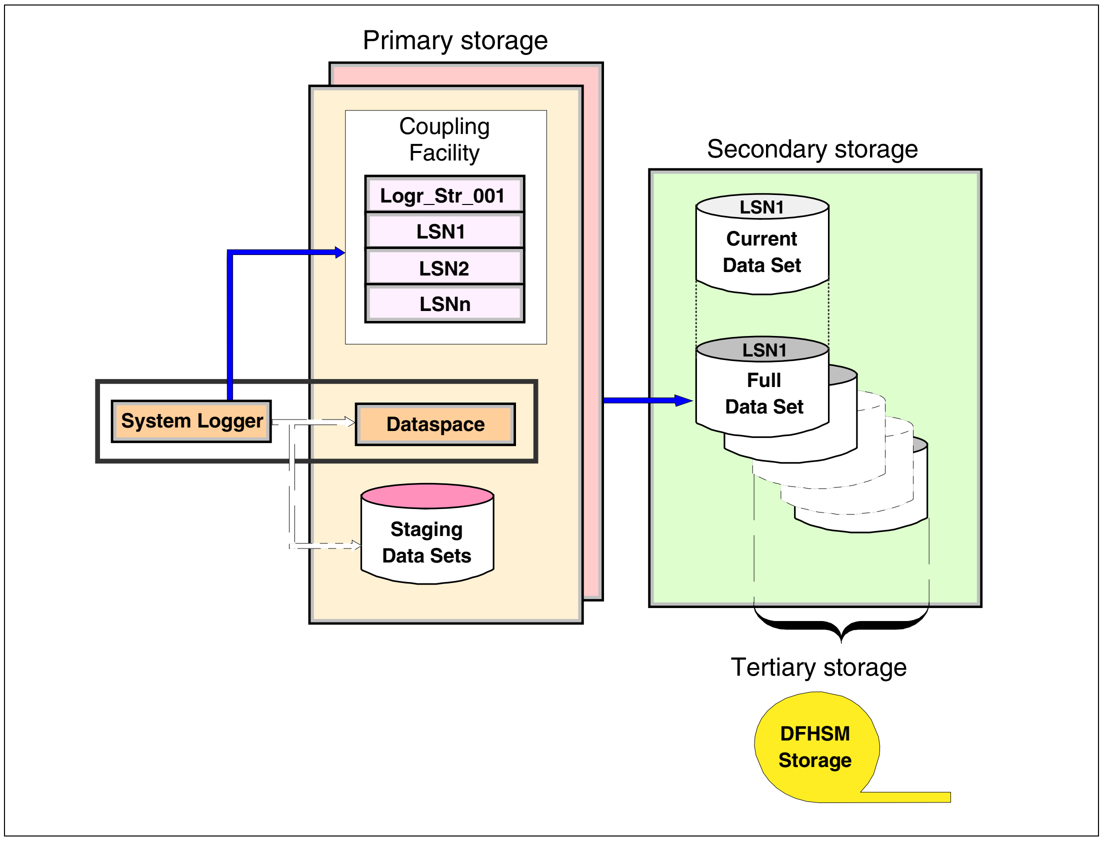


Figure 2-2 Log Stream recording media

Log stream recording media

Depending on the life span of the log data, it can be moved from the initial interim storage to different media (offload data sets) by System Logger. There is a hierarchy for how the log data is stored in the system. The data should be stored on a medium with good performance because each application has to wait for *commit back* that the data is logged successfully. The hierarchy is as follows:

1. Interim storage, also called primary storage, where data can be accessed quickly without incurring DASD I/O. When the interim storage medium for a log stream reaches a user-defined threshold, the log data is off-loaded to DASD data sets called offload data sets.
2. DASD log data set (offload data set) or secondary, for longer term access.
3. If your installation is planning in keeping the log data for a long time, you might consider having a storage management tool that will migrate the offload data sets to a tertiary storage level, usually tapes. Remember that the offload data sets are still listed in the inventory of the LOGR couple data sets, even if they are migrated to a different medium.

2.3 z/OS System Logger benefits

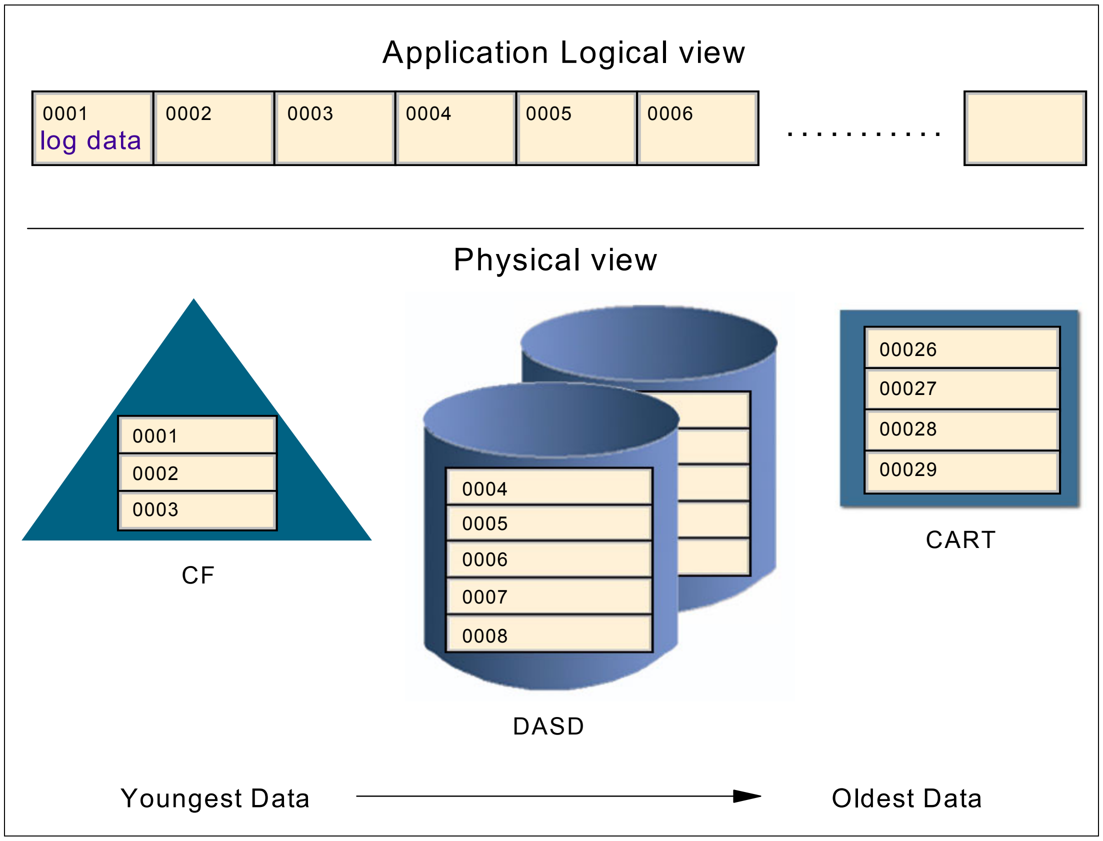


Figure 2-3 Logical and physical view of the log data

Benefits of using System Logger

System Logger is a z/OS component that provides a logging facility for applications running in a single-system or multisystem sysplex. Using System Logger services exempts the application from the responsibility for tasks such as:

- Saving the log data, with the requested persistence
- Retrieving the data from any system in the sysplex
- Archiving the data, and expiring the data

In addition, System Logger provides the ability to have a single or merged log containing log data from multiple instances of an application in the sysplex.

Now a quiz: Instead of using System Logger services, why not have all the CICS instances (from different z/OSs) just write the log records sequentially to the same data set, eliminating the need for System Logger?

Answer: If two log events occur in two CICS, there is no guarantee that the one requesting a write first (because it happened before) will be written in the common log before the other one, which happened later in other CICS, and there is no guarantee that the records will be ordered by the time stamp in the log data set.

Log data

Log data managed by the System Logger may reside in or on the following:

- Central storage (data space)
- Coupling facility structure
- DASD staging data sets
- Tape, potentially

However, regardless of where System Logger is currently storing a given log record, from the point of view of the exploiter, all the log records are kept in a single log stream. The task of tracking where a specific piece of log data is at any given time is handled by System Logger. Additionally, System Logger manages the utilization of its storage. As the space in one medium starts filling up (a coupling facility structure, for example), System Logger moves old data to the next lower level in the hierarchy. The location of the data, and the migration of that data from one level to another, is transparent to the application; it is managed completely by System Logger, with the objective of providing optimal performance while maintaining the integrity of the data.

Storing log data

When an application passes log data to System Logger, the data can *initially* be stored:

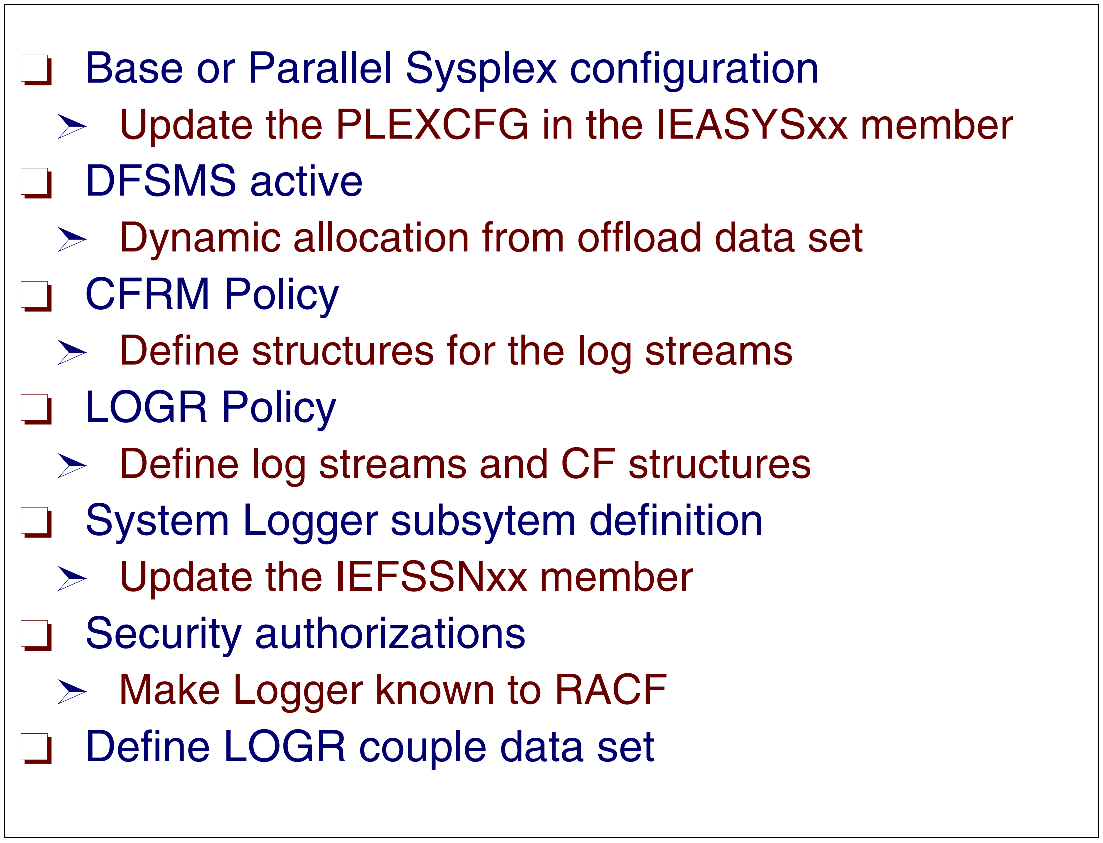
- On DASD, known as a DASD-only log stream
- In a coupling facility, known as a CF-structure log stream

The storage medium System Logger uses to hold interim log data affects how many systems can use the log stream concurrently:

- In CF list structures, exploiters on more than one z/OS in the same sysplex can write log data to the same log stream concurrently.
- In a DASD-only log stream, log data is contained in a dataspace in the z/OS system where the application is running. The dataspaces are associated with the System Logger address space, named IXGLOGR. In such a case, a DASD-only log stream can only be used by exploiters on one z/OS at a time.

By providing these capabilities using a standard interface, many applications can obtain the benefits that System Logger provides without having to develop and maintain these features themselves. This results in faster development, more functionality, and better reliability. Enhancements to System Logger, such as support for System Managed CF Structure Duplexing, become available to all System Logger exploiters as soon as they are implemented in System Logger, rather than having to wait for each exploiter to design, write, and test their own support.

2.4 Settings for System Logger services



- Base or Parallel Sysplex configuration
  - Update the PLEXCFG in the IEASYSxx member
- DFSMS active
  - Dynamic allocation from offload data set
- CFRM Policy
  - Define structures for the log streams
- LOGR Policy
  - Define log streams and CF structures
- System Logger subsytem definition
  - Update the IEFSSNxx member
- Security authorizations
  - Make Logger known to RACF
- Define LOGR couple data set

Figure 2-4 Settings for System Logger services

Base or Parallel Sysplex configuration

System Logger requires the system to be part of a sysplex, either Parallel Sysplex or monoplex. Verify that in the IEASYSxx member you have specified MULTISYSTEM or ANY in the PLEXCFG parameter.

DFSMS active

For either CF or DASD log streams it is necessary that DFSMS address space be active in z/OS. System Logger uses SMS to dynamically allocate offload and staging data sets. This address space is initialized at IPL time.

CFRM Policy

If your System Logger configuration is planning to use the coupling facility as interim medium for the log stream, you need to define these structures in the CFRM policy. The structure names in the CFRM policy must match the structure names associated with log streams in the LOGR Policy.

LOGR Policy

In this policy you define each log stream and its optional CF structure. This policy is created in the LOGR CDS using the IXCMIAPU utility. Note that each coupling facility structure that you define in the LOGR policy must also be defined in the current CFRM policy for the sysplex. Structure names in the LOGR policy must match those defined in the CFRM policy.

System Logger Subsystem Definition

You must define System Logger as a subsystem in the IEFSSNxx PARMLIB member, as follows:

```
SUBSYS SUBNAME(LOGR) INITRTN(IXGSSINT)
```

Security authorizations

You must make known the IXGLOGR address space to RACF, and give the log streams access to different classes.

Exploiters of the System Logger services must get the right permission to have access to System Logger services

LOGR Policy and CFRM Policy are formatted via the IXCMIAPU utility. You must permit users to have access to this utility and the resources that the utility accesses.

LOGR couple data set

It is necessary to format primary and secondary couple data sets for logger service via the IXCL1DU utility. Provide as input the maximum number of log streams and structure definitions you plan to define.

Update the COUPLExx member in SYS1.PARMLIB to identify the LOGR couple data set to the sysplex in the next IPL.

Make the LOGR couple data set available with an IPL or with the **SETXCF** command. There is no need to activate the policy because in the LOGR CDS there is room for just one policy.

2.5 Creating and formatting LOGR couple data set

```
//STEP1     EXEC  PGM=IXCL1DSU
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
     DEFINEDS SYSPLEX(PLEX1)
              DSN(SYS1.LOGR.CDS01) VOLSER(3380X1)
          DATA TYPE(LOGR)
               ITEM NAME(LSR) NUMBER(10)
               ITEM NAME(LSTRR) NUMBER(10)
               ITEM NAME(DSEXTENT) NUMBER(20)
     DEFINEDS SYSPLEX(PLEX1)
              DSN(SYS1.LOGR.CDS02) VOLSER(3380X2)
          DATA TYPE(LOGR)
               ITEM NAME(LSR) NUMBER(10)
               ITEM NAME(LSTRR) NUMBER(10)
               ITEM NAME(DSEXTENT) NUMBER(20)
/*
```

Figure 2-5 Creating and formatting LOGR couple data set

Creating a LOGR couple data set

Figure 2-5 shows sample JCL and commands to create and format a primary and alternate LOGR couple data sets. The LOGR couple data set must:

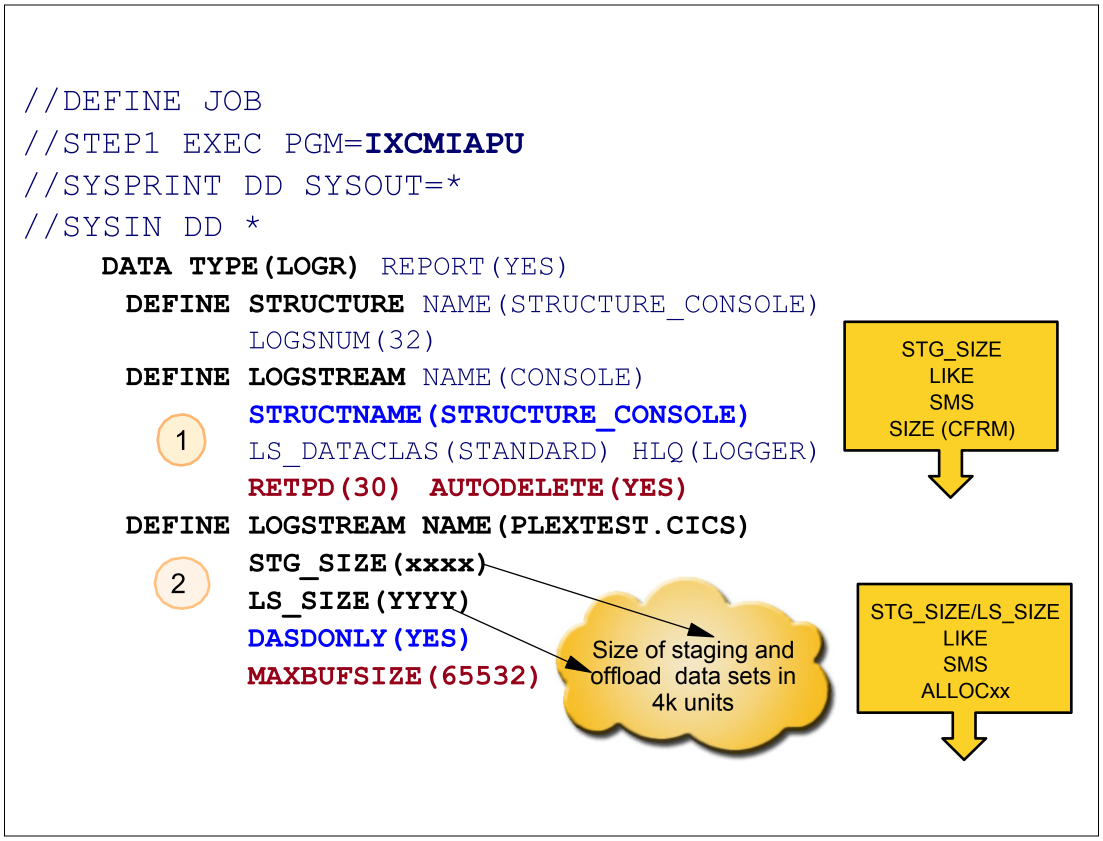
- Be created and formatted using the IXCL1DSU utility
- Be accessible by all systems in the sysplex
- Have a LOGR policy defined with the IXCMIAPU utility in the LOGR couple data set

Controlling the number of DASD log data sets

By default, each log stream is limited to a maximum of 168 offload data sets. If you have log streams that can exceed 168 offload data sets, you might consider increasing the number of directory extents available for the sysplex on the DSEXTENT parameter when formatting the LOGR couple data set. The directory extents specified in DSEXTENT allocate a *common pool* of inventory entries that can be available for any log stream in the sysplex. When a log stream exceeds the number of available slots, System Logger assigns an extent to cover 168 additional offload data sets. When all the offload data sets in a directory extent have been physically deleted, System Logger returns the directory extent record to the available pool.

Each log stream has an initial directory extent describing the first 168 offload data sets. This initial extent always belongs to the log stream and will never be reverted to the common pool even if unused.

2.6 LOGR policy definition



```
//DEFINE JOB
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
      DATA TYPE(LOGR) REPORT(YES)
        DEFINE STRUCTURE NAME(STRUCTURE_CONSOLE)
                  LOGSNUM(32)
        DEFINE LOGSTREAM NAME(CONSOLE)
                  STRUCTNAME(STRUCTURE_CONSOLE)
                  LS_DATACLAS(STANDARD) HLQ(LOGGER)
                  RETPD(30)  AUTODELETE(YES)
        DEFINE LOGSTREAM NAME(PLEXTEST.CICS)
                  STG_SIZE(xxxx)
                  LS_SIZE(YYYY)
                  DASDONLY(YES)
                  MAXBUFSIZE(65532)
```

1
2
STG_SIZE LIKE SMS SIZE (CFRM)
Size of staging and offload data sets in 4k units
STG_SIZE/LS_SIZE LIKE SMS ALLOCxx

Figure 2-6 LOGR policy definition

LOGR policy definition

Different from the other couple data sets, the LOGR CDS contains *only one* policy for the sysplex. You cannot specify the DEFINE POLICY NAME parameter in the LOGR policy, nor can you issue the `SETXCF START,POLICY` command to activate a policy.

The LOGR policy is defined with the IXCMIAPU utility in the LOGR couple data set. The LOGR policy contains log stream-related definitions, such as:

- Log stream name
- Log block size
- Retention period for the log data in the log stream
- Thresholds for offload processing
- Data to be used by System Logger when allocating staging and offload data sets:
  - SMS data class, storage class and management class
  - Data sets high level qualifier (HLQ)
  - Retention period
- The interim storage for the log stream: DASD-only or CF structure
- When the interim storage is located in the CF structure:
  - Structure name
  - Whether the structure is a candidate or not for duplexing to DASD staging data sets

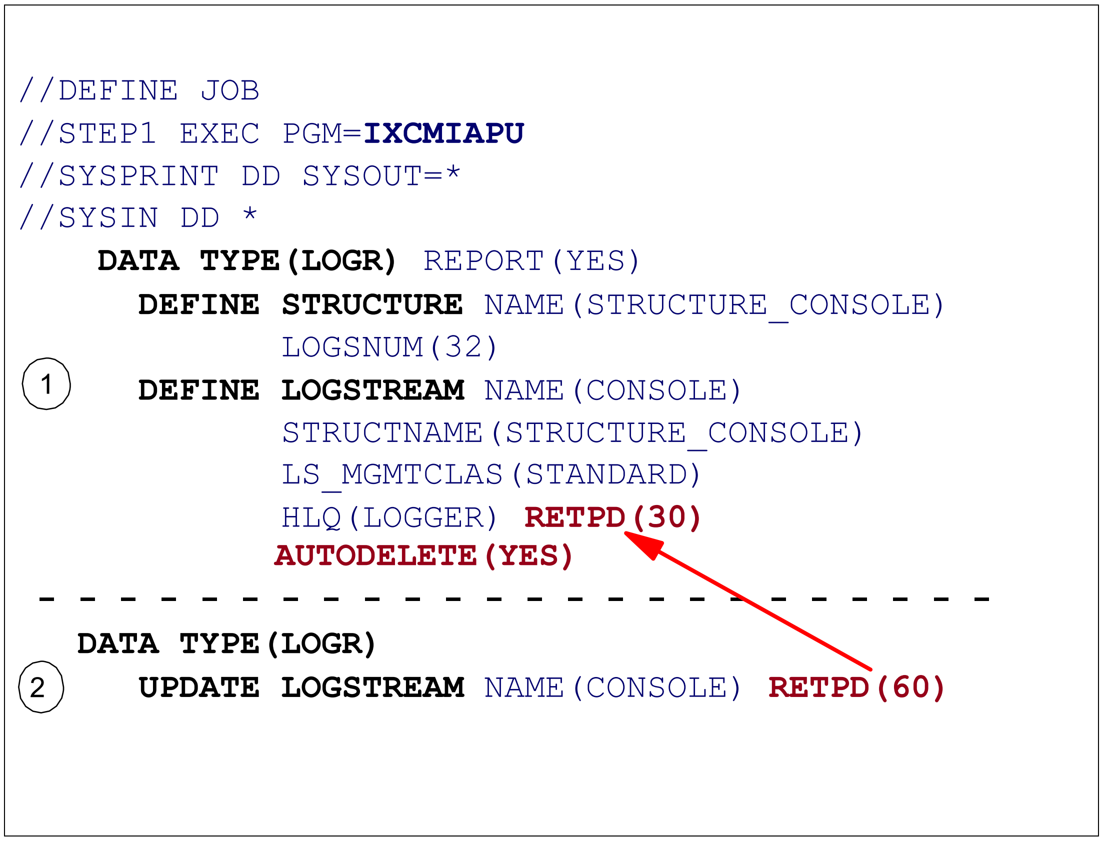
Figure 2-6 on page 124 shows JCL to run the IXCMIAPU utility to define the LOGR policy. The assigned numbers in the DEFINE LOGSTREAM keyword show the following:

1. A log stream definition to use the CF structure as interim storage
2. A DASD-only log stream as interim storage

Staging and offload data sets are allocated dynamically using the size, or model, or SMS data class specified in the policy, in that order. If nothing is specified, the installation dynamic allocation defaults are used.

Since z/OS V1R8 it is possible to rename a log stream dynamically. It allows you to bypass a "bad" log stream and continue operations. It enables you to get new work going after defining a new instance of the log stream with the original name in a timely fashion. Also, it allows you to maintain the current data in a log stream under a new name.

2.7 Managing log data



```
//DEFINE JOB
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DATA TYPE(LOGR) REPORT(YES)
      DEFINE STRUCTURE NAME(STRUCTURE_CONSOLE)
             LOGSNUM(32)
 1    DEFINE LOGSTREAM NAME(CONSOLE)
             STRUCTNAME(STRUCTURE_CONSOLE)
             LS_MGMTCLAS(STANDARD)
             HLQ(LOGGER) RETPD(30)
            AUTODELETE(YES)
 - - - - - - - - - - - - - - - - - - - - - - - -
   DATA TYPE(LOGR)
 2    UPDATE LOGSTREAM NAME(CONSOLE) RETPD(60)
```

Figure 2-7 Managing log data

Managing log data

System Logger provides support to make it easier to archive log data and manage the amount of data kept in a log stream. This applies to both coupling facility and DASD-only log streams. You can define a retention period and automatic deletion policy for each log stream.

The retention period and automatic deletion policy are specified on the RETPD and AUTODELETE parameters in a LOGR couple data set to help manage how long you keep data in the log stream and whether the data will be automatically deleted.

The RETPD and AUTODELETE parameters are used together to set up a retention period and automatic deletion for a log stream.

REPTD parameter

The RETPD parameter allows you to specify a retention period for a log stream. You specify the number of days that you want to keep data in the log stream, even if the data has been marked for deletion for the exploiter. For example, if you specify RETPD(7) in the LOGR policy for a log stream, the retention period for data in that log stream is seven days from the time the data is written to the log stream by the application. System Logger processes the retention period on a log data set basis. Once the retention period for the entire log data set has expired, the data set is eligible for deletion. System Logger may not physically delete data as soon as the retention period ends. When System Logger physically deletes the data depends on when the data set fills and what you specify on the AUTODELETE parameter.

AUTODELETE parameter

AUTODELETE(NO) means that the log data can be physically deleted only after the log data has been marked for deletion by the exploiter and after any retention period specified for the log stream has expired. AUTODELETE(NO) is the default.

AUTODELETE(YES) indicates the use of an automatic deletion policy to limit how long data is kept in the log stream. AUTODELETE(YES) means that log data can be physically deleted either when the data is marked for deletion by the exploiter or when a retention period specified for the log stream expires. Use care when specifying AUTODELETE(YES) because automatic deletion is designed to speed physical deletion of log data, which can mean deletion of data that an application needs.

UPDATE LOGSTREAM keyword

The UPDATE LOGSTREAM specification requests that an entry for a CF structure or a DASD-only log stream be updated in the LOGR policy. Except for the RETPD, AUTODELETE, and DIAG parameters, you cannot update a log stream while there are active connections (active or failed) to it.

Updating the LOGR policy

Updates made to the LOGR policy are made in the primary LOGR CDS. System Logger only uses the *policy information contained in the primary LOGR couple data set*. When switching to an alternate LOGR couple data set (making it now the primary LOGR CDS), System Logger copies the LOGR policy from the old primary couple data set into the new one. The reason for this behavior is that if you loose the primary LOGR CDS, System Logger keeps executing its functions because the policy is copied in System Logger address space.

2.8 Log stream in a coupling facility

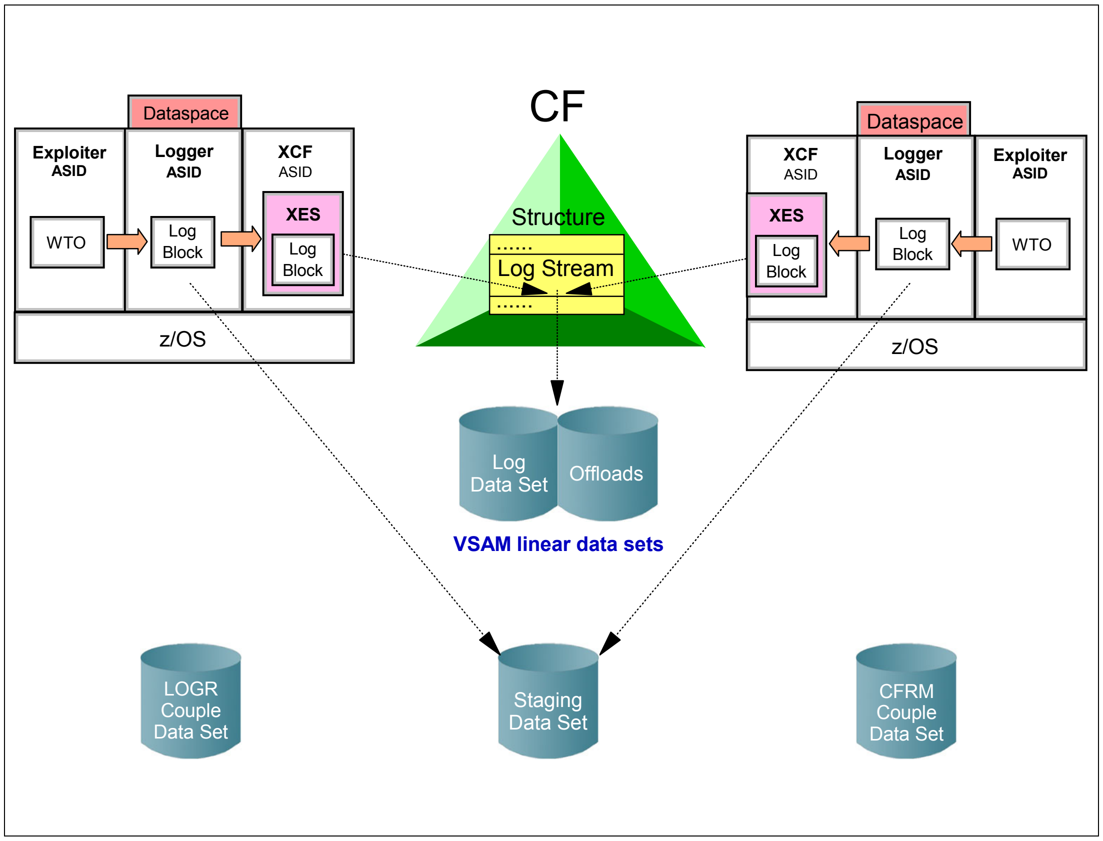


Figure 2-8 Log stream in a coupling facility

Log stream in a coupling facility

System Logger interacts with XES to connect and use the coupling facility for System Logger applications. There can be a single log stream or multiple log streams associated with a coupling facility list structure. A coupling facility log stream can contain data from multiple systems, allowing a System Logger application to merge data from systems across the sysplex.

Because the availability of log data is critical to many applications to allow successful recovery from failures, System Logger provides considerable functionality to ensure that log data is always available if needed. When System Logger writes a log block to a coupling facility log stream, it writes the log block first to a coupling facility list structure and then duplexes the log block according to the log stream definition. This describes the process when STG_DUPLEX(YES) and DUPLEXMODE(COND/UNCOND) are specified.

It is possible to specify STG_DUPLEX(YES) and DUPLEXMODE(DRXRC). With this specification System Logger duplexes the log stream data written to the coupling facility CF structure in a staging data set in an asynchronous manner using Extended Remote Copy (XRC). Refer to “System Logger XRC support” on page 132 for more on XRC and System Logger.

System Logger calls XES to update the log stream in the coupling facility. In addition, XRC is called, which will handle the update request to the staging data set. There is now a performance benefit because we no longer have to wait for the I/O completion from the staging data set. Use this option when you always want to use staging data sets for specific disaster recovery situations, but not use them for any local system log data recovery. Table 2-1 shows the conditions for duplexing. When not using System-Managed Duplexing, a single point of failure exists when the environment is vulnerable to permanent log data loss, as in the case of a CF structure and CF connector in the same CEC.

Duplexing log data

Duplexing ensures the data is still available even if the CF containing the log stream fails. Table 2-1 shows where System Logger writes the duplexed log data according to CF log stream definitions and the environment.

When the coupling facility structure space allocated for the log stream reaches the installation-defined threshold, System Logger offloads the log blocks from the coupling facility structure to VSAM linear DASD data sets, so that the coupling facility space for the log stream can be used to hold new log blocks. Figure 2-8 on page 128 shows how a coupling facility log stream spans these two levels of storage:

- The coupling facility for interim storage
- DASD log data set for offload storage

Table 2-1 System Logger duplexing control keywords

STG_DUPLEX	**Controls the log stream duplexing to staging data sets**	
	NO	Log data is duplexed only in IXCLOGR data space
	YES	Log data duplexing is controlled by DUPLEXMODE specification
DUPLEXMODE	**Specifies conditions under which log data is duplexed**	
	UNCOND	Log data is always duplexed
	COND	Log data is duplexed in a single point of failure environment
	DRXRC	Log data is always duplexed but only for specific disaster recovery purposes
LOGGERDUPLEX	**Specifies if Logger provides its own duplexing with System-Managed Duplexing (SMD) to other CF**	
	UNCOND	Logger provides its log data duplexing even with SMD
	COND	Logger provides its own log duplexing if both structure instances are in the same failure domain

System Logger and System-Managed Duplexing

The greatest benefit of System-Managed Duplexing by the System Logger comes from eliminating staging data sets as a requirement for availability. There are two major exploiters of staging data sets:

1. Resource Recovery Services (RRS)

   RRS has recommended that its data be backed by staging data sets to obtain better recoverability characteristics than using data spaces. With System-Managed Duplexing, IBM recommends a CF structure, and not DASD-only logging, to provide the ability to restart a resource manager on a different partition.

2. CICS TS

   CICS generally does not recommend the use of logger staging data sets due to the performance cost. However, there are cases when staging data sets are recommended. These are:

   – The coupling facility is volatile (no internal battery backup or UPS).

   – A coupling facility is not failure-isolated from the z/OS connector on different servers.

   – A structure failure resulted in the only copy of log data being in z/OS local storage buffers where DASD-only logging is implemented. For DASD-only log streams, staging data sets are the primary storage.

STG_DUPLEX(YES) and DUPLEXMODE(COND) for the system log and forward recovery log streams are recommended in order to cause the System Logger to automatically allocate staging data sets if the CF is not failure-isolated.

Staging data sets are also used in a disaster recovery environment to initiate a DASD remote copy of the data to the remote site.

Remember that you must also define the CF structure in the CFRM policy.

Table 2-2 Duplexed log data placement according to environment a log stream definition

System-Managed Duplexing	Single point of failure[a]	STG_DUPLEX	DUPLEXMODE	LOGGERDUPLEX
No	---	No	----	----
Yes	No	No	----	Cond
Yes	No	No	----	Uncond
Yes	No	Yes	Cond	Cond
Yes	No	Yes	Cond	Uncond
Yes	No	Yes	Uncond	Uncond
Yes	No	Yes	Uncond	Cond
Yes	Yes	Yes	Cond	Cond
No	No	Yes	Cond	----
No	Yes	Yes	----	----
No	No	Yes	Uncond	----

a. With System-Managed Duplexing, a single point of failure is when both structures are in the same failure domain.

Defining a CF structure log stream

Define a CF structure log stream in the LOGR policy, using IXCMIAPU utility. Figure 2-6 on page 124 shows a CF structure log stream definition. The first DEFINE LOGSTREAM keyword is a CF log stream. The log stream name is *console*.

The System Logger is designed to provide its own backup copy of log data written to a CF structure for recovery capability. The Logger keeps its backup copy of the structure data either in local buffers (data spaces associated with the IXGLOGR address space) or in log stream staging data sets. This allows the Logger to provide recovery of log data in the event of many types of CF, structure, or system failures. You can optionally elect to have coupling facility data duplexed to DASD staging data sets for a coupling facility log stream, as shown in Table 2-1 on page 129.

Table 2-3 shows the keywords in DEFINE or UPDATE log streams that apply only to CF structure log streams.

Table 2-3 Keywords applying only to CF structure log streams

STRUCTNAME
STG_DUPLEX
DUPLEXMODE
LOGGERDUPLEX

2.9 System Logger XRC support

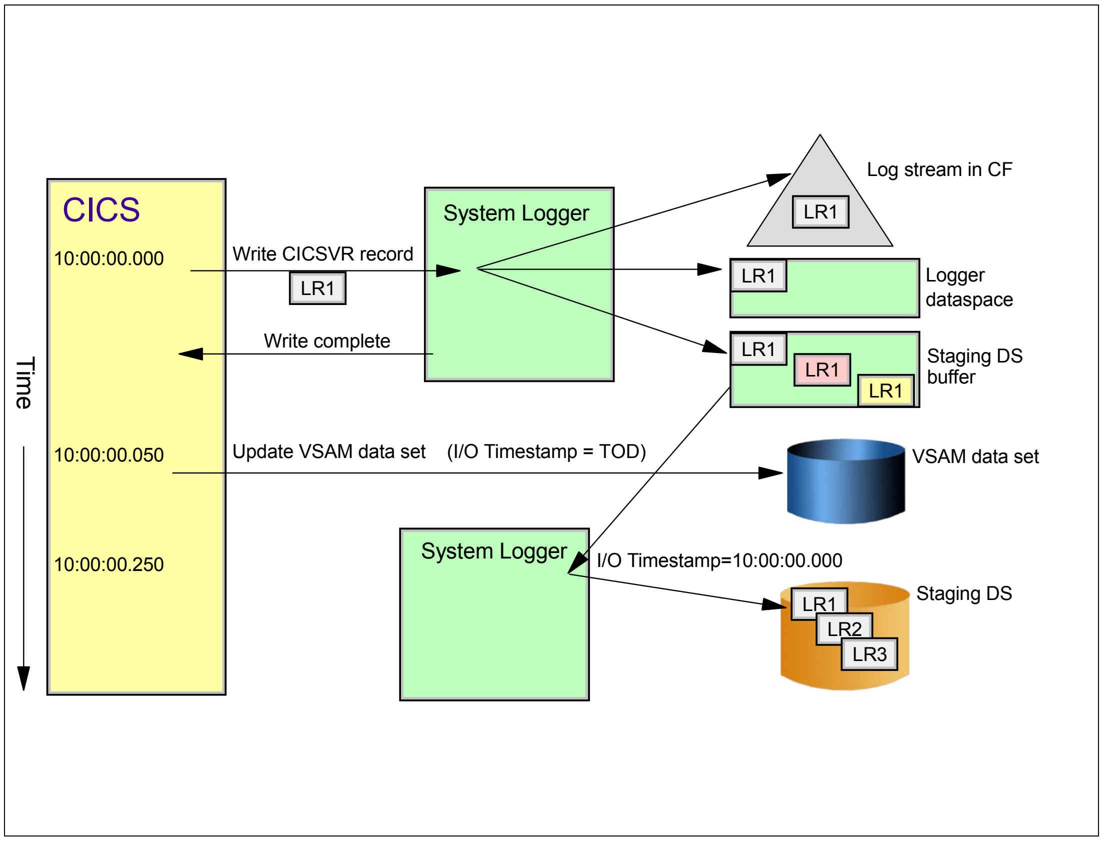


Figure 2-9 System Logger XRC support

System Logger XRC support

System Logger causes a problem when a disaster recovery topology is implemented in two sites together with XRC remote copy (asynchronous DASD remote copy). Prior to System Logger, consistency of remote disks was retained for databases and associated log files, as long as everything was mirrored. With System Logger, DB manager log records written to Logger might not reach disk (offload data sets) until many minutes after a database update has been applied. Because coupling facility is not remote copied, the remote site has the database update but not the related log records.

Staging data sets

To solve the problem, force System Logger to use staging data sets so control is not returned to DB manager until the log record has been written to disk. This ensures remote site consistency. But the use of staging data sets hurts performance for busy log streams because it is synchronous to transaction execution.

Use of staging data sets would not hurt if the I/O was not synchronous to transaction execution. If we could coordinate application of primary disk updates to the secondary disk by XRC with the contents of the staging data sets, we can ensure that remote database and log data sets are consistent. This is the function provided by XRC.

Using XRC

The remote copy must be asynchronous because the updates to the secondary disks must take place in a different sequence to the updates to the primary disks. By allowing the MVS

System Logger staging data sets to be written to asynchronously and providing the appropriate consistency group times on a recovery site, Logger and XRC provide a viable mechanism for an enterprise to use a long distance disk mirroring solution to recover log stream data written to a coupling facility structure in the event of a disaster at their primary sysplex site.

Staging data sets and DASD mirroring

Basically, System Logger exploiters continue to have a mechanism to use log stream staging data sets and DASD mirroring technology to provide remote site disaster recovery capability for coupling-facility-based log streams. The existing XRC capabilities, combined with the asynchronous writing to log stream data sets, provides a more complete method of mirroring log data written into a coupling facility structure. Peer to Peer Remote Copy (PPRC) or any non-XRC (LOGPLUS) configurations will not provide the correct environment for the proper use of DRXRC-type staging data sets.

2.10 DASD-only log streams

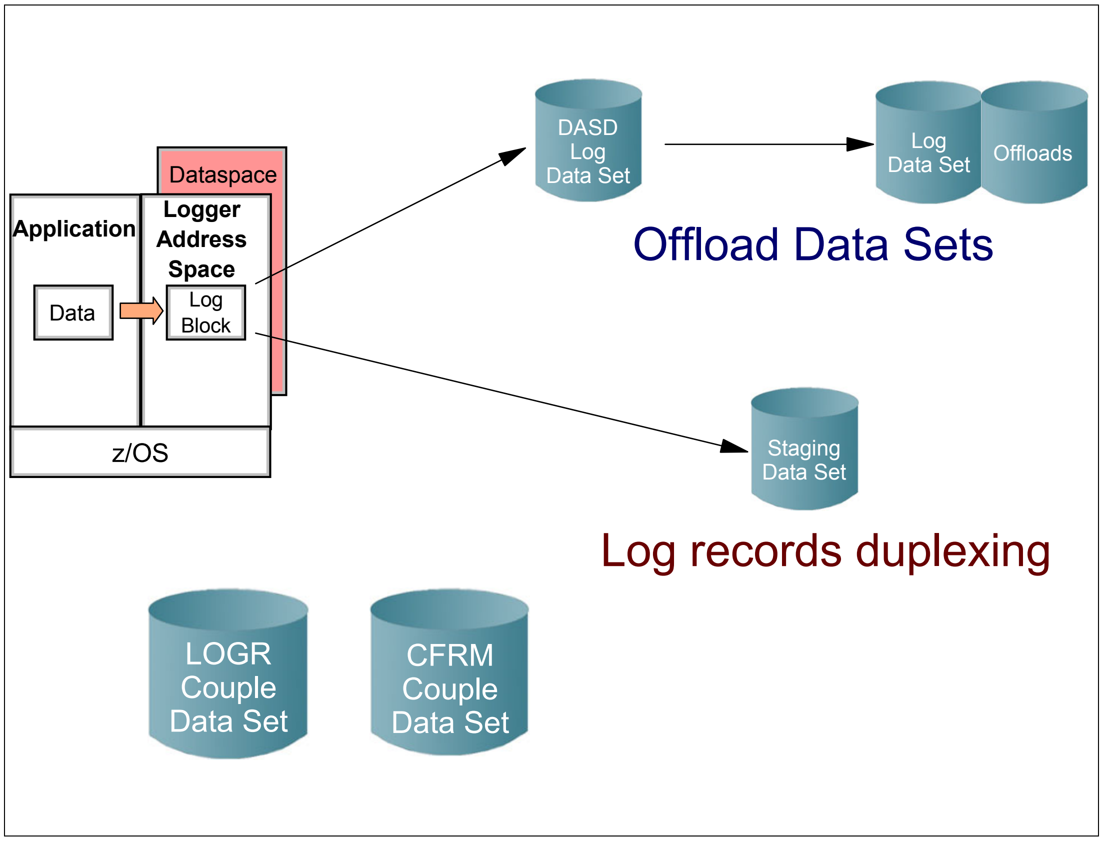


Figure 2-10 DASD-only log stream

DASD-only log stream

A DASD-only log stream has a single-system scope; only one system at a time can connect to a DASD-only log stream. Multiple applications from the same system can, however, simultaneously connect to a DASD-only log stream. A DASD-only log stream can be moved to a coupling facility log stream by updating the log stream definition in the LOGR policy.

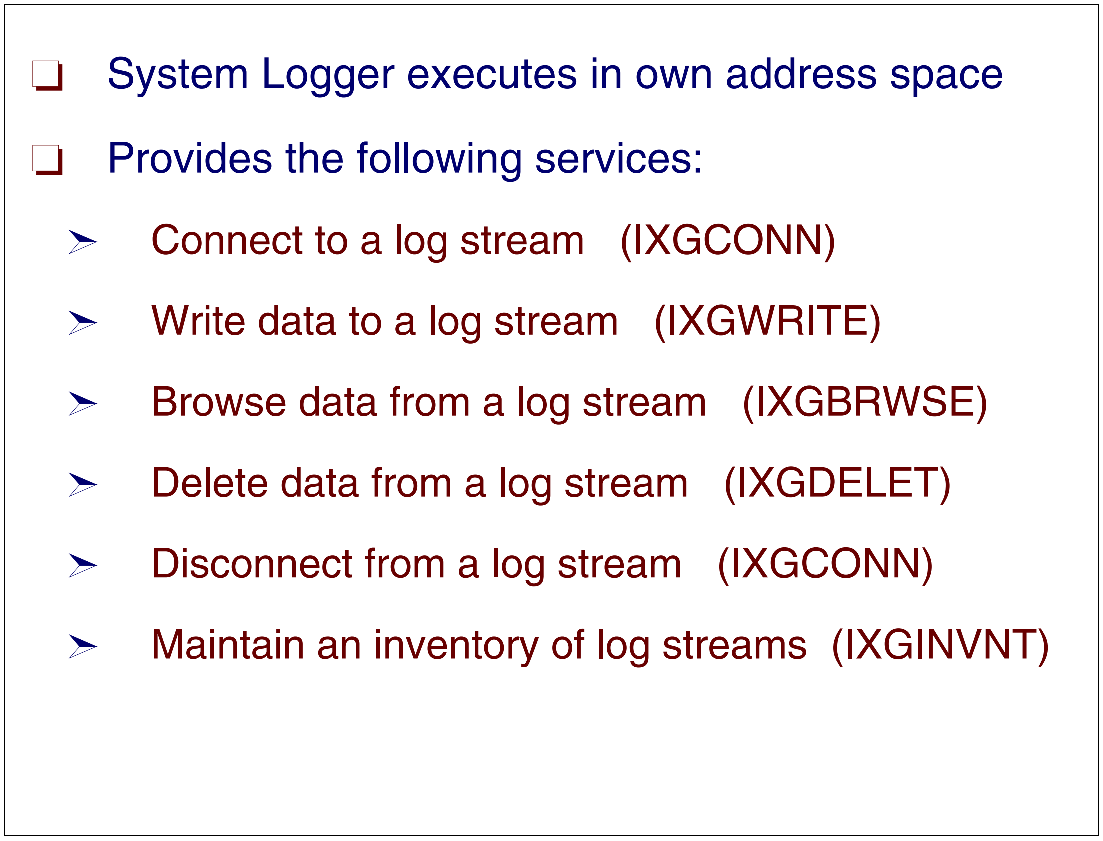
Defining a DASD-only log stream

You define an interim storage to DASD using the DASDONLY (YES) keyword in DEFINE LOGSTREAM, as shown at number 2 in Figure 2-6 on page 124. You can also specify the MAXBUFSIZE keyword to define the maximum buffer size that can be written to the DASD-only log stream. We recommend that you plan for using SMS to manage staging and offload data sets.

For DASD-only log streams, staging data sets are required. When the staging data set space allocated for the log stream reaches the installation-defined threshold, System Logger offloads the log blocks from local storage buffers to offload data sets. From a user's point of view, the actual location of the log data in the log stream is transparent.

When an System Logger application writes a log block to a DASD-only log stream, System Logger first writes the log block to the local storage buffers (IXCLOGR data spaces) and then duplexes the log block to a DASD staging data set associated with the log stream. The maximum number of connected DASD-only log streams increased from 1024 to 16384.

2.11 System Logger services



- ❑ System Logger executes in own address space
- ❑ Provides the following services:
  - ➢ Connect to a log stream (IXGCONN)
  - ➢ Write data to a log stream (IXGWRITE)
  - ➢ Browse data from a log stream (IXGBRWSE)
  - ➢ Delete data from a log stream (IXGDELET)
  - ➢ Disconnect from a log stream (IXGCONN)
  - ➢ Maintain an inventory of log streams (IXGINVNT)

Figure 2-11 System Logger services

System Logger executes in its own address space

The System Logger component resides in the IXGLOGR address space on each system in a sysplex.

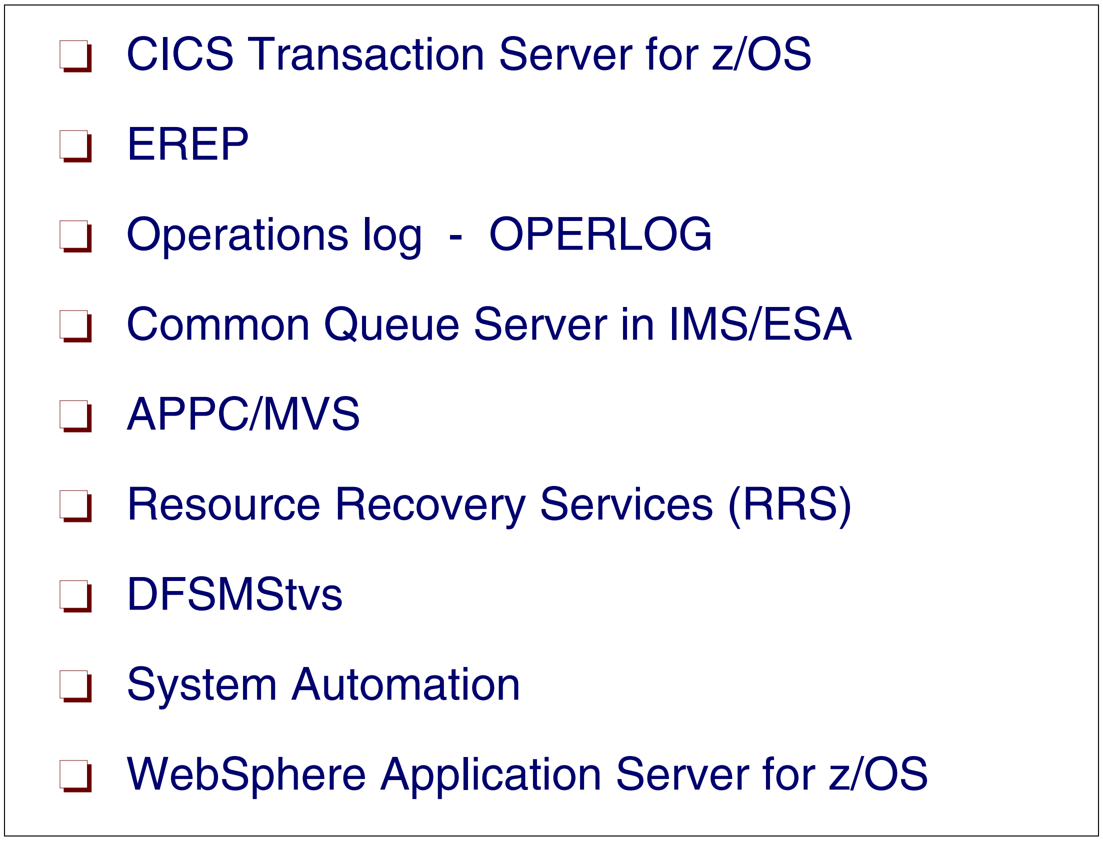
System Logger services

System Logger provides a set of functions that allows an application to update the log streams. The services are invoked through the use of assembler macros when writing a System Logger application, as follows:

- Connect to and disconnect from a log stream (IXGCONN)
- Browse a log stream (IXGBRWSE)
- Write to a log stream (IXGWRITE)
- Delete data from a log stream (IXGDELET)
- Define, update, and delete log stream and coupling facility list structure definitions in the LOGR policy (IXGINVNT service or IXCMIAPU service)

The services IXGCONN, IXGBRWSE, IXGWRITE, and IXGDELET contain parameters for both authorized and unauthorized programs. All other System Logger services and their parameters can be used by any program. We recommend that installations use Security Authorization Facility (SAF) that connects to RACF, to control access to System Logger resources such as log streams or coupling facility structures associated with log streams.

2.12 System Logger exploiters



- ❑ CICS Transaction Server for z/OS
- ❑ EREP
- ❑ Operations log - OPERLOG
- ❑ Common Queue Server in IMS/ESA
- ❑ APPC/MVS
- ❑ Resource Recovery Services (RRS)
- ❑ DFSMStvs
- ❑ System Automation
- ❑ WebSphere Application Server for z/OS

Figure 2-12 System Logger exploiters

System Logger exploiters

System Logger exploiters can be supplied by IBM, independent software vendors, or written at your installation. For each of these exploiters some information is necessary for setup and use. Some of the IBM-supplied System Logger exploiters are discussed here.

CICS Transaction Server for z/OS

CICS TS R1.0 and subsequent releases exploit the System Logger for CICS backout and VSAM forward recovery logging as well as auto-journalling and user journalling. Exploiting the System Logger for the CICS backout log provides faster emergency restart. Exploiting the System Logger for VSAM forward recovery logs and for journalling allows merging of the logged data across the Parallel Sysplex in real time. This simplifies forward recovery procedures and reduces the time to restore the VSAM files in the event of a media failure.

Logrec log stream

Logrec log stream is a z/OS System Logger application that records hardware failures, selected software errors, and selected system conditions across the sysplex. Using a logrec log stream rather than a logrec data set for each system can streamline logrec error recording.

Operations log (OPERLOG)

The operations log is a z/OS System Logger application that records and merges console messages and commands from each z/OS in a sysplex that activates OPERLOG. Use

OPERLOG rather than the system log (SYSLOG) as your hardcopy medium when you need a permanent chronological log about operating conditions and maintenance for all systems in a sysplex.

IMS common shared queues (CQS) log manager

The IMS common shared queues (CQS) log manager is a System Logger application that records the information necessary for CQS to recover structures and restart. CQS writes log records into a separate log stream for each coupling facility list structure pair that it uses. IMS CQS log manager uses coupling-facility-based log streams.

APPC/MVS

APPC/MVS is a z/OS component and uses System Logger to record events related to protected conversations. An installation-defined log stream is required for APPC/MVS to track protected conversations.

Resource Recovery Services (RRS)

Many computer resources are so critical to a company's work that the integrity of these resources must be guaranteed. If changes to the data in the resources are corrupted by a hardware or software failure, human error, or a catastrophe, the computer must be able to restore the data. These critical resources are called *protected resources* or, sometimes, recoverable resources. Resource recovery is the protection of the resources. Resource recovery consists of the protocols, often called *two-phase commit protocols*, and program interfaces that allow an application program to make consistent changes to multiple protected resources.

Recoverable Resource Management Services (RRMS) is a z/OS component that provides commit coordination or syncpoint management services. These services enable transactional capabilities in recoverable resource managers. Additionally, these services provide Protected Communication Resource Managers distributed transactional capabilities in many z/OS application environments. The Recoverable Resource Management Services consist of three parts for managing transactional work in z/OS:

- Context Services for the identification and management of work requests
- Registration Services for identifying a Resource Manager to the system
- Resource Recovery Services or RRS to provide services and commit coordination for all the protected resource managers participating in a work request's transactional scope in many of z/OS application environments

Through RRMS capabilities, extended by communications resource managers, multiple client/server platforms can have distributed transactional access to transaction server programs and to database servers.

RRS uses five log streams that are shared by the RRS within the same logging group in the sysplex. RRS supports both coupling facility log streams and DASD-only log streams. A DASD-only log stream has a single-system scope and cannot be used in a multisystem sysplex environment except in particular circumstances. For example, you might have an instance of RRS on a test image that uses its own logging group that is not shared with any other system in the sysplex. In this particular configuration, RRS can use DASD-only log streams. Usually, either for restart issues or because of the workload configuration, RRS is configured to use coupling facility log streams.

All instances of RRS in the same logging group must have access to the coupling facility structures and log stream data sets used by the RRS log streams for that logging group. This allows other RRS instances in a sysplex to access data in the event of failure of an RRS instance or system. This is required to permit resource managers to be restarted on different systems in a sysplex.

DFSMStvs

DFSMStvs uses System Logger service to log any update to a VSAM data set. With this concept DFSMStvs is able to recover VSAM data sets and to back out any process that was not successful.

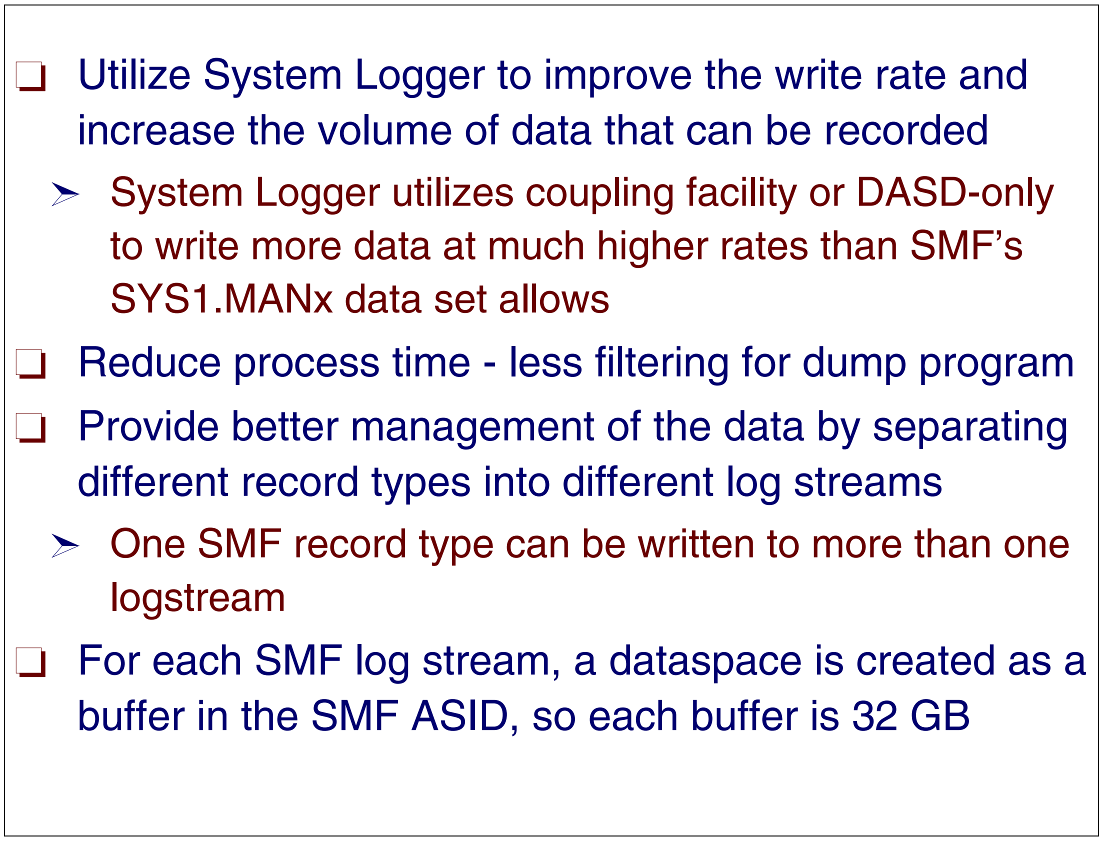
System Automation

IBM Tivoli System Automation for z/OS uses log streams to log the work items that have been processed and to store the application messages from System Automation and Netview Agents. Using log streams System Automation can trigger any event in the system.

WebSphere Application Server

WebSphere for z/OS is structured as a multiple address space application: it has a control region where the authorized code can run and multiple server regions where the application code runs. These server regions are dynamically started based on the amount of work by WLM services. All of these regions use a log stream to accumulate error messages. Each region can have its own log stream or can share the log stream with other servers.

2.13 SMF recording to logstreams



- ❑ Utilize System Logger to improve the write rate and increase the volume of data that can be recorded
  - ➢ System Logger utilizes coupling facility or DASD-only to write more data at much higher rates than SMF’s SYS1.MANx data set allows
- ❑ Reduce process time - less filtering for dump program
- ❑ Provide better management of the data by separating different record types into different log streams
  - ➢ One SMF record type can be written to more than one logstream
- ❑ For each SMF log stream, a dataspace is created as a buffer in the SMF ASID, so each buffer is 32 GB

Figure 2-13 SMF recording to logstreams

SMF recording to logstreams

You can choose to record your SMF records in logstreams by using either a coupling facility structure space (for coupling facility log streams) or DASD space for offloading the log stream data.

This new support uses the following functions and improvements:

- System Logger can improve the write rate and increase the volume of data that can be recorded.
- System Logger utilizes modern technology, such as the coupling facility and media manager, to write more data at much higher rates than the current SMF SYS1.MANx data set allows.
- Data management is improved by enhancing SMF to record its data to multiple System Logger log streams, based on record type. The record data is buffered in dataspaces, instead of the SMF address space private storage, thus allowing increased buffering capacity.
- Keywords are provided on the OUTDD keyword of the dump program that allows data to be read once and written many times. By recording different records to different log streams, SMF dump processing is also improved because a dump program per log stream can be submitted, with each being more efficient, since fewer records are read and ignored. This reduces processing time because there is less filtering needed by the dump program.

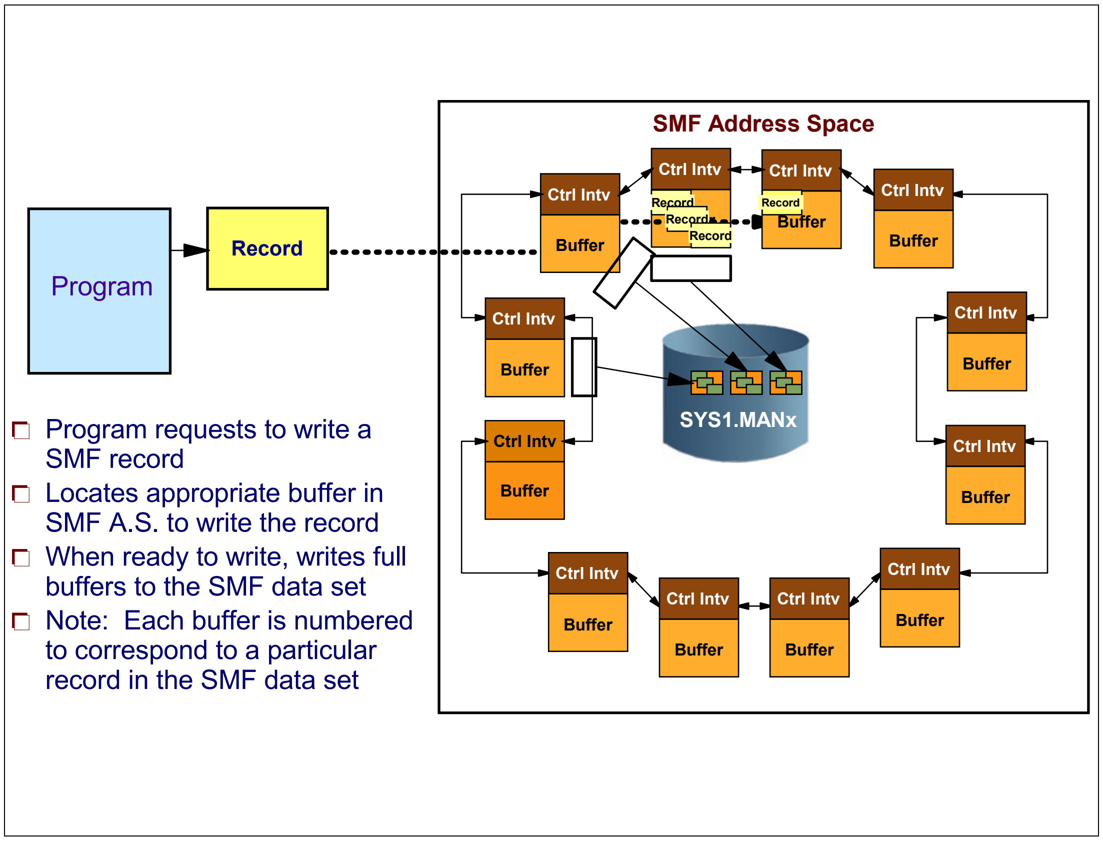
System Management Facility (SMF)

System Management Facility (SMF) collects and records system- and job-related information for an installation. SMF formats the information that it gathers into system-related records (or job-related records). System-related SMF records include information about the configuration, paging activity, and workload. Job-related records include information about the CPU time, SYSOUT activity, and data set activity of each job step, job, APPC/MVS transaction program, and TSO/E session.

To record SMF records in SMF data sets, an installation must allocate direct access space and catalog the SMF data sets. You should catalog the SMF data sets in the master catalog. SMF should have a minimum of two data sets for its use; a minimum of three SMF data sets will ensure availability.

SMF can optionally utilize the System Logger to record SMF records into log streams, which can improve the write rate and increase the volume of data that can be recorded.

2.14 SMF recording to SYS1.MANx



- Program requests to write a SMF record
- Locates appropriate buffer in SMF A.S. to write the record
- When ready to write, writes full buffers to the SMF data set
- Note: Each buffer is numbered to correspond to a particular record in the SMF data set

Figure 2-14 SMF recording to SYS1.MANx

SMF recording

When a subsystem or user program wants to write an SMF record, it invokes the SMF record macro SMFEWTM. This macro takes the user record and invokes SMF code to locate an appropriate buffer in the SMF address space and copy the data there, as shown in Figure 2-14. If the record is full, another SMF program is scheduled to locate full SMF buffers and write them to the SYS1.MANx data set. Each buffer is numbered to correspond to a particular record in the SMF data set. This allows the records to be written in any order and to place them correctly in the data set.

Although this is not shown in Figure 2-14, after all records have been written and the SYS1.MANx data set is full, SMF switches to a new SYS1.MANx data set and marks the full one as DUMP REQUIRED. That data set cannot be used again until it is dumped and cleared. Scheduling the SMF dump program must be done in a timely manner to ensure that the SMF MANx data set is returned to use as soon as possible to ensure that no data is lost due to an “all data sets full” condition.

Current implementation situations

The current implementation deficiencies are addressed by the support with SMF recording to the System Logger. Following are current situations with SMF recording to SYS1.MANx data sets:

- Several SYS1.MANx data sets can be defined. SMF records can be lost if all the SYS1.MANx data sets are filled up and not dumped, or during the SMF switch data set.

- There are situations when that system can generate a significant amount of SMF records, exceeding the system capacity to write them into the SYS1.MANx data sets.
- The current implementation of SMF can lose data if writes are being held up, when all SYS1.MANx data sets have become full or across SMF data set switch processing.
- In addition to the possibility of losing data when recording, the SMF dump program is required to read and write every record to move the data to archive data sets, where it can then be processed by other programs (such as for sorting), or by the SMF dump program again to further filter and partition the data for use. This can result in the data being read by the SMF dump program several times, as it is read and copied for use by the various exploiters of SMF data.
- The SMF records are written in any order, so they need to be sorted for post processing. In a sysplex environment, it is necessary to merge all system information to obtain a sysplex-wide analysis.
- Many installations have already set up automation to dump the SMF data sets using IFASMFDP as a postprocessor and they are satisfied with this function. This support continues the same. However, for installations that need more SMF data record write capacity, or for those whose SMF records are lost during a data set switch, it is unacceptable.

2.15 SMF on System Logger

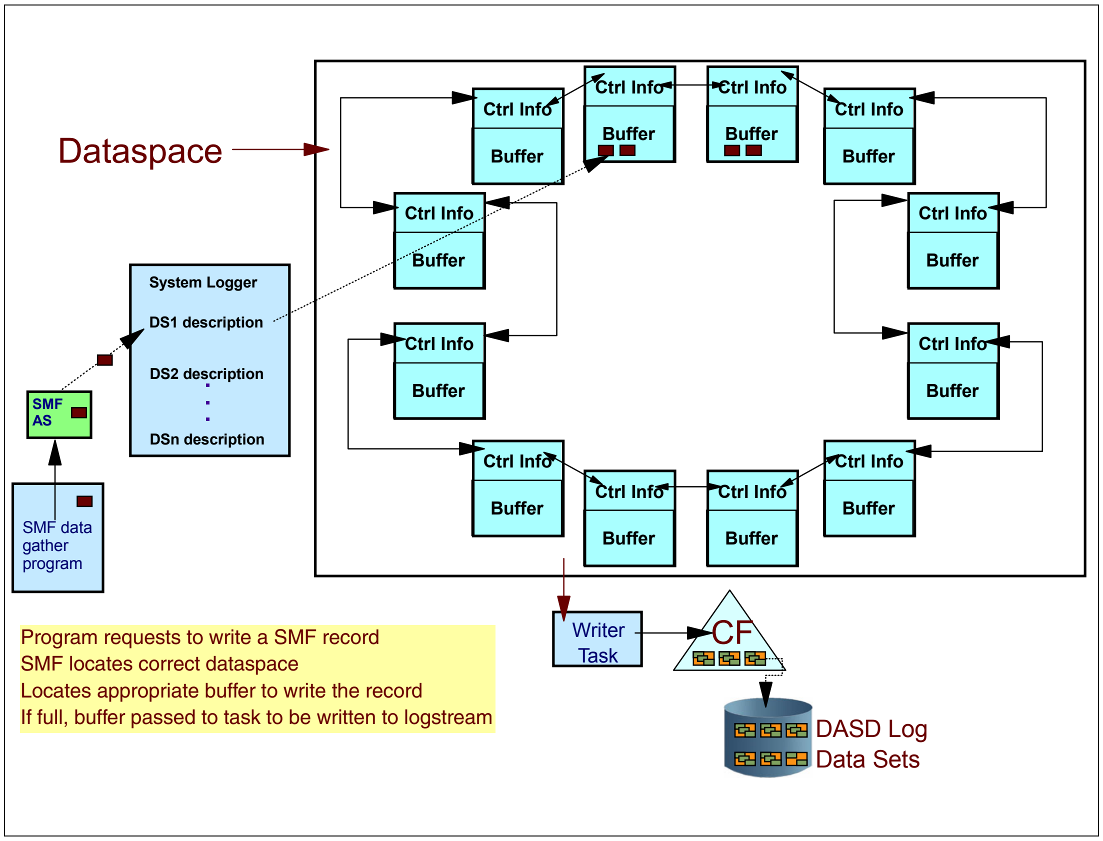

Dataspace
Ctrl Info
Buffer
System Logger
DS1 description
DS2 description
DSn description
SMF AS
SMF data gather program
Program requests to write a SMF record
SMF locates correct dataspace
Locates appropriate buffer to write the record
If full, buffer passed to task to be written to logstream
Writer Task
CF
DASD Log Data Sets

Figure 2-15 SMF on System Logger

SMF records on System Logger

An additional capability, as opposed to SYS1.MANx, is to write SMF records to log streams managed by System Logger. With this new capability you can define several log streams for several groups of SMF records. You can define one log stream to write only the RMF records types, so during postprocessing they are already isolated. When you define one log stream, you must define the SMF records type that will be written to this log stream. You can define a default log stream to write all the remaining SMF records types not defined to a specific log stream.

Note: You can create as many dataspaces as needed to create SMF log streams.

Recording to log streams

When recording to log streams, as shown in Figure 2-15, subsystems or user programs still invoke the SMFEWTM macro to take the user record and invoke SMF code. However, instead of locating a buffer in SMF private storage, SMF locates a data space corresponding to the user's record type and log stream where the record will be written. A buffer with space to hold the record is located and the record is copied there. When the record is full, a writer task is posted.

Unlike the scheduled approach in SMF data set recording, this task is already started and ready to write. In addition, writes to System Logger are done at memory-to-memory speeds, with System Logger accumulating numerous records to write out, resulting in improved access not possible with current SMF data set recording.

Using a data space to hold the records for a given log stream allows a full 2 GB of pageable memory to be used for buffering records in the event of delays in log stream writing in System Logger. This allows more data to be buffered than with SMF data set recording, which is limited to the amount of available private storage in the SMF address space.

The benefit in all this is that you can write more data faster, with more functionality. The System Logger was created to handle large volumes of data. With minimal SMF switch processing and no record numbering schemes to maintain, this eliminates the switch SMF command bottleneck.

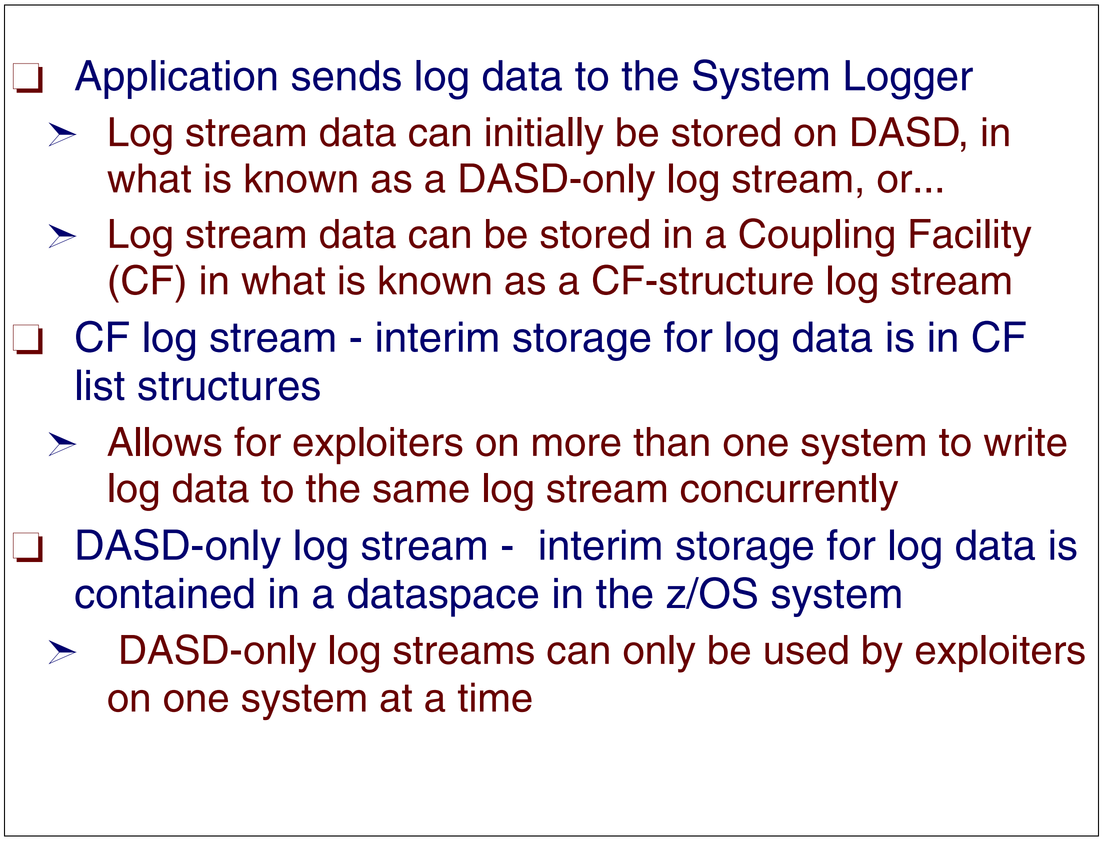
Note: The use of log streams for SMF Data is optional. Existing SYS1.MANx function continues to exist for installations satisfied with this functionality.

SMF with System Logger improvements

This new support provides the following functions and improvements:

- System Logger can improve the write rate and increase the volume of data that can be recorded.
- System Logger utilizes modern technology such as the Coupling Facility and Media Manager to write more data at much higher rates than the current SMF SYS1.MANx data set allows.
- Data management is improved by enhancing SMF to record its data to multiple System Logger log streams, based on record type. The record data is buffered in data spaces, instead of the SMF address space private storage, thus allowing increased buffering capacity.
- Keywords are provided on the OUTDD keyword of the dump program that allows data to be read one time and written many times. By recording different records to different log streams, SMF dump processing is also improved because a dump program per log stream can be submitted, with each being more efficient, because fewer records are read and ignored. This reduces processing time because there is less filtering needed by the dump program
- One SMF record type can be written to more than one log stream. By selecting log streams based on record type, the data can be partitioned at the point of its creation, resulting in less reprocessing by the SMF dump program, which means less data read per dump program instance.
- For each SMF log stream, a database is created as a buffer in the SMF ASID, so each buffer is 2 GB. Within SMF, each databases will have a task dedicated to writing its data to a particular log stream, increasing the rate at which you can record data to the System Logger.

2.16 Where System Logger stores data



- Application sends log data to the System Logger
  - Log stream data can initially be stored on DASD, in what is known as a DASD-only log stream, or...
  - Log stream data can be stored in a Coupling Facility (CF) in what is known as a CF-structure log stream
- CF log stream - interim storage for log data is in CF list structures
  - Allows for exploiters on more than one system to write log data to the same log stream concurrently
- DASD-only log stream - interim storage for log data is contained in a dataspace in the z/OS system
  - DASD-only log streams can only be used by exploiters on one system at a time

Figure 2-16 System Logger log streams

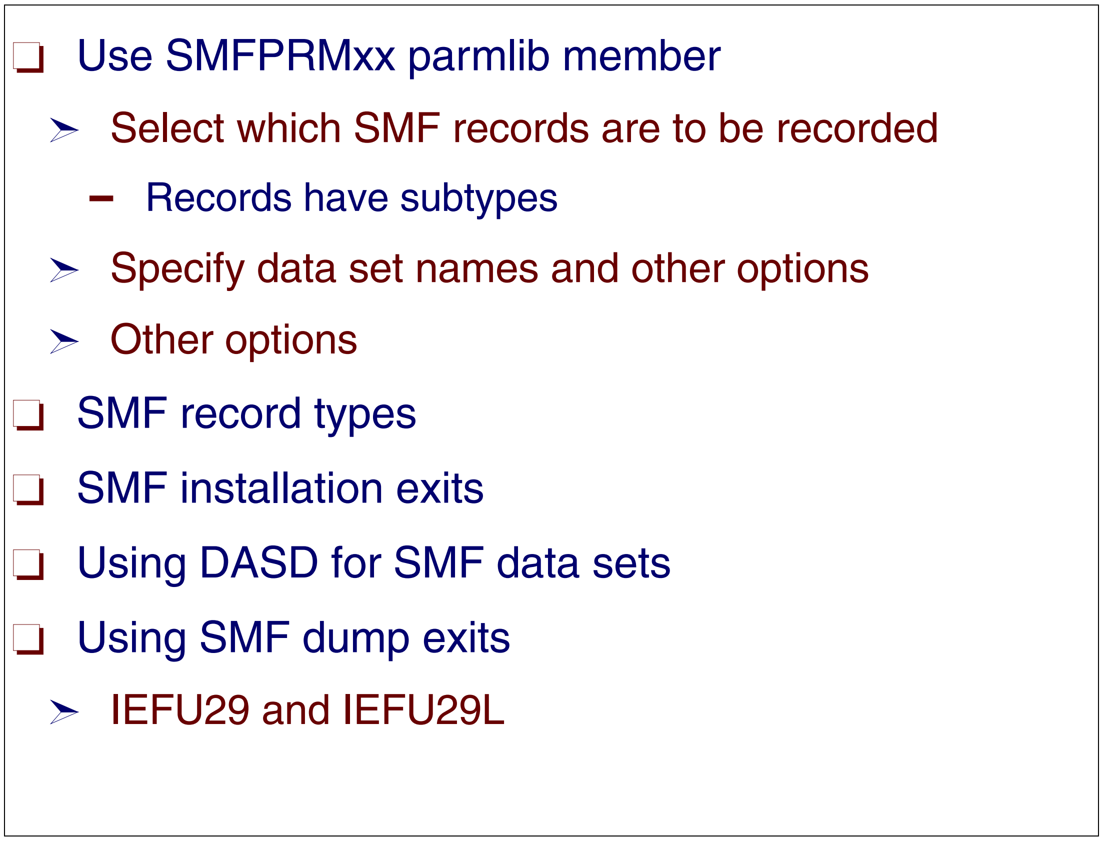
Where System Logger stores its data

When an application passes log data to the System Logger, the data can initially be stored on DASD, in what is known as a DASD-only log stream. Alternatively, it can be stored in a Coupling Facility (CF) in what is known as a CF-Structure log stream. The major differences between these two types of log stream configurations are the storage medium System Logger uses to hold interim log data, and how many systems can use the log stream concurrently, as explained here.

- In a CF log stream, interim storage for log data is in CF list structures. This type of log stream supports the ability for exploiters on more than one system to write log data to the same log stream concurrently.
- In a DASD-only log stream, interim storage for log data is contained in a data space in the z/OS system. The data spaces are associated with the System Logger address space, IXGLOGR. DASD-only log streams can only be used by exploiters on one system at a time.

Note: Your installation can use just Coupling Facility log streams, just DASD-only log streams, or a combination of both types of log streams. The requirements and preparation steps for the two types of log streams are somewhat different. See “Setting Up the System Logger Configuration” in *z/OS MVS Programming: Authorized Assembler Services Guide*, SA22-7608 and *Systems Programmer’s Guide to: z/OS System Logger,* SG24-6898.

2.17 Customizing SMF



- Use SMFPRMxx parmlib member
  - Select which SMF records are to be recorded
    - Records have subtypes
  - Specify data set names and other options
  - Other options
- SMF record types
- SMF installation exits
- Using DASD for SMF data sets
- Using SMF dump exits
  - IEFU29 and IEFU29L

Figure 2-17 Customizing SMF

Customizing SMF

Setting up SMF requires your installation to decide what kind of records it wants SMF to produce or what information it wants SMF to gather. Then you can make decisions about how to set up SMF to meet these requirements.

An installation has several ways to customize SMF to meet its needs:

- SMF parameters on the SMFPRMxx parmlib member
- Installation-written exit routines
- Operator commands

SMPRMxx parmlib member

The SMF parmlib parameters allow you to perform the following tasks:

- Record status changes
- Pass data to a subsystem
- Select specific records
- Specify data set names
- Specify the system identifier to be used in all SMF records
- Select specific subtypes
- Perform interval accounting
- Collect SMF statics
- Perform TSO/E command accounting
- Perform started task accounting

SMF record types

SMF records are selected by specifying either the type desired (or the types not desired) with the TYPE or NOTYPE option of the SYS or SUBSYS parmlib parameter. If any one of record types 14, 15, 17, 62, 63, 64, 67, or 68 is specified with the TYPE option, data is collected for *all* records. Likewise, if either record type 19 or 69 is specified with the TYPE option, data is collected for *both* records. However, only those records that are selected by TYPE or NOTYPE request are written to the SMF data set.

The TYPE option of the SYS and SUBSYS parameter provides inner parentheses to indicate the subtypes to be collected. If subtype selection is not used, the default is all subtypes. The following example illustrates how the TYPE option is to be used to record subtypes 1, 2, 3, 5, 6, 7, and 8 for the type 30:

```
SYS(TYPE(30(1:3,5:8))) or SUBSYS(STC,TYPE(30(1:3,5:8)))
```

SMF installation exits

Installation-written exit routines IEFU83, IEFU84, and IEFU85 (SMF writer) and IEFACTRT (termination) can control which records are to be written to the SMF data set. After inspecting an SMF record, these routines return a code to the system indicating whether the record is to be written to the SMF data sets.

SMF data sets

To record SMF records in SMF data sets, an installation can allocate direct access space and catalog the SMF data sets. A good practice is to catalog the SMF data sets in the master catalog. SMF is to have a minimum of two data sets for its use, and it is more desirable to run with a minimum of three SMF data sets to ensure availability.

When selecting DASD to handle the volume of data that SMF generates, if the I/O rate for a device is too slow, SMF places the data it generates in buffers. The buffers will eventually fill, which can result in lost data. Several factors, such as the specific system configuration, the amount of SMF data to be written, the size of SMF buffers (the control interval size), and your installation's report program requirements, determine which device type is most efficient for a particular installation.

SMF dump exits (IEFU29)

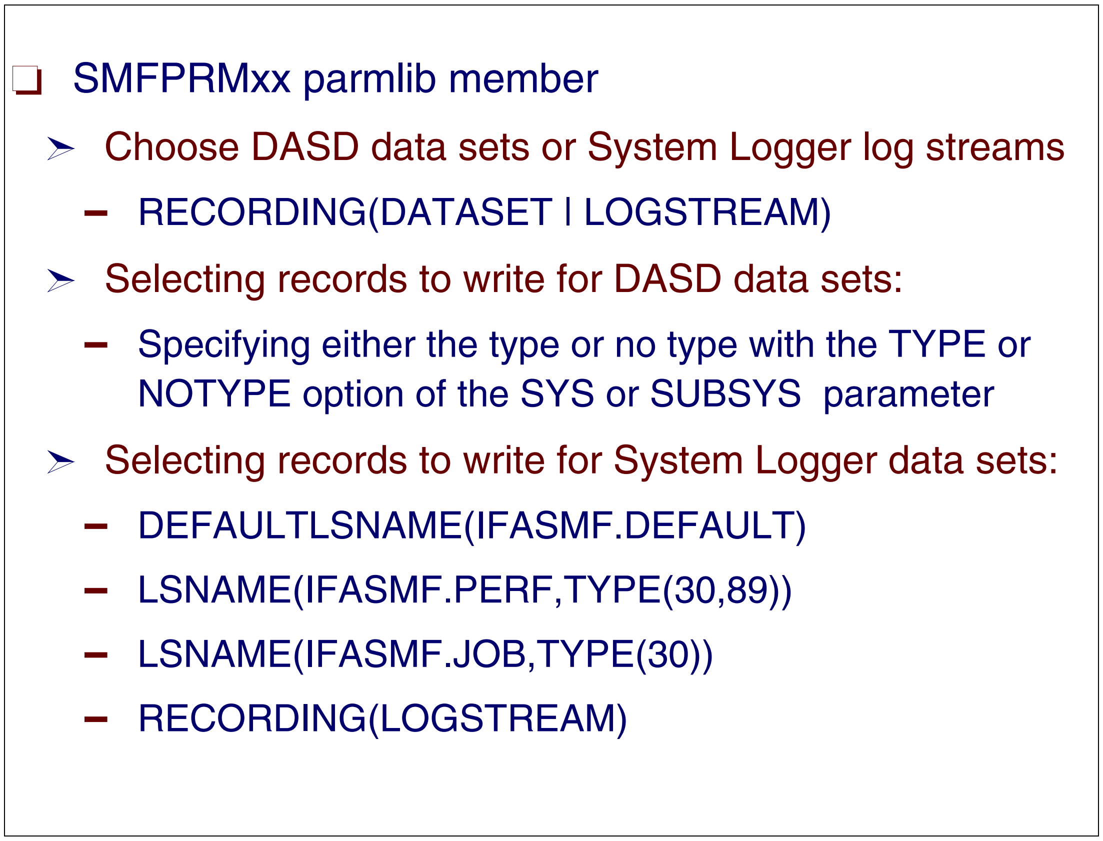
The SMF dump exit IEFU29 is invoked when the current recording data set cannot hold any more records because the SMF writer routine automatically switches recording from the active SMF data set to an empty SMF data set. This exit is also invoked when the writer switches recording data sets as a result of the `SWITCH SMF` command. A return code from this exit routine indicates whether a message that the SMF data set requires dumping is to be suppressed or not.

To allow the system to invoke IEFU29, define the exit in the SMF parmlib member (SMFPRMxx). Specify IEFU29 on the EXITS option of the SUBSYS parameter for the STC subsystem. If your installation chooses not to define a SUBSYS parameter for STC, you can specify IEFU29 on the EXITS option of the SYS parameter.

SMF dump exits (IEFU29L)

The SMF dump exit IEFU29L allows you to initiate the archiving of SMF data from a log stream. IEFU29L is invoked using the `SWITCH SMF` command. You can use the system logger to manage how long you retain the SMF log stream data and to automatically offload the log stream data to VSAM linear DASD data sets, so you might not need to use IEFU29L to drive archiving the SMF log stream data.

2.18 Which SMF records to record



- SMFPRMxx parmlib member
  - Choose DASD data sets or System Logger log streams
    - RECORDING(DATASET | LOGSTREAM)
  - Selecting records to write for DASD data sets:
    - Specifying either the type or no type with the TYPE or NOTYPE option of the SYS or SUBSYS parameter
  - Selecting records to write for System Logger data sets:
    - DEFAULTLSNAME(IFASMF.DEFAULT)
    - LSNAME(IFASMF.PERF,TYPE(30,89))
    - LSNAME(IFASMF.JOB,TYPE(30))
    - RECORDING(LOGSTREAM)

Figure 2-18 SMF records to record

Select DASD data sets or System Logger log streams

You must specify whether you want to write SMF records to MANx data sets or log streams. You can specify both SMF data sets and log streams in one SMFPRMxx member and then set either RECORDING(DATASET) or RECORDING(LOGSTREAM) on the **SETSMF** command to switch between data set and log stream recording. The default is: RECORDING(DATASET).

SMF records to record for DASD data sets

You can select the SMF records you want to write by specifying either the type desired (or the types not desired) with the TYPE or NOTYPE option of the SYS or SUBSYS parmlib parameter. If any one of record types 14, 15, 17, 62, 63, 64, 67, or 68 is specified with the TYPE option, data is collected for all of those record types. However, only records selected by a TYPE or NOTYPE request are written to the SMF data set, as shown here.

```
TYPE    {aa,bb(cc)       }
NOTYPE ({aa,bb:zz        })
        {aa,dd(cc:yy),...}
        {aa,bb(cc,...)   }
```

TYPE This specifies the SMF record types and subtypes that SMF is to collect. aa, bb, dd, and zz are the decimal notations for each SMF record type. cc and yy are the decimal notations for the SMF record subtypes. A colon (:) indicates the range of SMF record types (bb through zz) to be recorded or the range of subtypes (cc through yy for SMF record type dd) to be recorded.You can select SMF record subtypes on all SMF record types, and on user records.

NOTYPE This specifies that SMF is to collect all SMF record types and subtypes except those specified. aa, bb, and zz are the decimal notations for each SMF record type. cc and yy are the decimal notations for each subtype. A colon indicates the range of SMF record types (bb through zz) or the range of subtypes (cc through yy for SMF record dd) that are not to be recorded.

When RECORDING(DATASET) is specified, the default is SYS1.MANX and SYS1.MANY.

SMF records to record for System Logger data sets

The SMFPRMxx parmlib member parameters to use System Logger recording are listed here.

```
DEFAULTLSNAME(logstreamname)
LSNAME (logstreamname,TYPE({aa,bb}|{aa,bb:zz} | {aa,bb:zz,...}))
RECORDING(LOGSTREAM)
```

DEFAULTLSNAME(logstreamname)

This optional parameter specifies the default log stream name you want to use when you write SMF records to a log stream, except for the record types specified on LSNAME parameters. When you specify DEFAULTLSNAME (and do not override it with the LSNAME parameter), records will be queued and written to the default log stream name specified; for example:

```
DEFAULTLSNAME(IFASMF.DEFAULT)
```

LSNAME(logstreamname,TYPE({aa,bb}|{aa,bb:zz} | {aa,bb:zz,...}))

This optional parameter allows you to specify the log stream in which you want to record particular SMF record types on the TYPE subparameter.

The log stream name (logstreamname) must be composed as shown here.

- The first seven characters must be IFASMF. (which is the qualifier).
- You must have a minimum of 8 characters.
- You must have a maximum of 26 characters.

Note: All log stream names (logstreamname) must have the first name qualifier IFASMF. of seven characters in length.

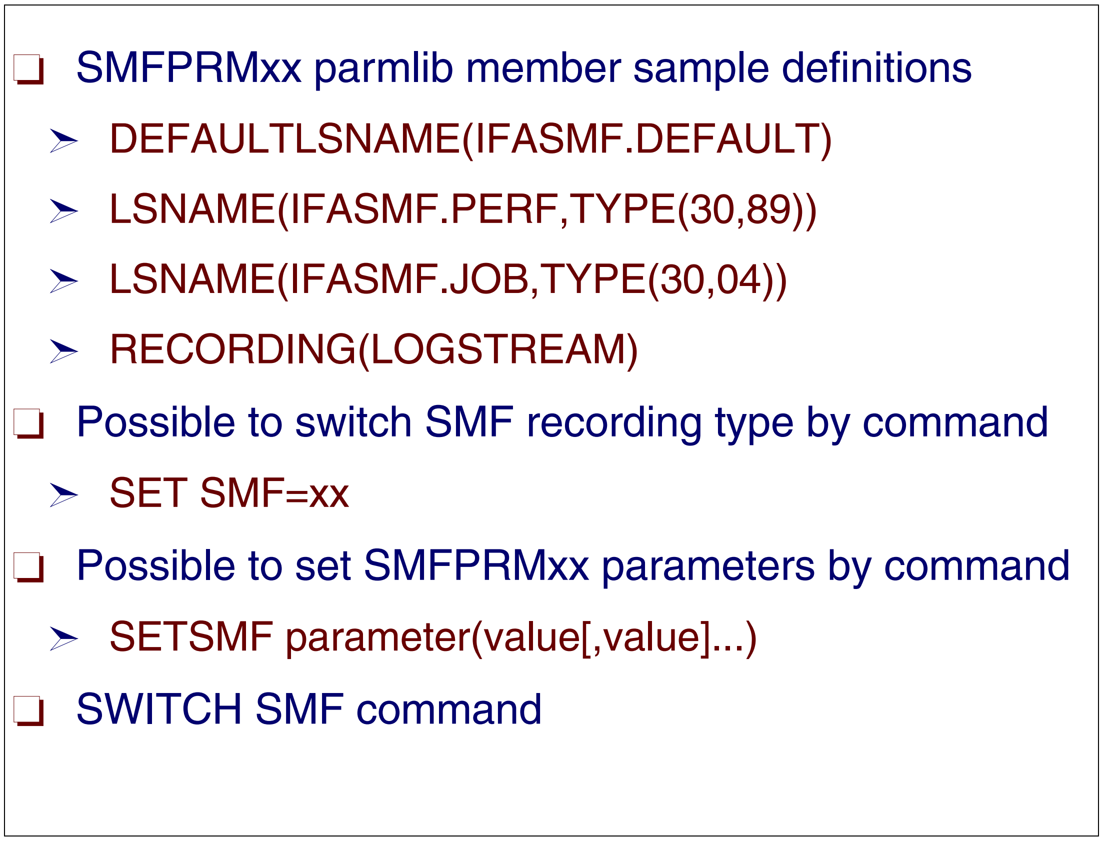
The TYPE operand specifies the SMF record types that SMF is to collect to the specified log stream on the LSNAME parameter. aa, bb, and zz are the decimal notations for each SMF record type. You cannot specify subtypes on the TYPE subparameter for LSNAME. A colon indicates the range of SMF record types (bb through zz) to be recorded.

Value Range: 0-255 (SMF record types)

Default: TYPE (0:255) (all types)

When RECORDING(LOGSTREAM) is specified, there is no default.

2.19 System Logger log streams



- SMFPRMxx parmlib member sample definitions
  - DEFAULTLSNAME(IFASMF.DEFAULT)
  - LSNAME(IFASMF.PERF,TYPE(30,89))
  - LSNAME(IFASMF.JOB,TYPE(30,04))
  - RECORDING(LOGSTREAM)
- Possible to switch SMF recording type by command
  - SET SMF=xx
- Possible to set SMFPRMxx parameters by command
  - SETSMF parameter(value[,value]...)
- SWITCH SMF command

Figure 2-19 System Logger log streams and commands

System Logger log streams

If you specify the same record type on two or more different LSNAME parameters, the system writes the record to all specified log streams. For example, you can have an SMFPRMxx parmlib member with the following contents:

```
DEFAULTLSNAME(IFASMF.DEFAULT)
LSNAME(IFASMF.PERF,TYPE(30,89))
LSNAME(IFASMF.JOB,TYPE(30))
RECORDING(LOGSTREAM)
```

These definitions allow you to collect job-related SMF data in the JOB log stream, and performance-related SMF data in the PERF log stream. Record type 30 fits into both categories, so you can specify that it is written to both log streams.

Note: This arrangement can result in duplicate records being recorded because each LSNAME statement with the same record type (30) is written into another log stream.

```
DEFAULTLSNAME(IFASMF.DEFAULT)
LSNAME(IFASMF.PERF,TYPE(30,89))
RECORDING(LOGSTREAM)
```

These definitions result in record types 30 and 89 going to log stream IFASMF.PERF. All other record types will go to default log stream IFASMF.DEFAULT.

Using the SET SMF command

This command can be used in several ways, as explained here.

- To switch from using SYS1.MAnx recording to System Logger recording

  Change the RECORDING parameter in a new SMFPRMxx parameter, and then use the **SET SMF=xx** command to switch the system to using that SMFPRMxx parmlib member.

Note: To stop recording to a particular SMF log stream, create a new SMFPRMxx parmlib member and remove the log stream you no longer want to use. Then issue the **SET SMF=xx** command to switch to using that SMFPRMxx parmlib member.

- To switch your SMF recording method

  Issue the **SETSMF RECORDING(DATASET | LOGSTREAM)** command to switch to the method you prefer. Note that this method requires PROMPT(LIST) or PROMPT(ALL) be specified in the active SMFPRMxx parmlib member to allow use of the SETSMF command, for example, if your current mode is SYS1.MANxx recording or option DATASET. Then issue the following command to start doing System Logger recording:

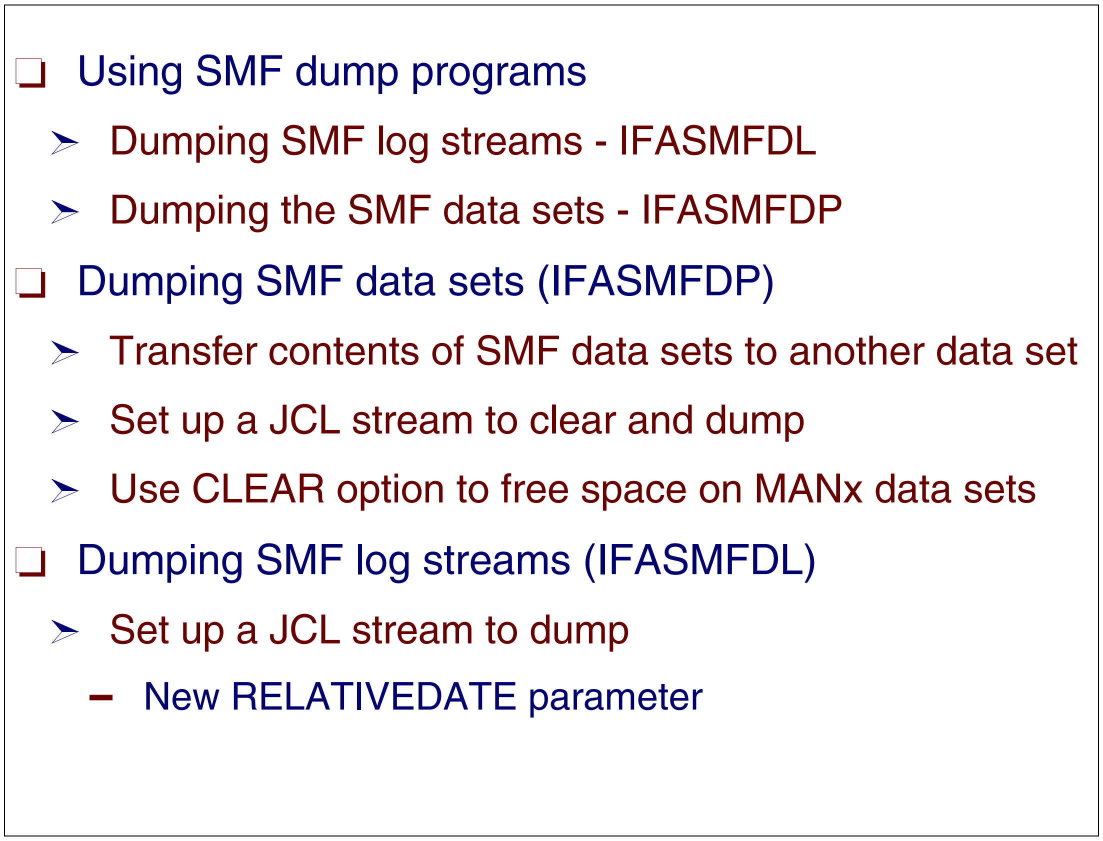
  ```
  SETSMF RECORDING(LOGSTREAM)
  ```

SWITCH SMF command

If you are using SMF data sets to record SMF records, the **SWITCH SMF** command manually switches the recording of SMF data from one data set to another. The **SWITCH SMF** command also passes control to the IEFU29 dump exit, if one exists.

If you are using log streams to record SMF records, the **SWITCH SMF** command writes data from an SMF buffer to a log stream in preparation for running the dump program to dump SMF data. The **SWITCH SMF** command also passes control to the IEFU29L dump exit, if one exists.

2.20 Dumping SMF data sets



- ❑ Using SMF dump programs
  - ➢ Dumping SMF log streams - IFASMFDL
  - ➢ Dumping the SMF data sets - IFASMFDP
- ❑ Dumping SMF data sets (IFASMFDP)
  - ➢ Transfer contents of SMF data sets to another data set
  - ➢ Set up a JCL stream to clear and dump
  - ➢ Use CLEAR option to free space on MANx data sets
- ❑ Dumping SMF log streams (IFASMFDL)
  - ➢ Set up a JCL stream to dump
    - – New RELATIVEDATE parameter

Figure 2-20 Dumping SMF data sets

Dumping SMF data sets (IFASMFDP)

When notified by the system that a full data set needs to be dumped, use the SMF dump program (IFASMFDP) to transfer the contents of the full SMF data set to another data set, and to clear the dumped data set so that SMF can use it again for recording data. The dump program copies the input data sets to the output data sets. During the copy process, the SMF data set dump program creates two SMF records and writes them to every output data set.

Note: SMF data sets cannot be shared across systems. Avoid having the SMF data set dump program be run from one system in an attempt to clear an SMF data set used by another system. The SMF dump program allows the installation to route various records to separate files and produce a summary activity report. Figure 2-21 shows a sample job stream.

```
//STEP1   EXEC PGM=IFASMFDP,REGION=4M
    //DUMPIN   DD  DSN=SYS1.SC42.MAN1,DISP=SHR
    //DUMPALL  DD DISP=(,CATLG),DSN=SMFDATA.BACKUP(+1),UNIT=SYSALLDA,VOL=SER=SMF002,
    //         SPACE=(CYL,(30,10),RLSE),DCB=SMFDATA.ALLRECS.GDGMODEL
    //SYSIN    DD  *
    INDD(DUMPIN,OPTIONS(ALL))
    OUTDD(DUMPALL,TYPE(000:255))
```

Figure 2-21 IFASMFD job stream to dump and clear the SMF data set

In the IFASMFD job stream to dump and clear the SMF data set shown in Figure 2-21, the **INDD(DUMPIN,OPTIONS(ALL))** parameter indicates that the SMF data set will be dumped and cleared. To process only a DUMP request, set **OPTIONS(DUMP)**. To process only a CLEAR request, set **OPTIONS(CLEAR)**. The **OUTDD(DUMPALL,TYPE(000:255))** parameter indicates that SMF records in the range of 000 to 255 will be dumped (that is, everything will be dumped). If you only want to dump TYPE30 Subtype 1, 3, 4 and 5, code **TYPE(30(1,3:5))**. Alternatively, if you do not want to dump these records, code **NOTYPE(30(1,3:5))**.

Note: The way you dump SMF data depends on whether you use log stream recording or SMF data set recording. The dump programs for both SMF log streams (IFASMFDL) and for SMF data sets (IFASMFDP) support the same exits.

Dumping to log streams (IFASMFDL)

To dump an SMF log stream to archive the data to a permanent medium such as tape, or so that existing user-written analysis routines can run against the dump data sets, you can dump SMF log stream data by issuing the **SWITCH SMF** command. This command first dumps the SMF data in the buffers out to the log streams, and then passes control to the IEFU29L SMF log stream dump exit.

Operators can use the SMF log stream dump program IFASMFDL to dump the specified log stream data to dump data sets. The SMF log stream dump program, shown in Figure 2-22, dumps the contents of one or more log streams to sequential data sets on either tape or direct access devices. The SMF log stream dump program allows the installation to route various records to separate files and produce a summary activity report.

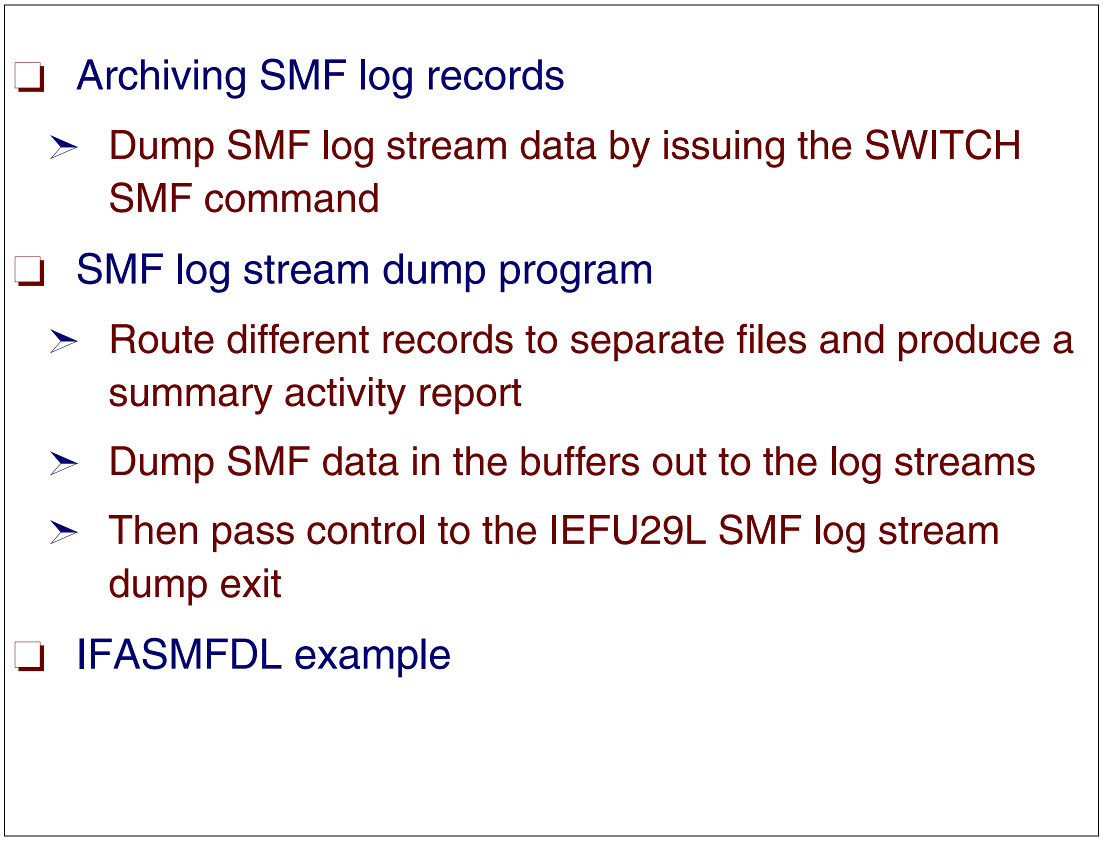
```
//DUMPX     JOB   MSGLEVEL=1
//STEP1     EXEC  PGM=IFASMFDL
//OUTDD1    DD    DSN=BASCAR.LSDEFLT.RECOR66,DISP=(NEW,CATLG,DELETE),
//                UNIT=3390,
//     SPACE=(CYL,(1,1),RLSE),DCB=(LRECL=32760,RECFM=VBS,BLKSIZE=0)
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
 LSNAME(IFASMF.MULTSYS.STREAM1,OPTIONS(DUMP))
 OUTDD(OUTDD1,TYPE(0:255))
  RELATIVEDATE (BYDAY,3,3)
  WEEKSTART (MON)
 SID(SY1)
/*
```

Figure 2-22 SMF log stream dump program JCL

Figure 2-22 requests records to be selected from the log stream that were recorded starting from three days before the current date, and ending at the third day, counting from that starting day. To illustrate further, if the JCL executes on Monday, July 28, 2008 (Julian date 2008.210), the data in the log stream from Friday July 25th 2008 (Julian date 2008.207) to Sunday July 27th (Julian date 2008.209) will be dumped.

Note: The RELATIVEDATE parameter is new with z/OS V1R11 and specifies a date range based on the current day to be selected by the IFASMFDL program. The start date of the date range is calculated by going backward *n* time units measured by day, week, or month from the current day, week, or month. The end date of the date range is calculated by moving *x* time units forward from the start date. You can use the RELATIVEDATE option instead of the DATE option; it is not compatible with the DATE parameter.

2.21 Dumping SMF records - log streams



- Archiving SMF log records
  - Dump SMF log stream data by issuing the SWITCH SMF command
- SMF log stream dump program
  - Route different records to separate files and produce a summary activity report
  - Dump SMF data in the buffers out to the log streams
  - Then pass control to the IEFU29L SMF log stream dump exit
- IFASMFDL example

Figure 2-23 Dumping log streams records

Dumping log stream records to archive

To dump an SMF log stream to archive the data to a permanent medium such as tape, or so that existing user written analysis routines can run against the dump data sets, you can dump SMF log stream data by issuing the **SWITCH SMF** command. This command first dumps the SMF data in the buffers out to the log streams, and then passes control to the IEFU29L SMF log stream dump exit. The operator can use the SMF log stream dump program IFASMFDL to dump the specified log stream data to dump data sets.

SMF log stream dump program

The SMF log stream dump program dumps the contents of one or more log streams to sequential data sets on either tape or direct access devices. The SMF log stream dump program allows the installation to route various records to separate files and produce a summary activity report.

Note: When you use SMF logging, you can write individual record types to multiple log streams if you prefer, producing duplicate records. If you are dumping several log streams that contain the same record types, the dump can contain duplicate records. In addition, if you dump a log stream that contains data from multiple systems, then coordinate the time zone of the recording systems and the dumping system.

IFASMFDL example

In the example shown in Figure 2-24, the three OUTDD statements refer to the three data sets defined in the JCL, and they specify the SMF record types to be written to each data set. The JCL is explained here.

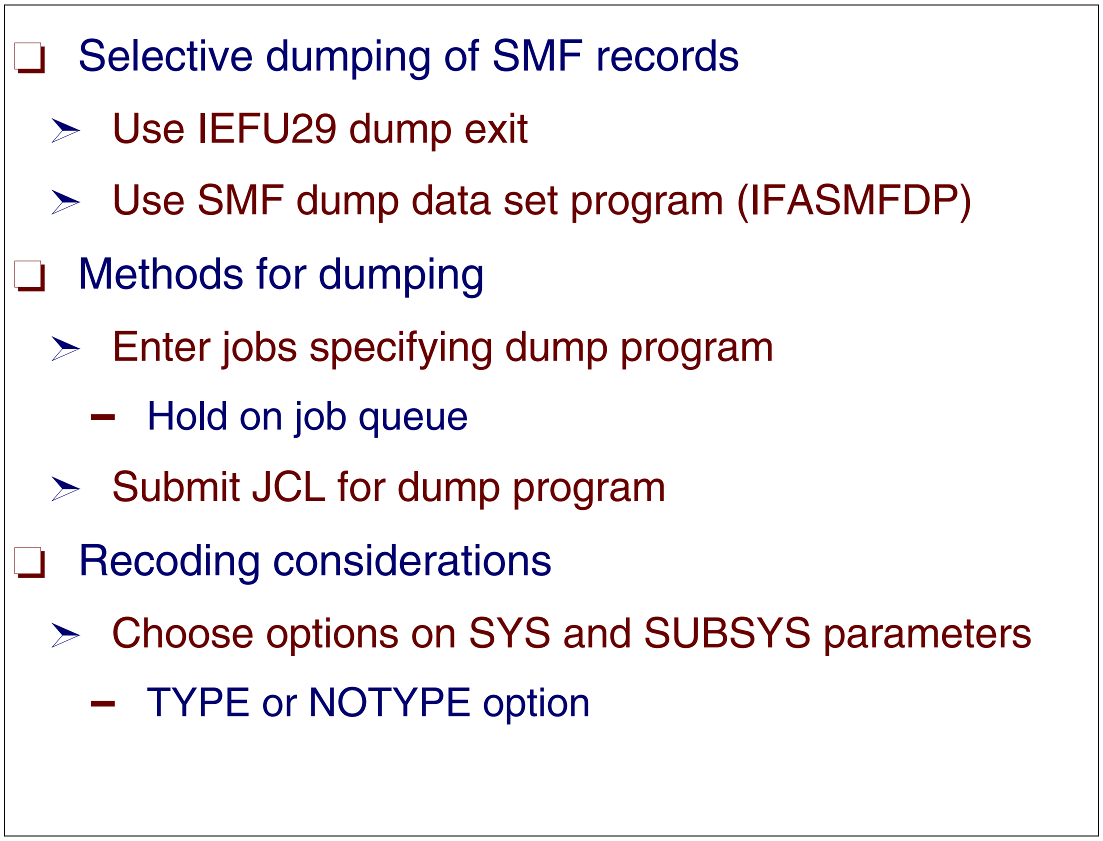
- The DCB= keyword has been coded for the output data set defined by OUTDD2. Any block size 4096 or greater can be specified. Choosing a block size suitable for the device type being used will improve storage resource use. For this job, the data set specified by OUTDD1 will have a system-determined block size. The data set specified by OUTDD2 will have a block size of 32000.
- The LRECL= keyword has been coded for an output data set defined as OUTDD3. For this job, the data set specified by OUTDD3 will have an LRECL of 32760. For OUTDD1 and OUTDD2, the LRECL will default to 32767.
- The LSNAME parameters contain the names of three log streams to be dumped.
- The OUTDD parameters contain filters selecting the SMF record types to be dumped:
  - OUTDD1 specifies that you want to dump record types 0,2,10, 5-30, and subtype 1 of record type 33 starting with those issued at 7:30 am and ending at 6:50 pm.
  - OUTDD2 specifies that you want to dump record types 10 through 255 from dates October 1, 2006 through November 30, 2006.
  - OUTDD3 specifies that you want to dump record types 10 through 255.
- The DATE parameter specifies, for those OUTDD statements which do *not* include the DATE subparameter, that data from January 1, 2006 through December 31, 2006 is to be written.
- The SID parameters specify that data will be dumped for systems 308A and 308B.

```
//IFASMFDL JOB accounting information
//STEP EXEC PGM=IFASMFDL
//OUTDD1 DD DSN=SMFREC.FEWTYPES,DISP=(NEW,CATLG,DELETE)
//OUTDD2 DD DSN=SMF.TYPE10.TYPE255,DISP=(NEW,CATLG,DELETE),
//    DCB=BLKSIZE=32000
//OUTDD3 DD DSN=SMF.TYPE10.TYPE255B,DISP=(NEW,CATLG,DELETE),
//    DCB=LRECL=32760
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
    LSNAME (IFASMF.DEFAULT)
    LSNAME (IFASMF.PERF)
    LSNAME (IFASMF.JOB)
    OUTDD (OUTDD1,TYPE(0,2,10,15:30,33(1)),START(0730),END(1850))
    OUTDD (OUTDD2,TYPE(10:255)),DATE(2006274,2006334)
    OUTDD (OUTDD3,TYPE(10:255))
    DATE (2006001,2006365)
    SID (308A)
    SID (308B)
    END(2400) - DEFAULT
    START(0000) - DEFAULT
```

Figure 2-24 Sample job for dumping SMF log streams

Note: There can be any number of input (LSNAME) or output (OUTDD) parameters in the SMF log stream dump program. The log streams are dumped in reverse order. For example, in Figure 2-24, three log streams are specified. After the SMF log stream dump program is processed, the output files contain the records from log stream IFASMF.JOB first, IFASMF.PERF next, and finally, the records from IFASMF.DEFAULT.

2.22 Dumping selective SMF records



- ❑ Selective dumping of SMF records
  - ➢ Use IEFU29 dump exit
  - ➢ Use SMF dump data set program (IFASMFDP)
- ❑ Methods for dumping
  - ➢ Enter jobs specifying dump program
    - – Hold on job queue
  - ➢ Submit JCL for dump program
- ❑ Recoding considerations
  - ➢ Choose options on SYS and SUBSYS parameters
    - – TYPE or NOTYPE option

Figure 2-25 Selective dump of SMF records

Selective dump of SMF records

When the current recording data set cannot accommodate any more records, the SMF writer routine automatically switches recording from the active SMF data set to an empty SMF data set, and then passes control to the IEFU29 SMF dump exit. The operator is then informed that the data set needs to be dumped. SMF data sets cannot be shared across systems. Avoid having the SMF data set dump program be run from one system in an attempt to clear an SMF data set used by another system.

When notified by the system that a full data set needs to be dumped, operators use the SMF data set dump program (IFASMFDP) to transfer the contents of the full SMF data set to another data set, and to reset the status of the dumped data set to empty so that SMF can use it again for recording data.

Important: Use the IEFU29 SMF dump exit to run the SMF data set dump program. When the current recording data set cannot hold any more records, the SMF writer routine automatically switches recording from the active SMF data set to an empty SMF data set, and passes control to the IEFU29 dump exit. This avoids the need for operator intervention.

Methods for dumping

One method of running the SMF data set dump program is to enter jobs that specify the SMF data set dump program into the system, and hold them on the job queue until a dump is required. Another method is to start a reader to an input stream containing the JCL for the SMF data set dump program. Figure 2-26 shows two sample procedures (DUMPX and DUMPY) for dumping the SMF data sets to a standard-labeled tape (VOL=SER=SMFTAP) with the operator **START** command.

```
//UPDATE    JOB   MSGLEVEL=1
//UPDATE    EXEC  PGM=IEBUPDTE,PARM=NEW
//SYSUT1    DD    DSN=SYS1.PROCLIB,DISP=SHR
//SYSUT2    DD    DSN=SYS1.PROCLIB,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    DATA
./          ADD   NAME=DUMPX,LIST=ALL
//DUMPX     PROC  TAPE=192
//SMFDMP    EXEC  PGM=IFASMFDP
//DUMPIN    DD    DSNAME=SMFDATA,UNIT=&TAPE,DISP=(MOD,KEEP),
//                LABEL=(,SL),VOL=SER=SMFTAP
//SYSPRINT  DD    SYSOUT=A
./          ADD   NAME=DUMPY,LIST=ALL
//DUMPY     PROC  TAPE=192
//SMFDMP    EXEC  PGM=IFASMFDP
//DUMPIN    DD    DSNAME=SYS1.MANY,DISP=SHR
//DUMPOUT   DD    DSNAME=SMFDATA,UNIT=&TAPE,DISP=(MOD,KEEP),
//                LABEL=(,SL),VOL=SER=SMFTAP
//SYSPRINT  DD    SYSOUT=A
./ENDUP
/*
```

Figure 2-26 Sample procedures for dumping the SMF data sets

Figure 2-26 illustrates the following:

- The DCB= keyword has been coded for the output data set defined by OUTDD2. Any block size 4096 or greater can be specified. Choosing a block size suitable for the device type being used will improve storage resource use. For this job, the data set specified by OUTDD1 will have a system-determined block size. The data set specified by OUTDD2 will have a block size of 32000.
- The LRECL= keyword has been coded for an output data set defined as OUTDD3. For this job, the data set specified by OUTDD3 will have an LRECL of 32760. For OUTDD2, the LRECL will default to 32767.

Recording considerations

Subtype selectivity for SMF records is an option of the TYPE or NOTYPE option on the SYS and SUBSYS parameters used for SMF recording. Subtype selectivity allows more flexibility in postprocessing records and helps control the amount of data stored in SMF data sets. The subtype selectivity function supports only records that use the standard SMF record header. The header requires the subtype field to be at offset 22 (decimal) and the “subtypes used” bit (bit position 1) of the system indicator byte at offset 4 (decimal) to be set to X'1'. SMF processes the record for subtype selectivity when both conditions are met. This support is limited to SMF record types 0 through 127 (user records are not supported).

3

Resource Recovery Services (RRS)

Many computer resources are so critical to a company's work that the integrity of these resources must be guaranteed. If changes to the data in the resources are corrupted by a hardware or software failure, human error, or a catastrophe, the computer must be able to restore the data. These critical resources are called protected resources or, sometimes, recoverable resources.

Resource recovery is the protection of the resources. Resource recovery consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources.

This chapter gives an introduction to the function of Resource Recovery Service (RRS) in a z/OS system. RRS provides a global syncpoint manager that any resource manager on z/OS can exploit. RRS is increasingly becoming a prerequisite for new resource managers, and for new capabilities in existing resource managers. Rather than having to implement their own two-phase commit protocol, these products can use the support provided by RRS. However, as more transaction managers have become RRS resource managers, and as the complexity of the exchanges of transactional data increases, more and more systems and application programmers will need to use RRS. This chapter covers the following topics:

- Introduction to Resource Recovery Services (RRS)
- Two-phase commit protocol
- Two-phase commit process
- Two-phase backout process
- Resource Manager
- Setting up RRS in the z/OS system
- Exploiters of RRS
- RRS commands

Note: More information about RRS is available in:

- *Systems Programmer's Guide to Resource Recovery Services (RRS),* SG24-6980
- *z/OS MVS Programming: Resource Recovery,* SA22-7616

3.1 Introduction to Resource Recovery Services (RRS)

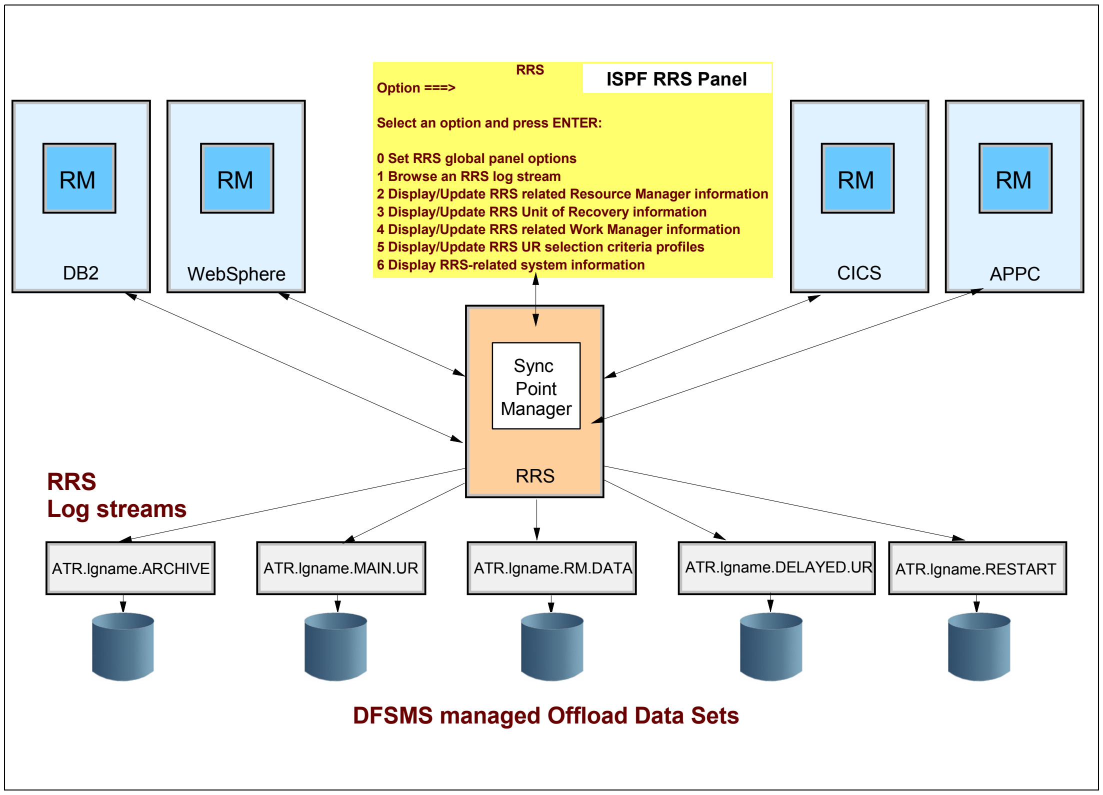

RM
DB2
RM
WebSphere
RRS
Option ===>
ISPF RRS Panel
Select an option and press ENTER:
0 Set RRS global panel options
1 Browse an RRS log stream
2 Display/Update RRS related Resource Manager information
3 Display/Update RRS Unit of Recovery information
4 Display/Update RRS related Work Manager information
5 Display/Update RRS UR selection criteria profiles
6 Display RRS-related system information
RM
CICS
RM
APPC
Sync Point Manager
RRS
RRS Log streams
ATR.lgname.ARCHIVE
ATR.lgname.MAIN.UR
ATR.lgname.RM.DATA
ATR.lgname.DELAYED.UR
ATR.lgname.RESTART
DFSMS managed Offload Data Sets

Figure 3-1 Introduction to Resource Recovery Services (RRS)

Introduction to Resource Recovery Services (RRS)

To add the RRS panels as an option on your ISPF primary panel, you should make a copy of the ISPF primary option menu – ISR@PRIM. Then, add the following information to the processing section of the panel:

```
RRS,'PANEL(ATRFPCMN) NEWAPPL(RRSP)'
```

Make sure you concatenate the library containing your customized primary panel before any others in your logon procedure or CLIST.

Note: There is an alternative way to access the RRS panels from the ISPF Main Menu. You can do this by using the following steps:

1. Go to your ISPF Main Menu and choose option 7.1 to perform dialog testing:
   ```
   Option ===> 7.1
   ```
2. After the ISPF Dialogue Test Menu is displayed, invoke the selection panel by doing the following:
   a. Put `atrfpcmn` in the panel option.
   b. Use `atrk` as the ID.
   c. Press Enter to display the RRS main panel.

RRS exploiters

There are many exploiters of *Resource Recovery Services (RRS),* and each has its own *resource manager (RM)*. With the increasing number of resource managers now available on z/OS, there was clearly a need for a general syncpoint manager on z/OS that any resource manager could exploit. This is the role of RRS, a component of z/OS. It enables transactions to update protected resources managed by many resource managers. Applications must be sure that changes to data have been made or backed out and so the applications must access this data via the RM.

Figure 3-1 shows how RRS acts as a syncpoint manager in the z/OS system.

RRS functions

RRS coordinates the two-phase commit process, which is a protocol used by the RM exploiter. RRS runs in its own address space and normally is started at IPL time, but if RRS suffers problems it is restartable. RRS is also responsible for the association between a *Unit of Recovery (UR)* and a Work Context. RRS builds this UR each time an RM wants to make a transaction. A UR is the base element of a transaction. A Work Context is a representation of a work request (transaction) and may consist of a number of Units of Recovery. RRS is an exploiter of the System Logger and stores the logs with logger functions in the coupling facility.

The advantage of using RRS is to ensure data integrity for all applications that exploit RRS because if a transaction fails, RRS notifies the resource managers.

3.2 Two-phase commit protocol

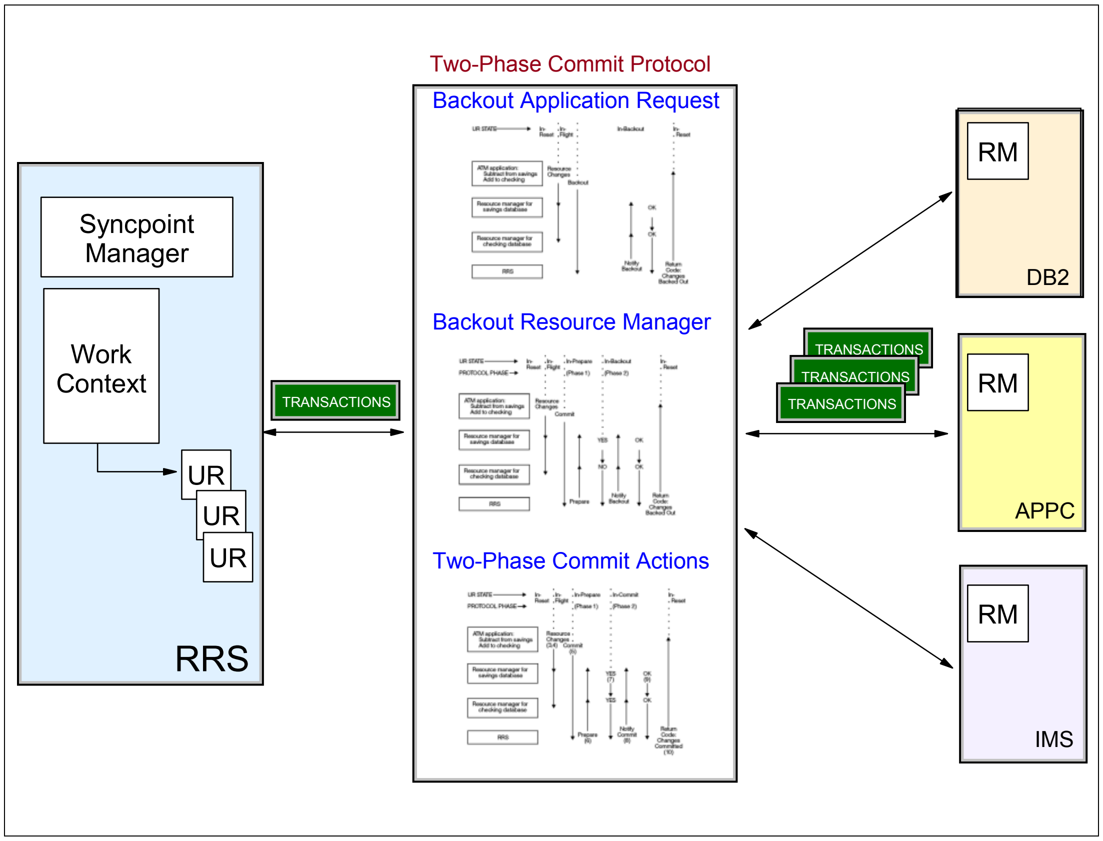


Figure 3-2 Two-phase commit protocol

Two-phase commit protocol

RRS is a syncpoint manager that provides services to each RM and informs the RM about any changes. An exit manager is an authorized program that gives control to an RM in case there is an event that should be triggered. RRS only checks that each event is processed in the right sequence, and in case there are problems RRS informs the RM. This means that RRS itself does not perform any actions on a DB2 database or back out changes to an IMS database. Instead, RRS triggers resource manager-supplied exits based on certain events. An application can request RRS to commit a unit of recovery (UR), and RRS then invokes the commit exits for all the resource managers involved in the UR.

Over the years resource managers on z/OS have evolved to provide two-phase commit processing between them. When the application is ready to commit or back out its changes, the application invokes RRS to begin the two-phase commit protocol.

RRS supports the two-phase commit protocol as shown in Figure 3-2. The two-phase commit protocol is a set of actions used to make sure that an application program either makes *all* changes to the resources represented by a single unit of recovery (UR), or makes *no* changes at all. This protocol verifies that either all changes or no changes are applied even if one of the elements (such as the application, the system, or the resource manager) fails. The protocol allows for restart and recovery processing to take place after system or subsystem failure.

Phase 1

In the first phase, each resource manager prepares to commit the changes. A resource manager typically prepares by writing the unchanged data image, often called undo data, and the changed data image, often called redo data, in a resource manager log that it can access during restart.

If the resource manager can then commit the changes, it tells RRS that it agrees to allow the commit to continue. If the resource manager cannot commit the changes, it tells RRS to back out the changes.

The decision to commit or back out the changes represented by a UR depends on responses from all of the resource managers. If the decision is to commit the changes, RRS hardens the decision, meaning that it stores the decision in an RRS log, and phase 2 begins. If the decision is to back out the changes, RRS generally does not harden the decision, and phase 2 begins as soon as the decision is made.

Once a commit decision is hardened, the application changes are considered to be committed. If the application, the system, RRS, or a resource manager fails after the decision is hardened, the application changes will be made during restart. Before the decision is hardened, a failure of the application, the system, RRS, or a resource manager would cause the changes to be backed out during restart.

Phase 2

In the second phase, the resource managers commit or back out the changes represented by a UR.

Two-phase protocol

The two-phase commit protocol has two functions; Figure 3-3 on page 164 and Figure 3-4 on page 165 describe this protocol:

- **Commit -** During the commit process, all changes to both local and distributed resources are made permanently.
- **Backout -** During the backout process, all pending changes to both local and distributed resources are not made.

The set of changes that are to be made or not made as a unit are called a unit of recovery (UR). A UR represents an application program's changes to resources since the last commit or backout or, for the first UR, since the beginning of the application. Each UR is associated with a context, which consists of the UR, or more than one UR, with the associated application programs, resource managers, and protected resources.

A context, which is sometimes called a work context, represents a work request. The life of a context consists typically of a series of URs. The two-phase commit protocol is initiated when the application is ready to commit or back out its changes. The application program that initiates the commit or backout does not need to know who the syncpoint manager is or how two-phase commit works. This is hidden from the application; a program simply calls a commit or backout service and receives a return code indicating whether it has been successfully completed.

3.3 Two-phase commit process

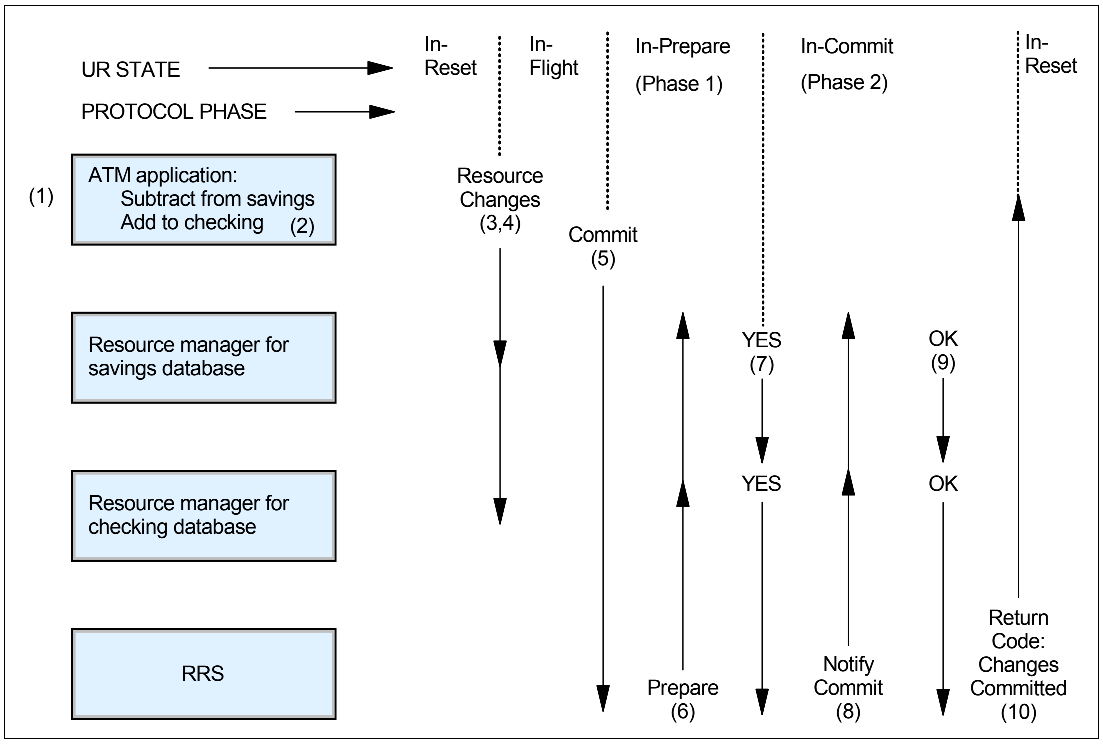


Figure 3-3 Two-phase commit process

Two-phase commit process

The two-phase commit protocol is a set of actions used to make sure that an application program makes all changes to the collection of resources represented by a UR or makes no changes to the collection. The protocol verifies the all-or-nothing changes even if the application program, the system, RRS, or a resource manager (RM) fails.

In the first phase, each resource manager prepares to commit the changes. A resource manager typically prepares by writing the unchanged data image, often called undo data, and the changed data image, often called redo data, in a RM log access during restart.

If the RM can then commit the changes, it tells RRS that it agrees to allow the commit to continue. If the RM cannot commit the changes, it tells RRS to back out the changes.

The decision to commit or back out the changes represented by a UR depends on responses from all of the resource managers. If the decision is to commit the changes, RRS hardens the decision, meaning that it stores the decision in an RRS log, and phase 2 begins. If the decision is to back out the changes, RRS generally does not harden the decision, and phase 2 begins as soon as the decision is made.

Once a commit decision is hardened, the application changes are considered to be committed. If the application, the system, RRS, or a resource manager fails after the decision is hardened, the application changes will be made during restart. Before the decision is hardened, a failure of the application, the system, RRS, or a resource manager would cause the changes to be backed out during restart.

In the second phase, the resource managers commit or back out the changes represented by a UR.

3.4 Two-phase backout process

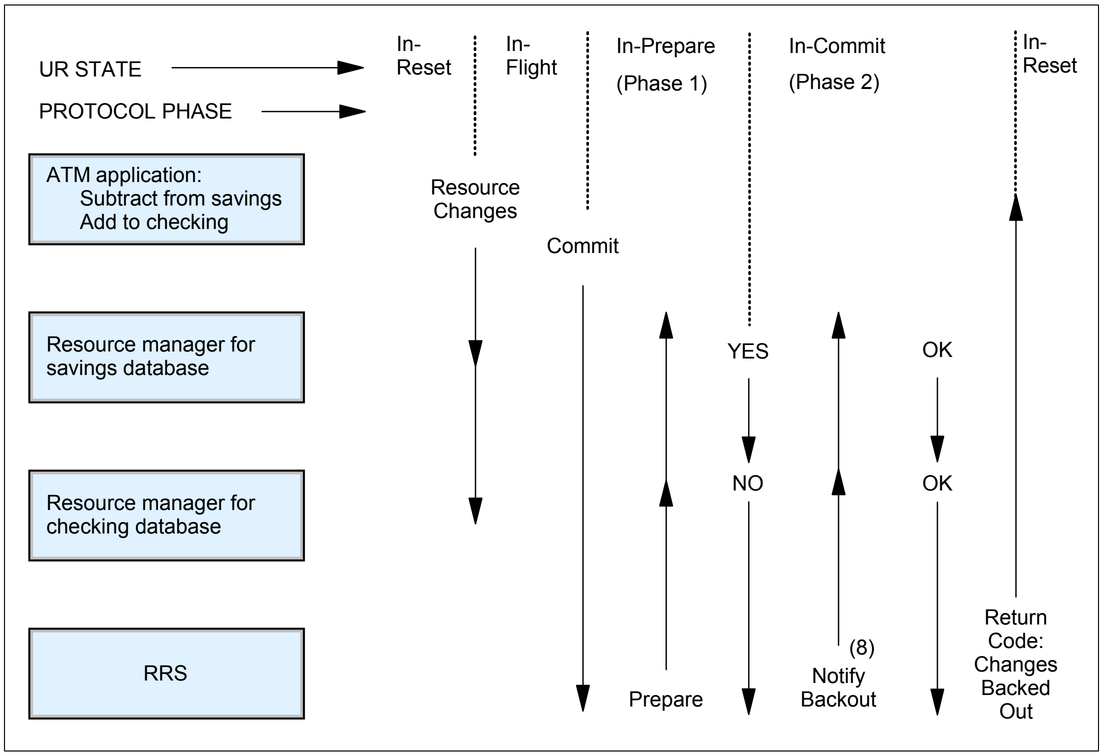


Figure 3-4 Two-phase back out process

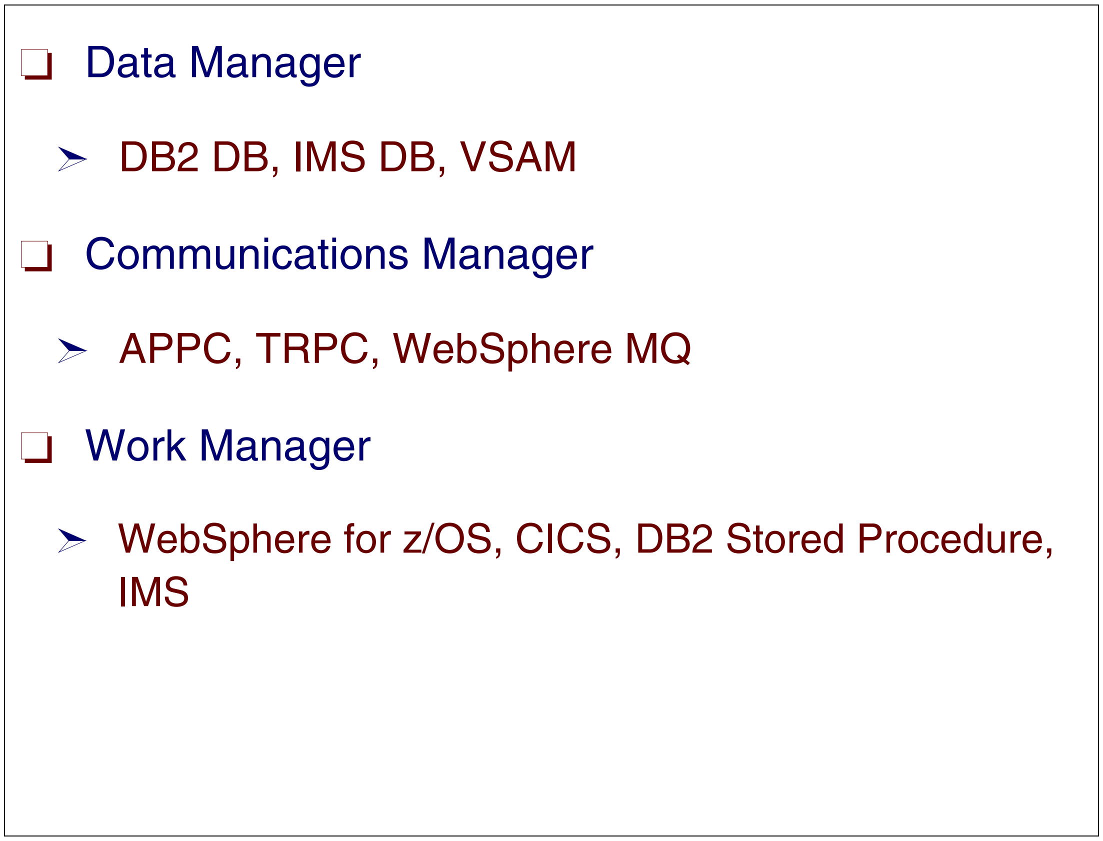
Two-phase back out process

If, for any reason, the application cannot complete the transaction, the application requests backout in step 5, instead of commit. In this case, the changes are backed out and are not actually made in any database.

A resource manager can also request backout. If a resource manager cannot make the change to its database, the resource manager votes NO during prepare. If any resource manager votes NO, all of the changes are backed out.

This example describes the module flow after a backout request from a resource manager. Figure 3-4 shows that the application cannot complete the transfer. After the user request to access information, the resource manager requests backout in step 8, instead of commit. Now all changes are backed out and are not actually made in any database.

3.5 Resource Manager



- Data Manager
  - DB2 DB, IMS DB, VSAM
- Communications Manager
  - APPC, TRPC, WebSphere MQ
- Work Manager
  - WebSphere for z/OS, CICS, DB2 Stored Procedure, IMS

Figure 3-5 Resource Manager

Resource Manager

An RM is a subsystem or component such as CICS, IMS, VSAM, WebSphere for z/OS or DB2, that manages resources that can be involved in transactions. Each exploiter of RRS communicates with the syncpoint manager of RRS to keep information updated. There are three types of RM:

Data managers

Data managers allow an application to read and change data. To process a syncpoint event, a data resource manager would take actions such as committing or backing out changes to the data it manages for:

- DB2, IMS DB, VSAM

Communication managers

Communication managers control access to distributed resources and act as an extension to the syncpoint manager. A communications resource manager provides access to distributed resources by allowing an application to communicate with other applications and resource managers, possibly located on different systems. It acts as an extension to the syncpoint manager by allowing the local syncpoint manager to communicate with other syncpoint managers as needed to ensure coordination of the distributed resources the application accesses for:

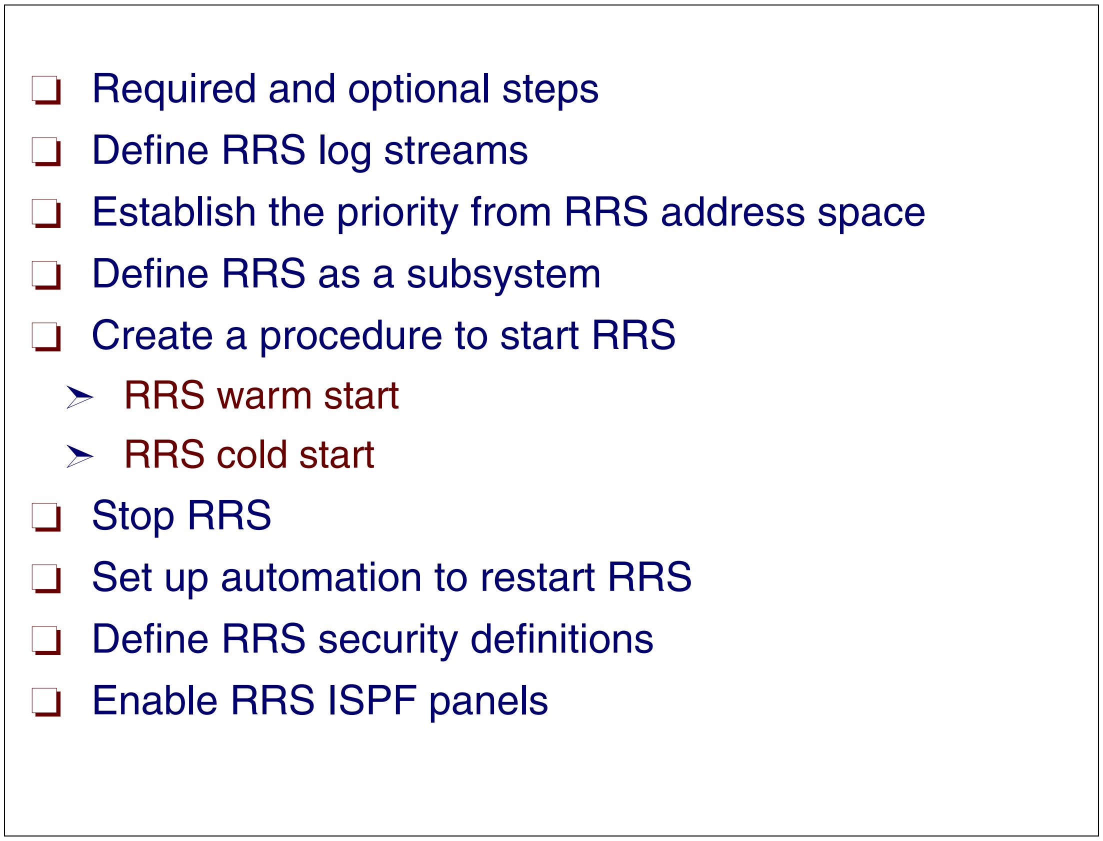
- APPC, TRPC, and WebSphere MQ

Work managers

Work managers are resource managers that control applications' access to system resources. To process a syncpoint event, a work manager might ensure that the application is in the correct environment to allow the syncpoint processing to continue for:

- IMS, CICS, DB2 Stored Procedure, and WebSphere for z/OS

3.6 Set up RRS in the z/OS system



- Required and optional steps
- Define RRS log streams
- Establish the priority from RRS address space
- Define RRS as a subsystem
- Create a procedure to start RRS
  - RRS warm start
  - RRS cold start
- Stop RRS
- Set up automation to restart RRS
- Define RRS security definitions
- Enable RRS ISPF panels

Figure 3-6 Set Up RRS in the z/OS system

Required and optional steps

There are some steps required to set up RRS on a z/OS system. Some are optional but should be done for recovery actions. Table 3-1 shows which steps should be performed.

Table 3-1 RRS setup tasks

Task	Required/Optional
Define the RRS log streams	Required
Establish the priority for the RRS address space	Required
Define RRS as a subsystem	Required
Create procedures to start RRS	Required
Set up automation to restart RRS	Optional
Define RRS security definitions	Optional
Enable RRS ISPF panels	Optional (enables ISPF applications to look at RRS log streams)
Set up RRS component traces	Optional

Define the RRS log streams

There are five RRS log streams that can be shared by multiple systems in a sysplex, as displayed in Table 3-2. Except for the Archive log stream, the other four log streams are necessary to start RRS. It is possible to share the log streams in a sysplex. Then the log streams must reside in a structure that can be written to a Coupling Facility (CF) or to DASD.

There can only be one RRS active in one system. To reduce the risk of problems, RRS on different systems can run in one logging group. A RRS logging group is a group of systems that share RRS log streams. If RRS fails now at one system in the sysplex, another RRS in the same logging group can take over the outstanding work, which is written to the RESTART log stream.

Table 3-2 Content from the RRS log streams

Log stream type	Log stream name	Content
RRS archive log	ATR.lgname.ARCHIVE	Information about completed URs.This log is optional. See Note for further details.
RRS main UR state log	ATR.lgname.MAIN.UR	The state of active URs. RRS periodically moves this information into the RRS delayed UR state log when UR completion is delayed.
RRS resource manager data log	ATR.lgname.RM.DATA	Information about the resource managers using RRS services.
RRS delayed UR state log	ATR.lgname.DELAYED.UR	The state of active URs, when UR completion is delayed.
RRS restart log	ATR.lgname.RESTART	Information about incomplete URs needed during restart. This information enables a functioning RRS instance to take over incomplete work left over from an RRS instance that failed.
Note: If the write activity to the RRS ARCHIVE log stream is very high, this might impact the performance throughput of ALL RRS transactions if this log stream is defined and actively in use by RRS. This log stream is fully optional and only needed by the installation for any type of post transaction history type of investigation. To avoid this performance impact, the installation can DELETE the archive log stream. To do so, disconnect the log stream from RRS on *all* systems and then use the IXCMIAPU utility, DELETE LOGSTREAM NAME(ATR.<groupname>.ARCHIVE) to delete the log stream definition and, using the IDCAMS utility, DELETE IXGLOGR.ATR.<groupname>.ARCHIVE.* SCRATCH to delete the data set associated with the log stream. If you choose not to use the ARCHIVE log stream, a warning message is issued at RRS startup time about not being able to connect to the log stream (however, RRS will continue its initialization process).		

Use the Coupling Facility Structure Sizer Tool (CFSIZER) to plan your structure sizes. The sizing recommendations are always done for the highest available CFCC level. The CFSIZER is provided at the following address:

`http://www-1.ibm.com/servers/eserver/zseries/cfsizer/`

Establish the priority for the RRS address space

RRS must run in the same service class or higher as an application that uses the RRS service. IBM recommends to put RRS in the SYSSTC service class.

Define RRS as a subsystem

To define RRS as a subsystem, place the following statement in the IEFSSNxx parmlib member:

```
SUBSYS SUBNAME(RRS)
```

Place this statement after the statement that defines the primary subsystem. You can replace RRS with a subsystem name of your choice, but do not supply any other parameters. In particular, do not supply an initialization routine.

Or issue the console command after IPL to dynamically define the RRS subsystem to the SSI. You can replace RRS with a subsystem name of your choice, but do not supply any other parameters.

```
SETSSI A DD,SUBNAME=RRS
```

Create a procedure to start RRS

RRS is a system address space and normally started at IPL Time. It stays active on the system as long as the resource manager requires RRS function. If Logger is active and JES is up, then RRS can be started with SUB=MSTR with the operator command:

```
START RRS
```

IBM supplies a cataloged procedure named ATRRRS in SYS1.SAMPLIB that you can use to start RRS after system initialization. Your installation should copy SYS1.SAMPLIB(ATRRRS) to SYS1.PROCLIB(RRS). The member name RRS is specified here and can be replaced with any other member name, as long as it matches the subsystem name specified in the SYS1.PARMLIB(IEFSSNxx) used by the installation. If the names do not match, you may receive error messages when you start the subsystem.

Figure 3-7 shows a message flow from the starting sequence.

```
S RRS
...
ATR221I RRS IS JOINING RRS GROUP SANDBOX ON SYSTEM SC70
...
IXL014I IXLCONN REQUEST FOR STRUCTURE RRS_DELAYEDUR_1 246
WAS SUCCESSFUL.  JOBNAME: IXGLOGR ASID: 0016
CONNECTOR NAME: IXGLOGR_SC70 CFNAME: CF1
....
ASA2011I RRS INITIALIZATION COMPLETE. COMPONENT ID=SCRRS
```

Figure 3-7 Message sequence after S RRS

RRS warm start

With a warm start, RRS can complete work that was in progress when a previous RRS instance failed or was intentionally stopped. A warm start occurs when all of the RRS log streams are intact and available to the restarting RRS instance. A warm start also occurs when RRS is joining an existing logging group in a sysplex. In effect, any attempt to start RRS when its logs are not empty is a warm start.

RRS performs a warm start once it has access to the RM.DATA log stream and there is data from URs in this log stream. If RRS successfully warm starts, then all log streams (except the ARCHIVE log stream) should be intact because RRS checks the log stream during startup.

RRS cold start

An RRS cold start is performed when the RM.DATA log steam is empty. This means that there are no more RMs connected to the RM.DATA log stream and the log stream must be deleted and defined via IXCMIAPU. An RRS cold start applies to all members in the sysplex that are in the same logging group. This is because all the members of the logging group are sharing the same log stream. In case there are problems with an RRS and a cold start is necessary, all RMs that are connected to RRS must be shut down and the following steps should be performed:

3. Issue the following command to discover the LSN and structure name for the RM.DATA log stream:

   `D LOGGER,L`

4. Route a **`SETRRS CANCEL`** command to all systems belonging to this logging group in the sysplex where RRS is active. Shut down all the RMs using the RRS log stream.
5. The next command will show if there are any outstanding requests and *must* be issued from each system in the logging group. If there are any pending requests, an RRS cold start is not possible.

   `D LOGGER,L,LSN=<log stream name from RM.DATA>`

6. The next step is to determine the attributes from the allocated log streams.
7. Run the IXCMIAPU job to delete and redefine a new RM.DATA log stream.

```
//* ------------------------------------------------------------------
//* DELETE RRS LOGSTREAM FOR COLDSTART
//* ------------------------------------------------------------------
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 DATA TYPE (LOGR) REPORT(NO)
   DELETE LOGSTREAM
          NAME(<log stream name from RM.DATA>)
   DEFINE LOGSTREAM
          STRUCTNAME(RRS_MAINUR_1)
          NAME(ATR.SANDBOX.MAIN.UR)
          LS_DATACLAS(SHAR)
          HLQ(LOGR) MODEL(NO)
          LS_SIZE(1024)
          LOWOFFLOAD(0)
          HIGHOFFLOAD(80)
          STG_DUPLEX(NO)
          RETPD(15)
          AUTODELETE(YES)
          OFFLOADRECALL(YES)
          DASDONLY(NO)
          DIAG(NO)
          LOGGERDUPLEX(UNCOND)
          EHLQ(NO_EHLQ)
```

Figure 3-8 Sample JCL for deleting and redefining a new RM.DATA log stream

8. Start RRS now on the affected systems in the sysplex. Because RM.DATA is empty, RRS will perform a cold start. RRS will move any data from MAIN.UR and DELAYED.UR to the ARCHIVE log. Check the SYSLOG for any ATR* messages. Use the RRS ISPF panel to display these incomplete URs.
9. Start the resource manager, which uses the RRS service, and look for messages in the SYSLOG and joblog.

Stopping RRS

RRS can be stopped in case of problems or before you IPL a system. Bring down all applications and RMs that utilize RRS services prior to cancelling RRS. This will minimize the amount of manual intervention required when you restart the applications and RMs. Cancelling RRS before an IPL results in a cleaner system recovery. Figure 3-9 shows the message flow in the SYSLOG after stopping RRS. RRS can be stopped with the following command.

```
SETRRS CANCEL
```

If this command does not stop RRS, you can use the following command:

```
FORCE RRS,ARM
```

```
SETRRS CANCEL
...
ATR101I CANCEL REQUEST WAS RECEIVED FOR RRS.
ATR143I RRS HAS BEEN DEREGISTERED FROM ARM.
...
ASA2960I RRS SUBSYSTEM FUNCTIONS DISABLED. COMPONENT ID=SCRRS
ATR167I RRS RESMGR PROCESSING COMPLETED.
```

Figure 3-9 Command sequence from a SETRRS CANCEL

Set up automation to restart RRS

If RRS fails, it can use automatic restart management (ARM) to restart itself in a different address space on the same system. RRS, however, will not restart itself following a SETRRS CANCEL command. To stop RRS and cause it to restart automatically, use the FORCE command with ARM and ARMRESTART.

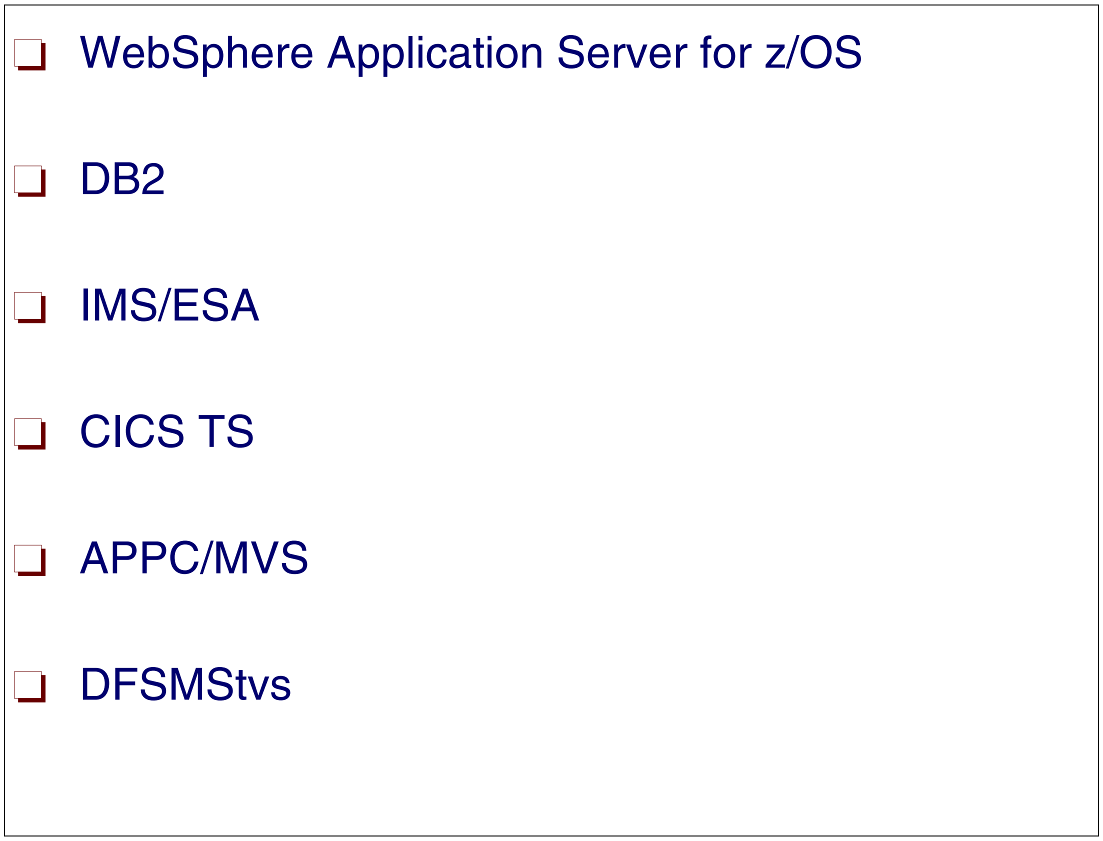
Define RRS security definitions

In your installation, you can configure RRS to allow a user to manage all the RRS images in the sysplex from a single image. Access to RRS system management functions is controlled by the following RACF resource. To control RRS access across a sysplex, RRS uses the MVSADMIN.RRS.COMMANDS.*gname.sysname* resource in the FACILITY class, where *gname* is the logging group name and *sysname* is the system name. If you are running RRS on a single system, RRS can use either the MVSADMIN.RRS.COMMANDS.*gname.sysname* resource or the MVSADMIN.RRS.COMMANDS resource in the FACILITY class to control access to RRS system management functions on the current system.

Enable RRS ISPF panels

RRS provides an interface to communicate via ISPF. With ISPF it is possible to display information from URs and RMs. We recommend that you set up the RRS panels, because you may encounter failure scenarios in which you would need to use the panel information to clean up outstanding transactions. There is no other mechanism for determining the state of the various resource managers. If you have a problem running RRS, you will need to use the RRS panels to help identify and fix the trouble in your sysplex. To enable this panel you have to add some libraries to your ISPF concatenation.

3.7 RRS exploiters



- WebSphere Application Server for z/OS
- DB2
- IMS/ESA
- CICS TS
- APPC/MVS
- DFSMStvs

Figure 3-10 Exploiters of RRS

WebSphere Application Server for z/OS

WebSphere for z/OS uses RRS services as a part of its base implementation to provide two-phase commit support. WebSphere for z/OS does not start without RRS being available and it abends in the event of an RRS failure.

DB2

The DB2 RM responsible for global commit coordination for a given transaction depends on the scope of the transaction, where the transaction originated, and the interfaces and products used. RRS will only be involved in a DB2 transaction if an external syncpoint manager is required—and that requirement generally only exists if there is more than one resource manager involved in a transaction and a part of the transaction executes outside the control of that DB2 resource manager.

IMS/ESA

RRS manages the syncpoint process for all applications that are participating with IMS in protected conversation.

CICS

CICS provides a two-phase commit protocol for all communication over the external CICS interface (EXCI). EXCI needs RRS as syncpoint manager to manage the requests between CICS and the client application.

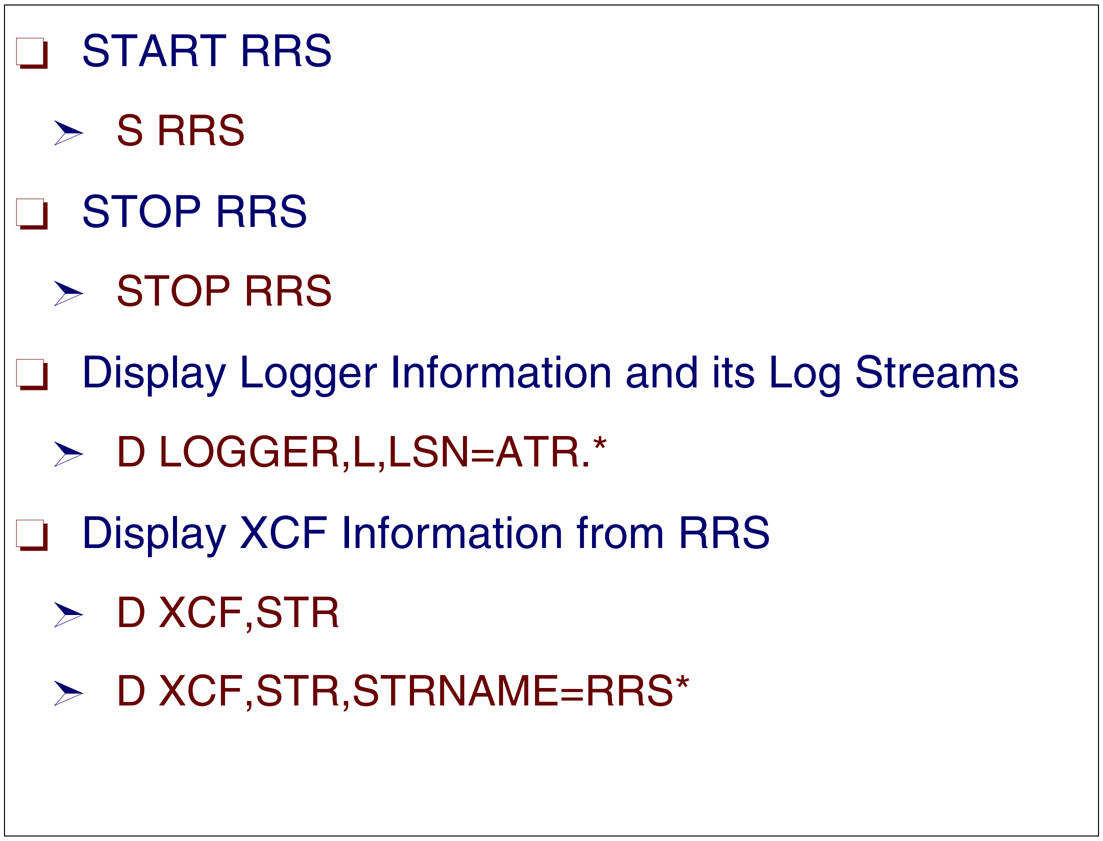
APPC/MVS

APPC supports protected conversations to communicate with the partner Logical Unit (LU). If one of the LUs is ready to commit or back out a change in the process then RRS comes into the game and handles this request as syncpoint manager in the sysplex.

DFSMStvs

DFSMStvs is an enhancement of VSAM RLS and allows batch applications to update recoverable data sets. DFSMStvs uses the same logging mechanism as VSAM RLS and this support is managed from RRS. To update a data set, DFSMStvs uses functions that are managed from RRS.

3.8 RRS commands



- START RRS
  - S RRS
- STOP RRS
  - STOP RRS
- Display Logger Information and its Log Streams
  - D LOGGER,L,LSN=ATR.*
- Display XCF Information from RRS
  - D XCF,STR
  - D XCF,STR,STRNAME=RRS*

Figure 3-11 RRS commands

RRS commands

This section shows a summary of commands that are useful to get information about RRS. There are no commands to get information about the RRS logstream content. This information is available through the ISPF panel.

Starting RRS

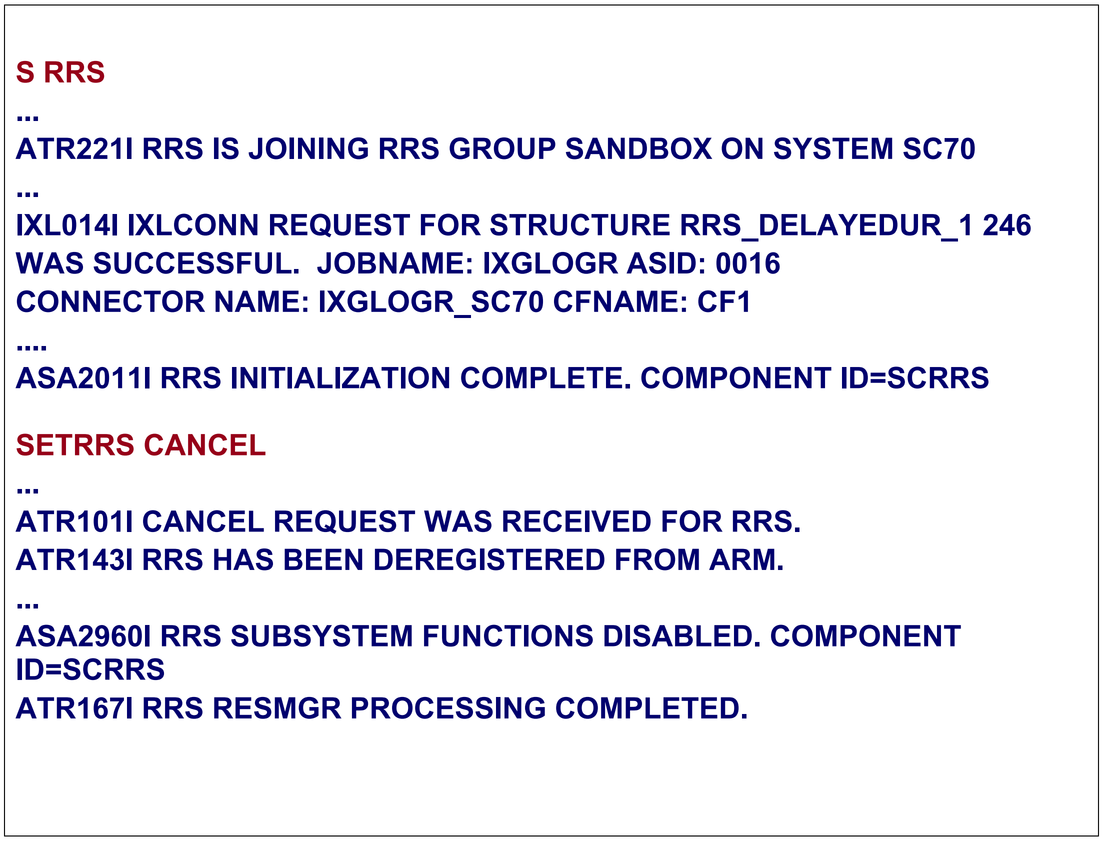
Use the **`START RRS`** command to start resource recovery services (RRS). To start RRS during system initialization, add a **`START RRS`** command to the COMMNDxx parmlib member.

Before you can start RRS, your installation must have defined RRS as a subsystem in the IEFSSNxx parmlib member. For RRS to process requests for resources, System Logger must be active.

RRS procedure

The name of the cataloged procedure that IBM supplies in SYS1.SAMPLIB for starting the RRS subsystem is ATRRRS. Your installation should copy SYS1.SAMPLIB(ATRRRS) to SYS1.PROCLIB(RRS). If your installation replaces membername RRS with its own procedure for starting RRS, it should ensure that the name of its procedure matches the name of the subsystem specified in the IEFSSNxx parmlib member it uses. Otherwise, you may receive error messages when you start the subsystem.

3.9 Command output from RRS



```
S RRS
...
ATR221I RRS IS JOINING RRS GROUP SANDBOX ON SYSTEM SC70
...
IXL014I IXLCONN REQUEST FOR STRUCTURE RRS_DELAYEDUR_1 246
WAS SUCCESSFUL.  JOBNAME: IXGLOGR ASID: 0016
CONNECTOR NAME: IXGLOGR_SC70 CFNAME: CF1
....
ASA2011I RRS INITIALIZATION COMPLETE. COMPONENT ID=SCRRS

SETRRS CANCEL
...
ATR101I CANCEL REQUEST WAS RECEIVED FOR RRS.
ATR143I RRS HAS BEEN DEREGISTERED FROM ARM.
...
ASA2960I RRS SUBSYSTEM FUNCTIONS DISABLED. COMPONENT
ID=SCRRS
ATR167I RRS RESMGR PROCESSING COMPLETED.
```

Figure 3-12 START and STOP RRS command output

START commands

Figure 3-12 shows the message output after the operator issued the **S RRS** and **STOP RRS** commands from the console. Only one copy of RRS can be running on a system. The system will reject an attempt to start a second RRS, even if you specify a different procedure as the first parameter of the **START** command. The syntax of the command follows:

```
S RRS | membername [,CTMEM=CTnRRSxx] [,GNAME=lgrpname ][,JOBNAME=jobname]
```

Where:

membername	Invokes the RRS procedure and creates the RRS address space. If your installation has created a different procedure for starting RRS, use the member name of your procedure.
CTMEM=CTnRRSxx	Identifies the CTnRRSxx parmlib member that contains the options the RRS component trace is to use when RRS starts the trace. If you omit this optional parameter, RRS traces only unexpected events until a time when the TRACE CT command specifies different trace options.
GNAME=lgrpname	Specifies the log group name. A log group is a group of systems that share an RRS workload. Specify a value if your installation has multiple RRS workloads. Otherwise, the name defaults to the sysplex name. If you specify a name, it must be 1-8 characters long. The first character must be alphabetic or one of the national characters ($, #, or @), while the remaining characters may be alphanumeric or $, #, or @.

3.10 Display Logger information

```
D LOGGER,L
IXG601I   19.09.06  LOGGER DISPLAY 913
INVENTORY INFORMATION BY LOGSTREAM
LOGSTREAM                  STRUCTURE        #CONN  STATUS
---------                  -----------      -----  ------
ATR.SANDBOX.ARCHIVE        RRS_ARCHIVE_1    000004 IN USE
  SYSNAME: SC64
    DUPLEXING: LOCAL BUFFERS
  SYSNAME: SC63
    DUPLEXING: LOCAL BUFFERS
  SYSNAME: SC65
    DUPLEXING: LOCAL BUFFERS
  SYSNAME: SC70
    DUPLEXING: LOCAL BUFFERS
ATR.SANDBOX.DELAYED.UR     RRS_DELAYEDUR_1  000004 IN USE
  SYSNAME: SC64
    DUPLEXING: LOCAL BUFFERS
  SYSNAME: SC63
    DUPLEXING: LOCAL BUFFERS
  SYSNAME: SC65
    DUPLEXING: LOCAL BUFFERS
  SYSNAME: SC70
    DUPLEXING: LOCAL BUFFERS
```

Figure 3-13 Displaying Logger information from RRS

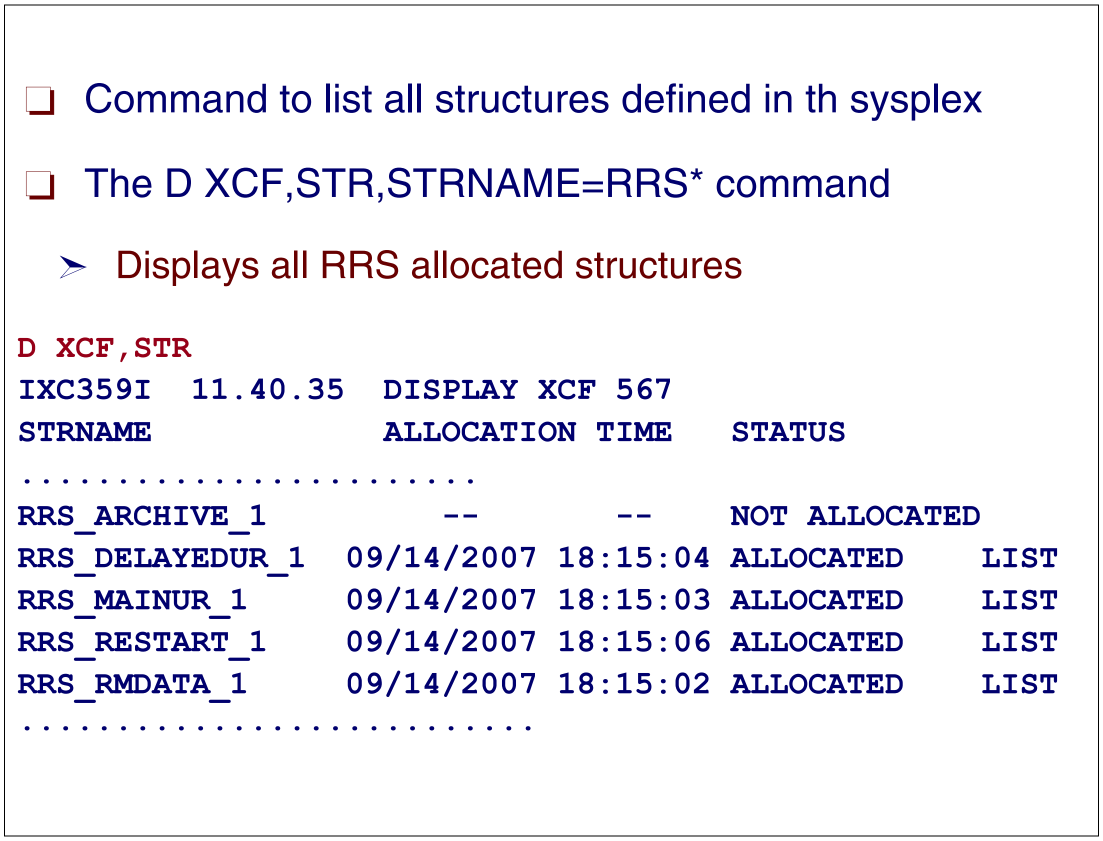
Displaying System Logger information

To get information about the active RRS log streams and the status of these log streams, the command `D LOGGER,L,LSN=ATR.*` should be issued at the console.

When displaying log stream sysplex information, using `D LOGGER,L`, the options are as follows:

LSName or LSN = logstreamname	This filter requests a display of all defined log streams that match the specified log stream name.
STRName or STRN = structurename	This filter requests a display of all log streams on the sysplex that are defined to a structure that matches the specified structure name.
DASDONLY	This filter requests a display of all log streams that match other filters that have a DASDONLY configuration.

3.11 XCF information about RRS structures



- Command to list all structures defined in th sysplex
- The D XCF,STR,STRNAME=RRS* command
  - Displays all RRS allocated structures

```
D XCF,STR
IXC359I  11.40.35  DISPLAY XCF 567
STRNAME            ALLOCATION TIME   STATUS
........................
RRS_ARCHIVE_1         --       --    NOT ALLOCATED
RRS_DELAYEDUR_1  09/14/2007 18:15:04 ALLOCATED    LIST
RRS_MAINUR_1     09/14/2007 18:15:03 ALLOCATED    LIST
RRS_RESTART_1    09/14/2007 18:15:06 ALLOCATED    LIST
RRS_RMDATA_1     09/14/2007 18:15:02 ALLOCATED    LIST
...........................
```

Figure 3-14 Display XCF information about allocated XCF structures

Displaying information about RRS structures

The **D XCF,STR** command shows you a summary of all the structures that are currently defined in the sysplex.

The command requests information about the coupling facility structures in the policy. If specified without further qualification, summary information (message IXC359I) will be displayed about all coupling facility structures that are in the policy. Using the STRNAME keyword requests the system to display detail information.

Figure 3-15 shows detailed information about all allocated RRS structures. The **D XCF,STR,STRNAME=RRS*** command gives you information about the status, the time when the structure was allocated, and the connectors to this structure.

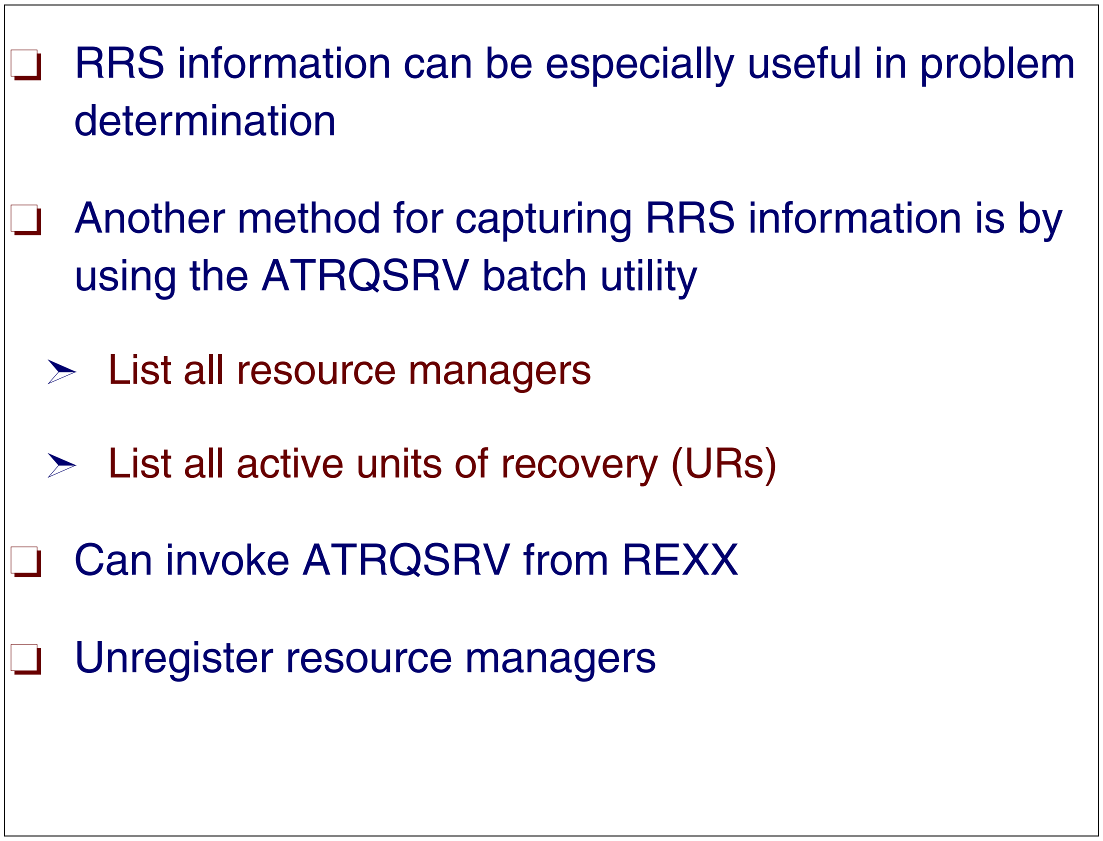
```
D XCF,STR,STRNAME=RRS*
IXC360I  08.52.40  DISPLAY XCF 965
STRNAME: RRS_MAINUR_1
 STATUS: ALLOCATED
 TYPE: LIST
 POLICY INFORMATION:
  POLICY SIZE    : 16000 K
  POLICY INITSIZE: 8000 K
  POLICY MINSIZE : 0 K
  FULLTHRESHOLD  : 80
  ALLOWAUTOALT   : NO
  REBUILD PERCENT: 5
  DUPLEX         : DISABLED
  PREFERENCE LIST: CF1      CF2
  ENFORCEORDER   : NO
  EXCLUSION LIST IS EMPTY

 ACTIVE STRUCTURE
 ----------------
  ALLOCATION TIME: 02/24/2005 14:22:39
  CFNAME         : CF1
  COUPLING FACILITY: 002084.IBM.02.000000026A3A
                     PARTITION: 1F   CPCID: 00
  ACTUAL SIZE    : 8448 K
  STORAGE INCREMENT SIZE: 256 K
  ENTRIES:  IN-USE:          6 TOTAL:       3151,   0% FULL
  ELEMENTS: IN-USE:         48 TOTAL:      12743,   0% FULL
  PHYSICAL VERSION: BC9F02EE 03D48CA2
  LOGICAL  VERSION: BC9F02EE 03D48CA2
  SYSTEM-MANAGED PROCESS LEVEL: 8
  DISPOSITION    : DELETE
  ACCESS TIME    : 0
  MAX CONNECTIONS: 32
  # CONNECTIONS  : 4

 CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME  ASID STATE
 ---------------- -- -------- -------- -------- ---- -------
 IXGLOGR_SC63     01 00010266 SC63     IXGLOGR  0015 ACTIVE
 IXGLOGR_SC64     02 000200FC SC64     IXGLOGR  0016 ACTIVE
 IXGLOGR_SC65     03 0003010E SC65     IXGLOGR  0017 ACTIVE
 IXGLOGR_SC70     04 0004003B SC70     IXGLOGR  0017 ACTIVE
```

Figure 3-15   Display information about a specific RRS structure

3.12 ATRQSRV batch utility support



- ❑ RRS information can be especially useful in problem determination
- ❑ Another method for capturing RRS information is by using the ATRQSRV batch utility
  - ➢ List all resource managers
  - ➢ List all active units of recovery (URs)
- ❑ Can invoke ATRQSRV from REXX
- ❑ Unregister resource managers

Figure 3-16 ATRQSRV batch utility support

ATRQSRV batch utility

Currently there are three ways to unregister RMs:

- RRS ISPF panels
- Applications via the updated ATRSRV interface
- JCL via the ATRQSRV batch utility

Figure 3-17 shows a sample job for obtaining RRS RM information.

```
//LUTZRRS  JOB ,'KUEHNER',NOTIFY=&SYSUID,
//    CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
//LISTATR  EXEC PGM=ATRQSRV
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  RMINFO  RMNAME(IGWTV070020071121456262931740000)
//
```

Figure 3-17 Sample JCL for ATRQSRV use

List all resource managers

The following JCL can be used to list all resource managers:

```
//LISTRM      JOB
//STEP1       EXEC PGM=ATRQSRV
//SYSPRINT    DD  SYSOUT=*
//SYSIN       DD  *
  RMINFO
```

List all active units of recovery (URs)

The following JCL can be used to list all active recovery units:

```
//LISTUR      JOB
//STEP1       EXEC PGM=ATRQSRV
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD *
  URINFO
```

Invoke ATRQSRV from REXX

The following code shows an example of invoking ATRQSRV from REXX. The output of the sample ATRQSRV is done using the variable `atrprint`.

```
/* REXX ***/* function : issue RRS functions
address tso "alloc f(SYSIN)    new"      /* allocate temporarly     */
                                         /*  dataset for SYSIN       */
address tso "alloc f(SYSPRINT) new"      /* allocate temporarly      */
                                         /*  dataset for SYSPRINT    */
queue 'SYSINFO'                          /* pass function to ATRQSRV */
                                         /*                          */
address tso 'execio 1 diskw SYSIN (finis' /* write SYSIN control     */
                                         /*  statement to temporarly */
                                         /*   SYSIN file             */
address tso 'ATRQSRV'                    /* invoke ATRQSRV service   */
                                         /*                          */
address tso 'execio * diskr SYSPRINT (stem atrprint.'
                                         /* read the program output  */
do i=1 to atrprint.0                     /* print or manage the      */
 say atrprint.i                          /*  output                  */
end                                      /*                          */
                                         /*                          */
address tso "free  f(SYSPRINT)"          /* deallocate SYSPRINT      */
address tso "free  f(SYSIN)"             /* deallocate SYSIN         */
exit                                     /* good buy                 */
```

Figure 3-18 Invoking ATRQSRV from REXX

Resource manager unregister

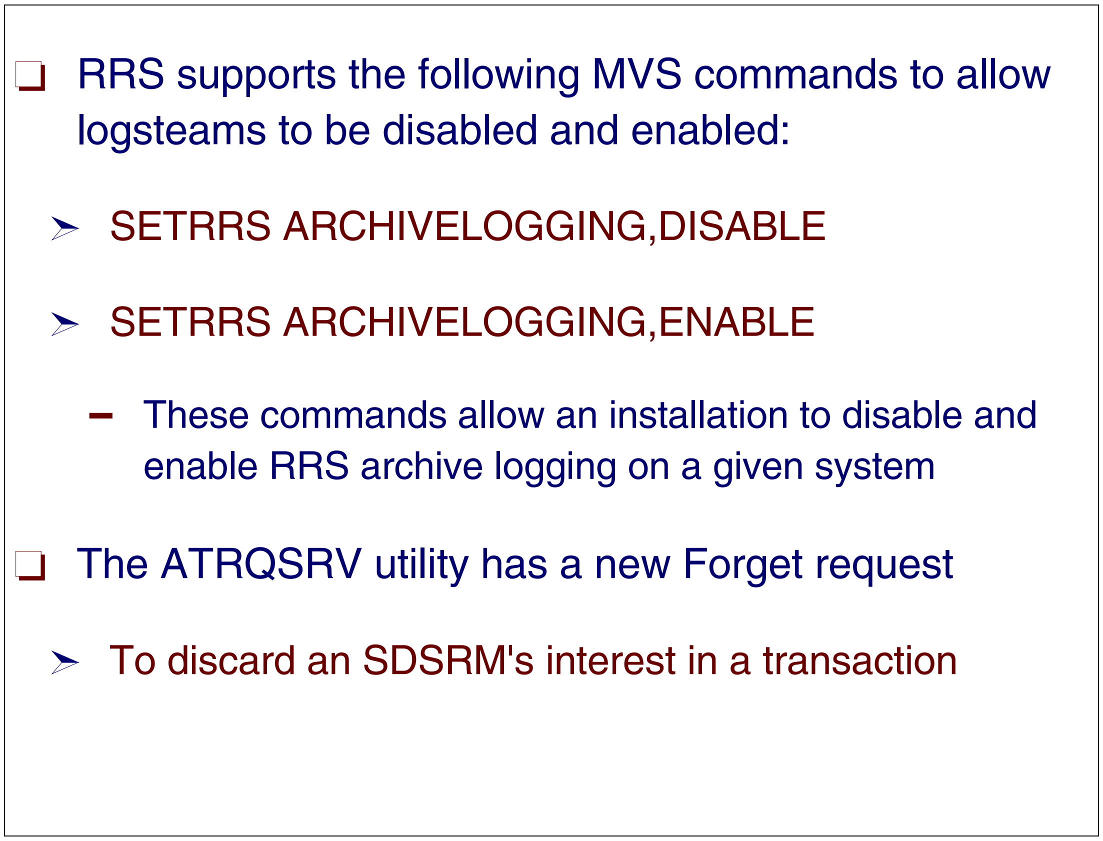
Resource manager unregister processing can be requested by one of the following means:

- Exit back-end processing as a result of RRS detecting an RM exit failed exit failure
- Registration services unregister exit (ATRBMUNR)

In either case, RRS performs the RM unregister processing to mark all UR interests for the failing RM as failed and unset the RM's RRS exits, as well as the RM unregistration processing with the registration services. If RRS experiences an internal failure before completing the RM unset processing, RM could be left in an unregistered state with registration services but still set with RRS. This situation cannot currently be resolved without recycling RRS and its resource managers, which is undesirable.

Instead, use the **`SETRRS CANCEL`** function to recycle the RRS address space. From the RRS ISPF panels Option 2, Display Resource Manager information, use the **`UNREGISTER RM`** command to unregister the resource manager with RRS.

3.13 RRS archive logging and ATRQSRV utility



- RRS supports the following MVS commands to allow logsteams to be disabled and enabled:
  - SETRRS ARCHIVELOGGING,DISABLE
  - SETRRS ARCHIVELOGGING,ENABLE
    - These commands allow an installation to disable and enable RRS archive logging on a given system
- The ATRQSRV utility has a new Forget request
  - To discard an SDSRM's interest in a transaction

Figure 3-19 Archive logging commands and ATQQSRV utility request

Archive logging commands

It is possible to experience severe performance impact of running RRS with the archive log stream. It is possible to run without it by deleting the archive log before starting any RRS image in the logging group or when RRS is operational.

After successfully defining the logstream, you can activate the archive logstream using the **SETRRS** command. A successful activation message ATR175I is issued.

The following command, **SETRRS ARCHIVELOGGING**, can be used to disable or enable RRS archive logging on a system:

```
SETRRS ARCHIVELOGGING,[DISABLE | ENABLE]
```

Where:

DISABLE	Indicates the system is to disable RRS archive logging for the subsequent transactions. RRS will stop writing the transaction completion records to the archive log and disconnect from the archive log stream.
ENABLE	Indicates the system is to enable RRS archive logging for the subsequent transactions. RRS will connect to the archive log stream and start writing the transaction completion records to the archive log. The ENABLE request is done by asynchronous processing and might take as long as 15 seconds before the Archive Log Stream is connected and the enable message (ATR175I) is issued.

4

Global resource serialization (GRS)

In a multitasking, multiprocessing environment, resource serialization is the technique used to coordinate access to resources that are used by more than one application task. When multiple users share data, a way to control access to that data is needed. Users that update data, for example, need exclusive access to that data. If several users try to update the same data at the same time, the result is data corruption. In contrast, users who only read data can safely access the same data at the same time.

Global Resource Serialization (GRS) is a z/OS component that offers the control needed to ensure the integrity of resources in a multisystem sysplex environment. Combining the z/OS systems that access shared resources into a global resource serialization complex enables you to serialize resources across multiple systems.

This chapter presents an overview of GRS covering the following topics:

- GRS benefits
- Resource access and GRS queue
- GRS services: enqueue, dequeue, and reserve
- GRS configurations: ring and star topology
- Migrating to a GRS star configuration
- Operational changes
- GRS exits
- GRS operator commands

4.1 GRS introduction

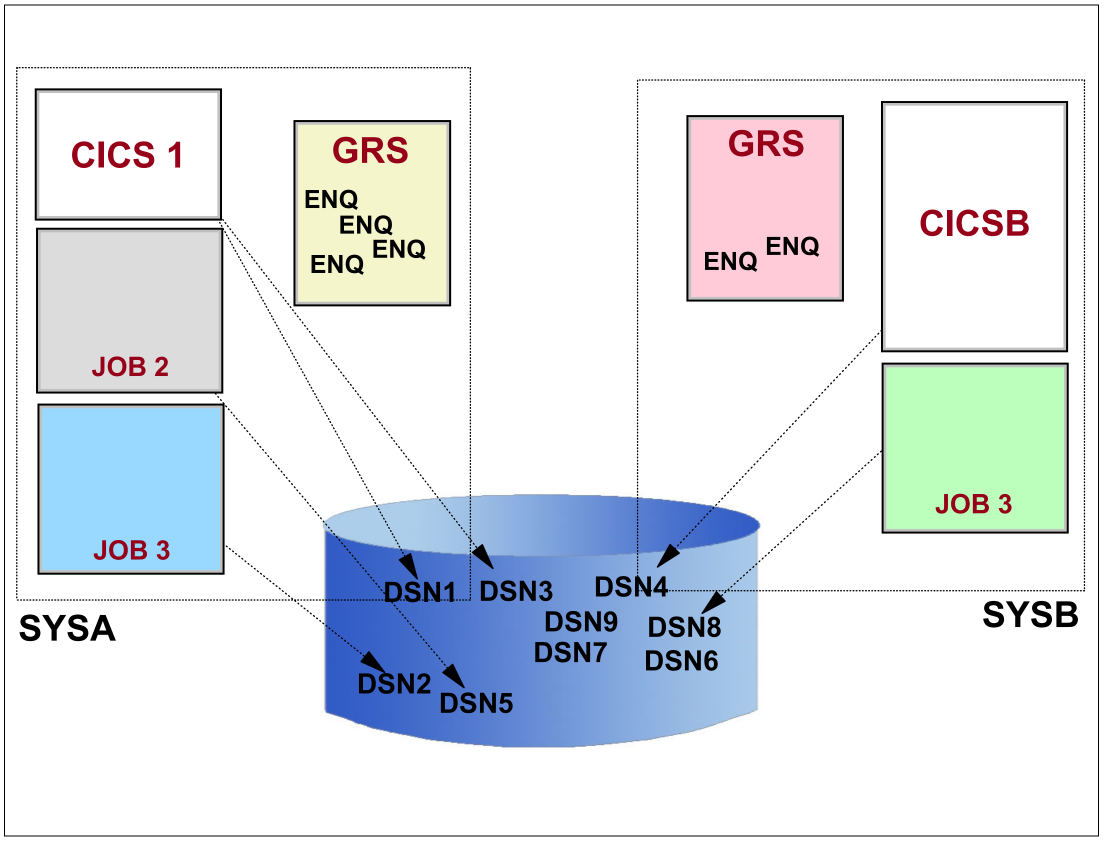


Figure 4-1 GRS introduction

GRS introduction

In a multisystem sysplex, z/OS provides a set of services that applications and subsystems can exploit. Multisystem applications reside on more than one z/OS system and give a single system image to the users. This requires serialization services that extend across z/OS systems.

GRS provides the services to ensure the integrity of resources in a multisystem environment. Combining the systems that access shared resources into a global resource serialization complex enables serialization across multiple systems.

GRS complex

In a GRS complex, programs can serialize access to data sets on shared DASD volumes at the data set level rather than at the DASD volume level. 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. Because GRS enables jobs to serialize resources at the data set level, it can reduce contention for these resources and minimize the chance of an interlock occurring between systems.

GRS ENQ

An ENQ (enqueue) occurs when GRS schedules use of a resource by a task. There are several techniques to implement serialization (or semaphores) among dispatchable units (TCBs and SRBs) in z/OS such as: enqueue (ENQ), z/OS locks, and z/OS latches. GRS deals with resources protected by ENQs only.

4.2 Resource access and GRS queue

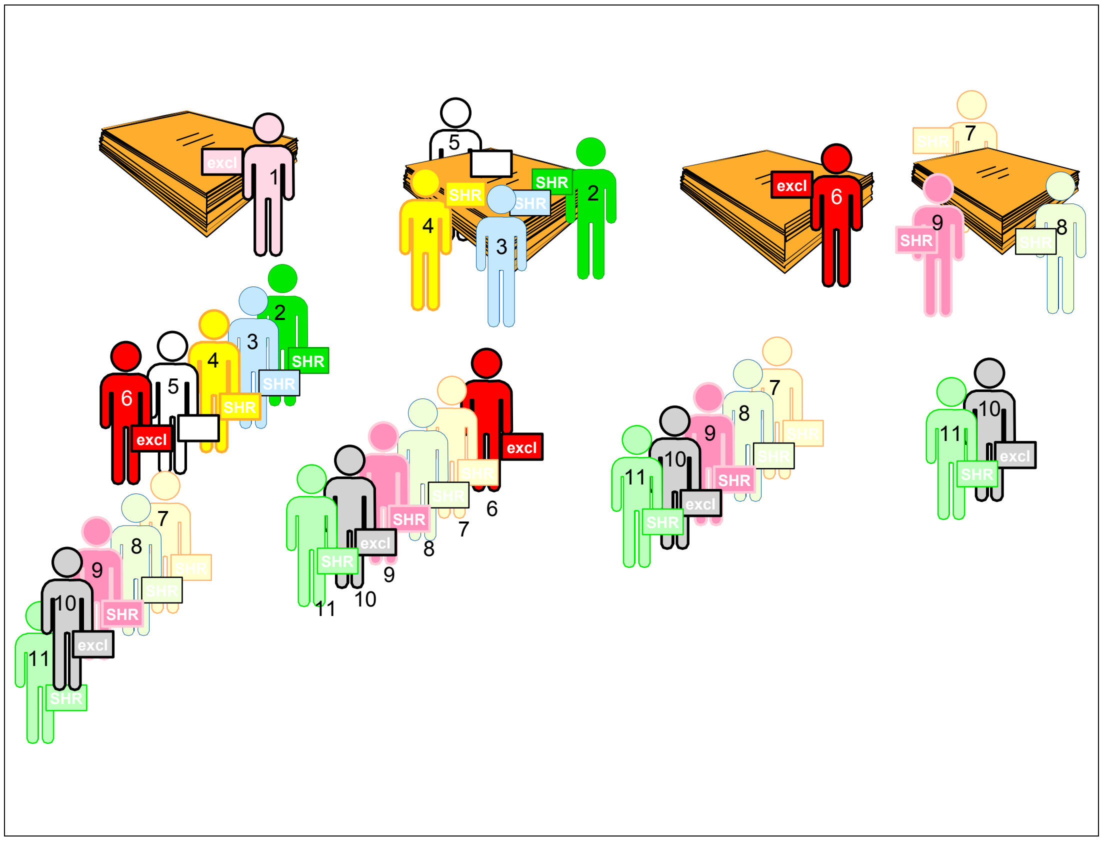


Figure 4-2 Resource access and GRS queue

Resource access and GRS queue

When a task (executing a program) requests access to a serially reusable resource, the access can be requested as *exclusive* or *shared*. When GRS grants shared access to a resource, no exclusive users are getting access to the resource simultaneously. Likewise, when GRS grants exclusive access to a resource, all other requestors for the resource wait until the exclusive requestor frees the resource.

Because global resource serialization 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. This 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. A global resource serialization complex also allows serialization of shared logical resources – resources that might not be directly associated with a data set or DASD volumes. An ENQ with a scope of SYSTEMS might be used to synchronize processing in multisystem applications.

GRS resource requests

GRS manages the requests to a resource on a first-in-first-out (fifo) basis. For better understanding of how GRS manages a queue, consider the example shown in Figure 4-2:

1. Instance T1, user1 has exclusive access to the resource, all others wait in the queue.

2. When user1 releases the resource, the next user in the queue (user2) receives access to the resource. User2 requested shared access. The following users in the queue requesting shared access also get access to the resource, as shown in the T2 instance. The queue stops at the next user waiting for exclusive access, user6. Note that users wanting shared access behind user6 still wait in the queue. This behavior is to avoid the aging of user6 in the queue. Then, just one exclusive requester can block the full queue.
3. When the last shared user releases the resource, user6 (T3) receives access. Because user6 requested exclusive access, only user6 uses the resource. It doesn't matter whether the access requested by the next user in the queue is shared or exclusive.
4. When user6 releases the resource, the next user in the queue (user7) receives access. Because the requested access is shared, GRS follows the queue, granting shared access, stopping at the next user requesting exclusive access, user10, as shown in T4.

4.3 Resource scope: Step, system, systems

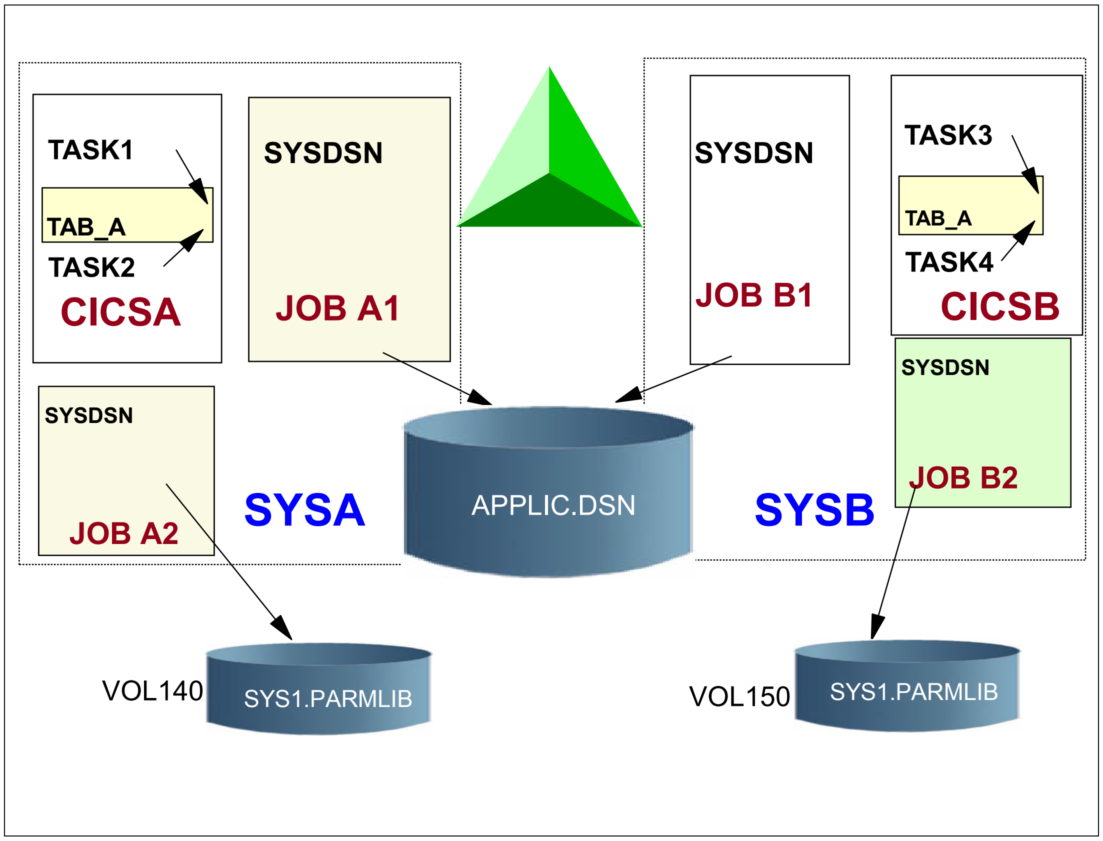


Figure 4-3 Resource scope: Step, system, systems

Resource scope: Step, system, systems

When requesting access to a resource, the program must provide the scope of the resource. The scope can be: STEP, SYSTEM, and SYSTEMS. To avoid data exposure or affecting system performance, the correct resource scope must be given to GRS when requesting serialized access to a resource. Figure 4-3 illustrates what happens when resource serialization occurs according to the scope requested:

1. Suppose TASK1, running in the CICSA address space, wants to alter an internal area named TAB_A, which is not used at the moment of the request. TAB_A is also accessible by other tasks running in the same address space, CICSA. To ensure data integrity, TASK1 must request exclusive access to TAB_A. A different resource, with the same name, exists also in another address space (CICSB) in system SYSB. Let's see what happens with TAB_A with different scope scenarios:

STEP	GRS prevents any other task from accessing TAB_A in CICSA ASID. If TASK2 requests access, TASK2 must wait until TASK1 *releases* TAB_A.
SYSTEM	GRS prevents any other task in any address spaces in SYSA from using a resource with the same name.
SYSTEMS	GRS prevents any other task in any address spaces in the Parallel Sysplex from using a resource with the name TAB_A. Then, if any task in CICSB address space requests access to TAB_A in the same address space, GRS enqueues the task until TASK1 releases TAB_A in CICSA ASID.

2. Suppose JOB A1, in SYSA, requests exclusive access to data set APPLIC.DSN, which is also accessible by other systems. To protect data set integrity, the resource access must be SYSTEMS.
3. Suppose JOB A2 requests access to data set SYS1.PARMLIB on VOL140, cataloged only in SYSA. At the same time JOB B2, in SYSB, requests access to SYS1.PARMLIB on VOL150, cataloged only on SYSB. To avoid any impact, *both must* request their access with scope SYSTEM. Otherwise the second request would wait even if different resources are accessed.

Local resource

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

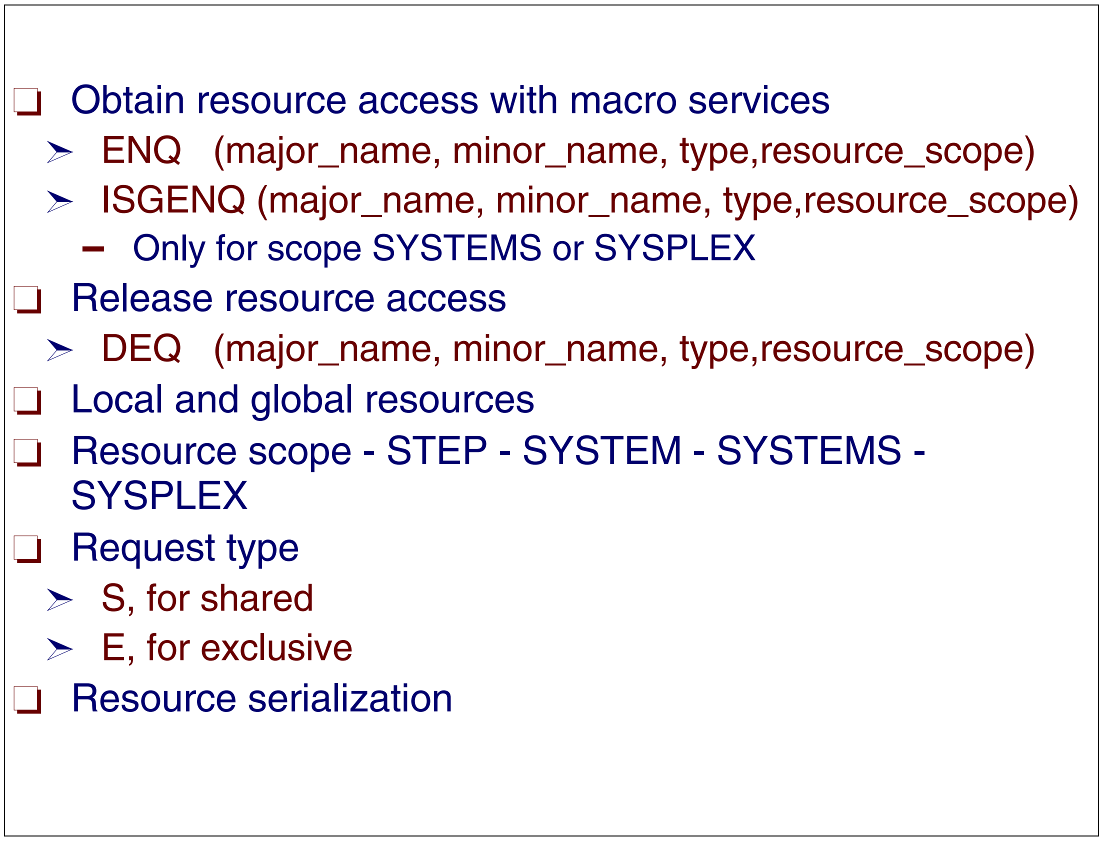
Global resource

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

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. There is an ENQ associated with the volume RESERVE; this is required because a volume is reserved by the z/OS image and not the requester's unit of work. As such, the ENQ is used to serialize across the requesters from the same system.

4.4 GRS macro services



- Obtain resource access with macro services
  - ENQ (major_name, minor_name, type,resource_scope)
  - ISGENQ (major_name, minor_name, type,resource_scope)
    - Only for scope SYSTEMS or SYSPLEX
- Release resource access
  - DEQ (major_name, minor_name, type,resource_scope)
- Local and global resources
- Resource scope - STEP - SYSTEM - SYSTEMS - SYSPLEX
- Request type
  - S, for shared
  - E, for exclusive
- Resource serialization

Figure 4-4 GRS macro services

GRS macro services

GRS services are available throughout assembler macros. The macros are: ENQ, DEQ, ISGENQ, and RESERVE. The ENQ API issuer provides a qname, rname, and scope to identify the resource to serialize; this is all that the DISPLAY GRS command can use to present information about the resource. In many cases, it is hard for the operator or systems programmer to understand what these values represent. Global resource serialization provides the ISGDGRSRES installation exit to allow the application to add additional information for a given resource on the DISPLAY GRS output.

ISQENQ macro

The ISGENQ macro combines the serialization abilities of the ENQ, DEQ, and RESERVE macros. ISGENQ supports AMODE 31 and 64 in primary or AR ASC mode.

With the ISGENQ macro you can:

- Obtain a single resource or multiple resources with or without associated device reserves.
- Change the status of a single existing ISGENQ request or multiple existing ISGENQ requests.
- Release serialization from a single or multiple ISGENQ requests.
- Test an obtain request.
- Reserve a DASD volume.

Local and global resource

When an application uses a macro (ISGENQ, ENQ, DEQ, and RESERVE) to serialize resources, global resource serialization uses the RNL, various global resource serialization installation exits, and the scope on the macro to determine whether a resource is a local resource or a global resource, as follows:

- 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 (with the exception of resources serialized with the RESERVE macro), regardless of scope (STEP, SYSTEM, SYSTEMS), are local resources.
- A global resource is a resource requested with a scope of SYSTEMS or SYSPLEX. It is serialized among all systems in the complex. The ISGENQ, ENQ, DEQ, and RESERVE macros identify a resource by its symbolic name. The symbolic name has three parts:
  - Major name: qname
  - Minor name: rname
  - Scope: The scope is one of the following:

    STEP - SYSTEM - SYSTEMS - SYSPLEX

    The *resource_scope* parameter determines which other tasks, address spaces, or systems can use the resource.

Resource serialization

Resource serialization offers a naming convention to identify the resources. All programs (z/OS inclusive) that share the resources must use the major_name, minor_name, and scope value consistently.

A resource could have a name of APPL01,MASTER,SYSTEM. The major name (qname) is APPL01, the minor name (rname) is MASTER, and the scope is SYSTEM. Global resource serialization identifies each resource by its entire symbolic name. That means a resource that is specified as A,B,SYSTEMS is considered a different resource from A,B,SYSTEM or A,B,STEP because the scope of each resource is different. In addition, the ISGENQ macro uses the scope of SYSTEMS and the scope of SYSPLEX interchangeably, so a resource with the symbolic name of A.B.SYSTEMS would be the same as a resource with the name A.B.SYSPLEX.

Obtaining a resource is done through the ENQ (or ISGENQ) macro, and requests (using a name) either shared or exclusive control of one or more serially reusable resources. Once control of a resource has been assigned to a task, it remains with that task until the DEQ macro is issued. If any of the resources are not available, the task is placed in a wait condition until all of the requested resources are available. If the resource is granted, the task is not placed in wait state and the next instruction in the program requesting the ENQ is naturally executed,

Release resource access

The issuing of the DEQ macro (or ISGDEQ) by the resource owner releases the resource from the task. Also, when a task ends abnormally the recovery termination manager (a z/OS component) issues automatically DEQs to all the ENQ resources held by the abending task.

Note: In addition, the ISGENQ macro uses the scope of SYSTEMS and the scope of SYSPLEX interchangeably, so a resource with the symbolic name of A.B.SYSTEMS would be the same as a resource with the name A.B.SYSPLEX. z/OS services use standard major names for resource serialization. For data set access, the major name is SYSDSN, the minor name is the data set name.

4.5 RESERVE macro service

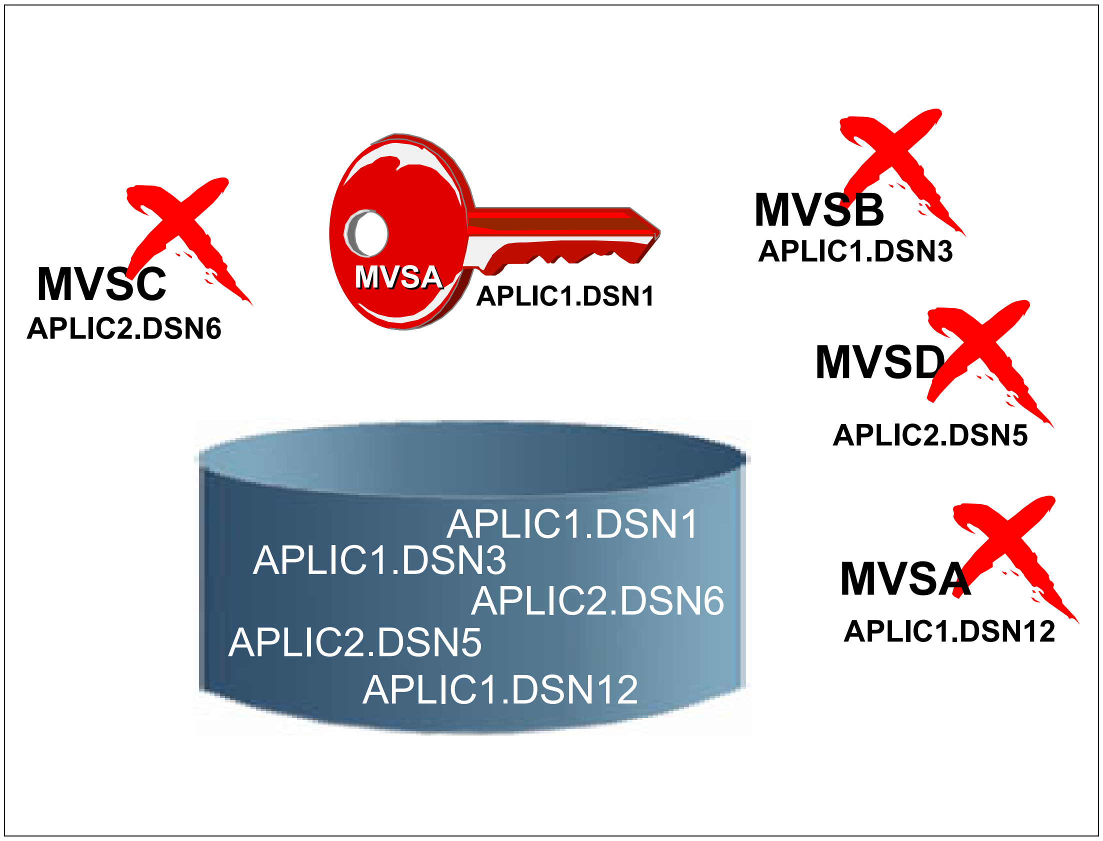


Figure 4-5 RESERVE macro service

RESERVE macro service

The RESERVE macro reserves a DASD device for use by a particular system. The macro must be issued by each task needing access to a shared DASD. The RESERVE macro locks out other systems sharing the device. Reserve is a combination of GRS ENQ (to protect tasks of the same z/OS) and hardware I/O reserve that is implemented by the DASD I/O controller.

The hardware I/O reserve actually occurs when the first I/O originated in the z/OS, where the reserve was originated done to the device after the RESERVE macro is issued. The Synchronous reserve feature (SYNCHRES option), allows an installation to specify whether the z/OS should obtain a hardware reserve for a device prior to granting a global resource serialization ENQ. This option might prevent jobs that have a delay between a hardware reserve request being issued and the first I/O operation to the device. Prior to the implementation of the SYNCHRES option, the opportunity for a deadlock situation was more likely to occur.

The RESERVE macro has by default a scope of SYSTEMS, contrary to ENQ and DEQ, which can have either STEP, SYSTEM, or SYSTEMS. When the reserving program no longer needs the reserved device, a DEQ must be issued to release and free the resource.

Collateral effects of RESERVE

The RESERVE macro serializes access to a resource (a data set on a shared DASD volume) by obtaining control of the entire volume. This is due to the hardware reserve function done by the DASD controller. Then, jobs on other systems are prevented from using any data set on this volume. Serializing access to data sets on shared DASD volumes by RESERVE macro protects the resource, although it might create several critical problems:

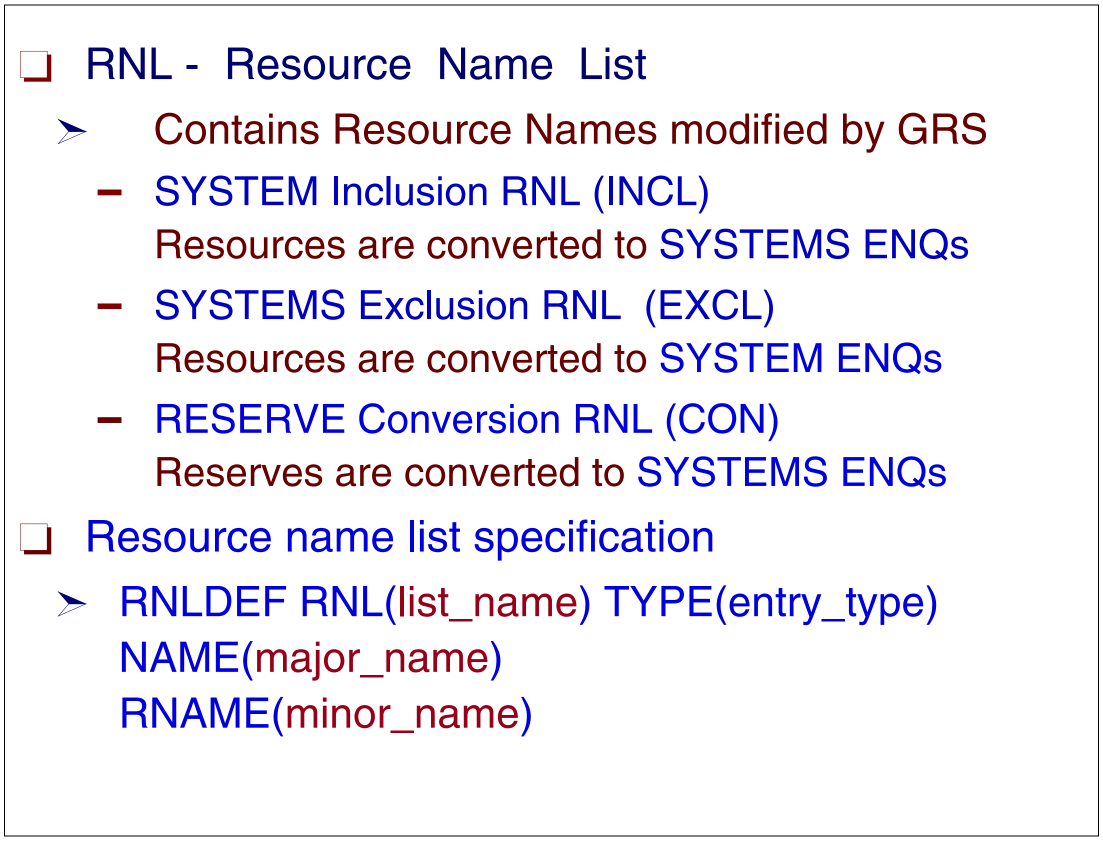
- When a task on one system has issued a RESERVE macro to obtain control of a data set, no programs on other systems can access any other data set on that volume. Protecting the integrity of the resource means delay of jobs that need the volume but do not need that specific resource, as well as delay 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. This scenario gets worse because now you may have 3390 volumes with a very big capacity, such as 27 GB.

  Because the reserve ties up an entire volume, it increases the chance of an interlock (also called a deadlock) occurring between two tasks on different systems, as well as the chance that the lock expands 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 DEQ macros are processed for all of those reserves. No other system can use the device until the reserving system releases the device. This is also called a *long* reserve.
- 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.

An installation using GRS can override RESERVE macros without changing existing programs, using GRS Resource Name Conversion Lists in SYS1.PARMLIB. 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. Using the conversion list the RESERVE macro behaves as a SYSTEMS ENQ, or in other word, it will be GRS the one in charge of the data set serialization.

The RESERVE macro is only recommended to serialize data sets among z/OS systems located in different sysplexes. In this case the current GRS version cannot protect the data set resource.

4.6 Resource name list (RNL)



- RNL - Resource Name List
  - Contains Resource Names modified by GRS
    - SYSTEM Inclusion RNL (INCL)
      Resources are converted to SYSTEMS ENQs
    - SYSTEMS Exclusion RNL (EXCL)
      Resources are converted to SYSTEM ENQs
    - RESERVE Conversion RNL (CON)
      Reserves are converted to SYSTEMS ENQs
- Resource name list specification
  - RNLDEF RNL(list_name) TYPE(entry_type)
    NAME(major_name)
    RNAME(minor_name)

Figure 4-6 Resource name list (RNL)

Resource name list (RNL)

Some environments may require a different resource scope from that specified by the application through ENQ and ISGENQ macros. GRS allows an installation to change the scope of a resource, without changing the application, through the use of lists named Resource Name List (RNL).

There are three RNLs available:

SYSTEM INCLUSION RNL	Lists resources for requests with scope SYSTEM in the macros to be converted to scope SYSTEMS.
SYSTEMS EXCLUSION RNL	Lists resources for requests with scope SYSTEMS in the macros to be converted to scope SYSTEM.
RESERVE CONVERSION RNL	Lists resources for RESERVE requests for which the hardware reserve is to be suppressed and the requests are treated as ENQ requests with scope SYSTEMS.

Resource name list specification

By placing the name of a resource in the appropriate RNL, GRS knows how to handle this specific resource. The RNLs enable you to build a global resource serialization complex without first having to change your existing programs.

A very important comment is that all the RNLS have a sysplex scope, that is, all the z/OS systems in the sysplex must share the same set of RNLs. Use the RNLDEF statement to define an entry in the RNL, as shown in Figure 4-6. Each RNL entry indicates whether the name is generic, specific, or pattern, as follows:

- Specific resource name entry matches a search argument when they are exactly same.
- Generic resource name entry is a portion of a resource name. A match occurs whenever the specified portion of generic resource name entry matches the same portion of the input search argument.
- Installation can use RNL wildcarding to pattern match resource names; you can use an asterisk to match on any number of characters up to the length of respective major_name (up to 8) or minor_name (up to 255) or use a (?) to match on a single character.

  Using wildcarding, you can convert all RESERVEs to global ENQs with a single statement. To use RNL wildcarding you have to specify TYPE(PATTERN) in the RNLDEF statement.

You can use the `DISPLAY GRS,RNL=xxxx` command to display all the lists or just one of them. Also with `SET GRSRNL=xxxx` is possible to change dynamically the RNLs in the sysplex.

When an application uses the ENQ, ISGENQ, DEQ, or RESERVE macros to serialize resources, GRS uses RNLs and the scope on the ENQ, DEQ, or RESERVE macro to determine whether a resource is a local resource or a global resource.

In the IEASYSxx, there is the parameter GRSRNL=EXCLUDE meaning that all the resources are going to be treated by GRS as local. The purpose of such is to allow customers to have other serialization product besides GRS. However, even in this case GRS must be active because certain z/OS components may use the RNL =NO option in the ENQ (or ISGENQ) macro to force the global serialization through GRS.

Important: RNLs for every system in the GRS complex must be identical; they must contain the same resource name entries, and these must appear in the same order. During initialization, global resource serialization checks to make sure that the RNLs on each system are identical. If they are different, global resource serialization does not allow the system to join the complex.

GRS exits

You can use ENQ/DEQ exits ISGGSIEX, ISGGSEEX, and ISGGRCEX to do the following:

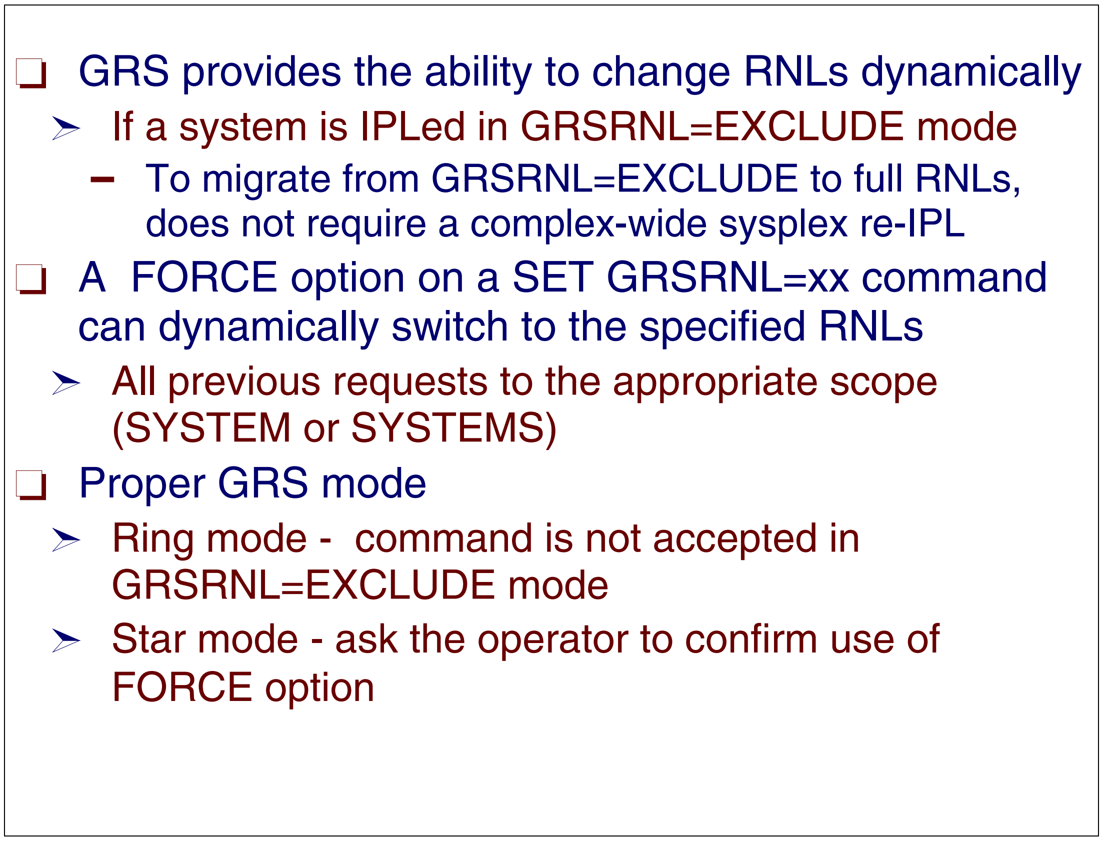
- Change resource names (major and minor names)
- Change the resource SCOPE
- Change the UCB address (for a RESERVE)
- Indicate to convert a RESERVE to ENQ or ENQ to a RESERVE (add a UCB specification)
- Indicate to bypass RNL processing

These exits are invoked only for SYSTEM and SYSTEMS requests. Important to note that they have a local scope, contrary of RNLs that are at sysplex level. They are required for example in the following scenario: All z/OS systems share the same SYS1.PARMLIB with the exception of one of them that has its own SYS.PARMLIB (with the same name). In this case the exit is activate in this z/OS to make such resource local.

GQSCAN macro service

The GQSCAN macro extracts information from Global Resource Serialization about contention (users and resources). It is used by performance monitors as RMF. It causes exchange of GRS messages through XCF links.

4.7 GRSRNL=EXCLUDE migration to full RNLs



- GRS provides the ability to change RNLs dynamically
  - If a system is IPLed in GRSRNL=EXCLUDE mode
    - To migrate from GRSRNL=EXCLUDE to full RNLs, does not require a complex-wide sysplex re-IPL
- A FORCE option on a SET GRSRNL=xx command can dynamically switch to the specified RNLs
  - All previous requests to the appropriate scope (SYSTEM or SYSTEMS)
- Proper GRS mode
  - Ring mode - command is not accepted in GRSRNL=EXCLUDE mode
  - Star mode - ask the operator to confirm use of FORCE option

Figure 4-7 GRSRNL=EXCLUDE option

GRSRNL=EXCLUDE option

Migrating from a GRSRNL=EXCLUDE environment to full RNLs does not require a sysplex-wide IPL for certain environments. A FORCE option for the **`SET GRSRNL=xx`** command processing can dynamically switch to the specified RNLs and move all previous requests to the appropriate scope (SYSTEM or SYSTEMS) as if the RNLs were in place when the original ENQ requests were issued. This feature eliminates the IPL of the entire sysplex when making this change.

Usage restrictions

The usage restrictions are enforced to ensure data integrity. The migration is canceled if any of these requirements are not met and GRSRNL=EXCLUDE remains in effect. Consider the following:

- When in GRS Ring mode, the ISG248I message is issued indicating that the SET GRSRNL command is not accepted in a GRSRNL=EXCLUDE environment.
- When in GRS Star mode, before the SET GRSRNL=xx command migration begins, message ISG880D prompts the user to confirm the use of the FORCE option, as follows:

```
ISG880D WARNING: GRSRNL=EXCLUDE IS IN USE. REPLYING FORCE WILL RESULT IN THE
USE OF SPECIFIED RNLS. REPLY C TO CANCEL
```

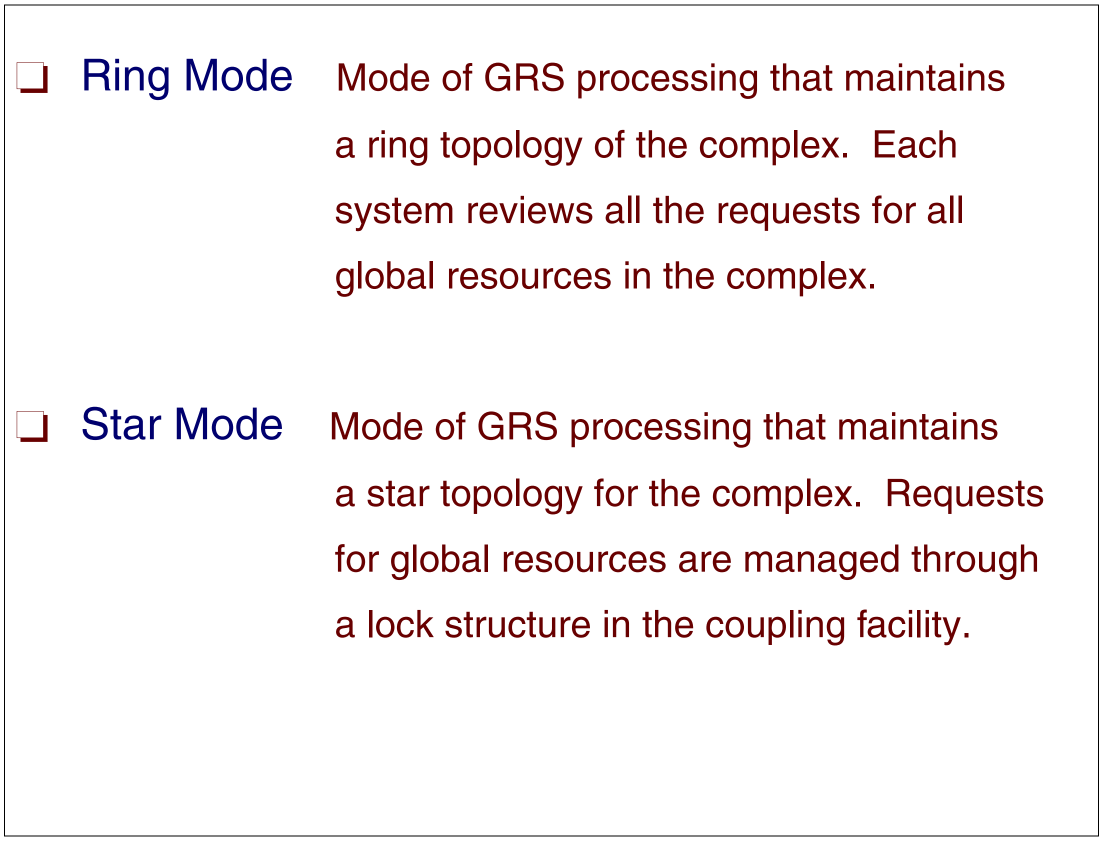
Usage considerations

The following restrictions are required for data integrity:

- Can only be issued in a single system GRS STAR sysplex with GRSRNL=EXCLUDE.
- No ISGNQXIT or ISGNQXITFAST exit routines can be active, nor requests that were affected by one.
- No ISGNQXITBATCH or ISGNQXITBATCHCND exit routines can be active.
- No local resource requests can exist that must become global where that global resource is already owned. This could result if some requests were issued with RNL=NO.
- No resources with a MASID request can exist where only some of the requesters will become global. This could result if some requests were issued with RNL=NO or if some requests were issued with different scopes.
- No RESERVEs can be held that were converted by the ISGNQXITBATCH or ISGNQXITBATCHCND exit that do not get converted by the new RNLs. This could result if a resource was managed globally by an alternative serialization product before the migration, but afterwards is to be serialized by a device RESERVE.

Note: Any errors in changing the RNL configuration can lead to deadlocks or data integrity errors. When doing migration to the specified RNLs, the only way to move back to GRSRNL=EXCLUDE is a sysplex-wide IPL. The FORCE option cannot be specified from the existing RNLs to another set of RNLs. Long-held resources can delay such a change indefinitely, and therefore a cancellation of jobs or even a sysplex-wide outage is required to end the job.

4.8 GRS configuration modes



- **Ring Mode** Mode of GRS processing that maintains a ring topology of the complex. Each system reviews all the requests for all global resources in the complex.
- **Star Mode** Mode of GRS processing that maintains a star topology for the complex. Requests for global resources are managed through a lock structure in the coupling facility.

Figure 4-8 GRS configuration modes

GRS configuration modes

Starting with MVS/ESA Version 4, it is required that GRS is active in the sysplex. Any multisystem sysplex is thus required to activate global serialization for all members of the sysplex. Any global ENQ implies that all GRS address spaces must be aware of such ENQ. There are two GRS configuration modes available, which make this awareness possible.

GRS ring mode

When GRS is configured in *ring mode*, each system in GRS complex maintains a list for all requests for global resources in the complex, through the manipulation of global ENQ/DEQ information that flows through GRSs in a ring topology. This configuration is old and it is not recommended due to performance and availability reasons.

GRS star mode

You can define a GRS complex in a *star mode* configuration. It is still possible to run all z/OS releases using the ring configuration. The main change in the star mode method of processing requests for global ENQs resources is that they are managed through a coupling facility lock structure, and that each GRS maintains only its own global requests. The star configuration has positive effects on sysplex growth, response time for global requests, continuous availability and processor and storage resources consumption.

If you build a multisystem sysplex, you automatically build a GRS complex with it. When a system joins the sysplex, the system also joins the global resource serialization complex.

4.9 GRS ring configuration

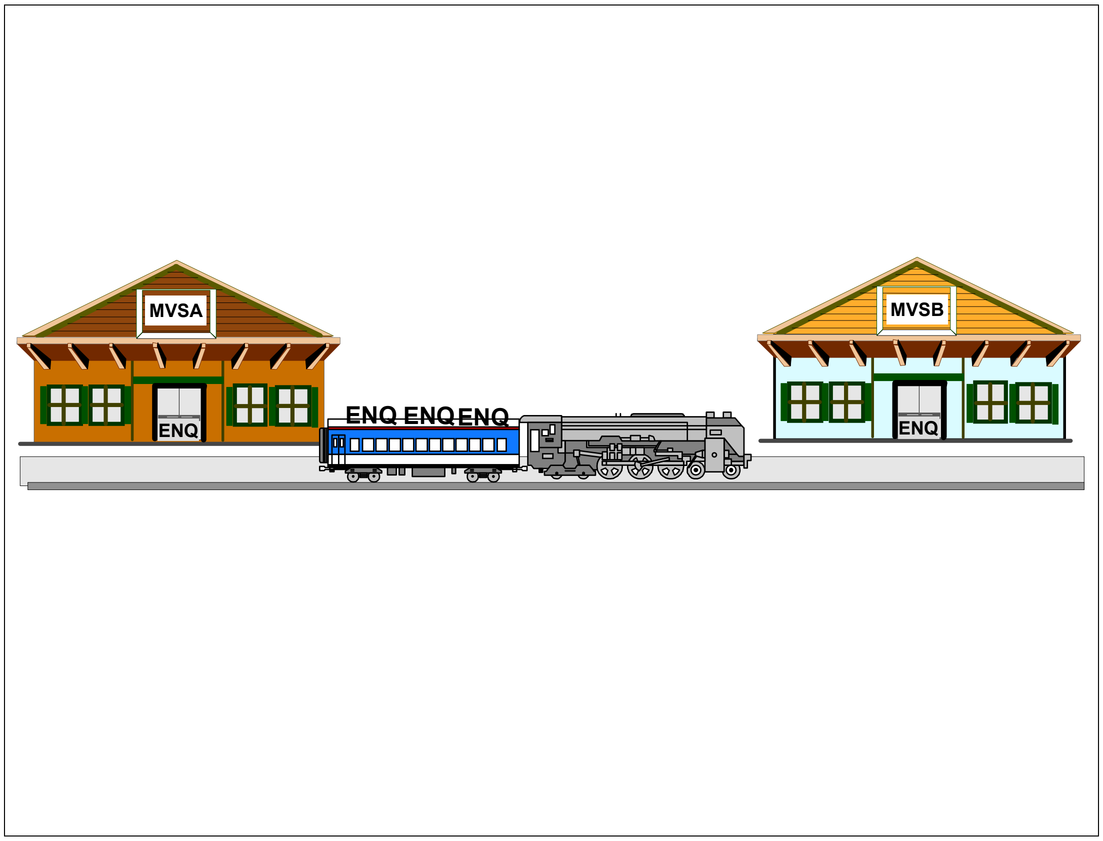


Figure 4-9 GRS ring configuration

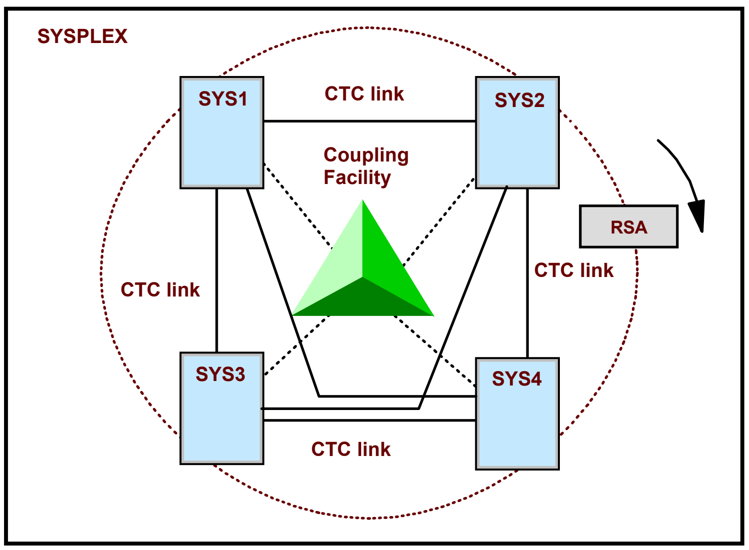
GRS ring configuration

The first implementation of GRS was a GRS ring mode and was the only method used for serializing requests for global resources. The ring consists of one or more GRS systems connected to each other by communication links. The links are used to pass information about requests for global resources from one GRS to another in the complex. Requests are made by passing a message or token, called the ring system authority message (RSA-message), between systems in a ring fashion. When a system receives the RSA, it inserts global ENQ and DEQ information and passes it along to the next GRS to copy. It also makes a copy of the global ENQ/DEQ information that was inserted by other GRS address spaces. When the RSA returns to the originating system, it knows that all other GRS systems have seen its request, so the request is removed. The GRS can now process those requests by adding them to the global resource queues, and can now determine which jobs own resources, and which jobs must wait for resources owned by other jobs. Each system takes this information and updates its own global queues.

Systems participating in a ring are dependent on each other to perform global resource serialization processing. For this reason, the following areas are all adversely affected as the number of systems in a ring increases:

- Storage consumption
- Processing capacity
- Response time
- CPU consumption
- Availability/Recovery time

4.10 GRS ring topology



- ❑ Connected via XCF managed path or GRS managed CTCs
- ❑ Time slicing via RSA-message
- ❑ Each system maintains complex-wide view of data

Figure 4-10 GRS ring topology

GRS ring topology

A global resource serialization ring complex consists of one 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.

Figure 4-10 shows a four-system global resource serialization ring complex. When all four systems in the complex are actively using global resource serialization, the complex and the ring are the same. A GRS complex consists of one or more systems connected to each other in a ring configuration by using:

- XCF communication paths (CTCs) and signalling paths through a coupling facility, for the z/OS systems that belong to a sysplex. If all z/OS systems belong to a sysplex, then GRS uses XCF services to communicate along the GRS ring; this configuration is called a sysplex matching complex.
- GRS-managed channel-to-channel (CTCs) adapters; for the z/OS systems that do not belong to a sysplex, this configuration is called a mixed complex.

The complex shown has a communication link between each system and every other system; such a complex is a fully-connected complex. A sysplex requires full connectivity between systems. Therefore, when the sysplex and the complex are the same, the complex has full connectivity. Although a mixed complex might not be fully connected, a fully-connected complex 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.

The concept of the global resource serialization ring is 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).

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. 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 requesters at the same time.

4.11 GRS star configuration

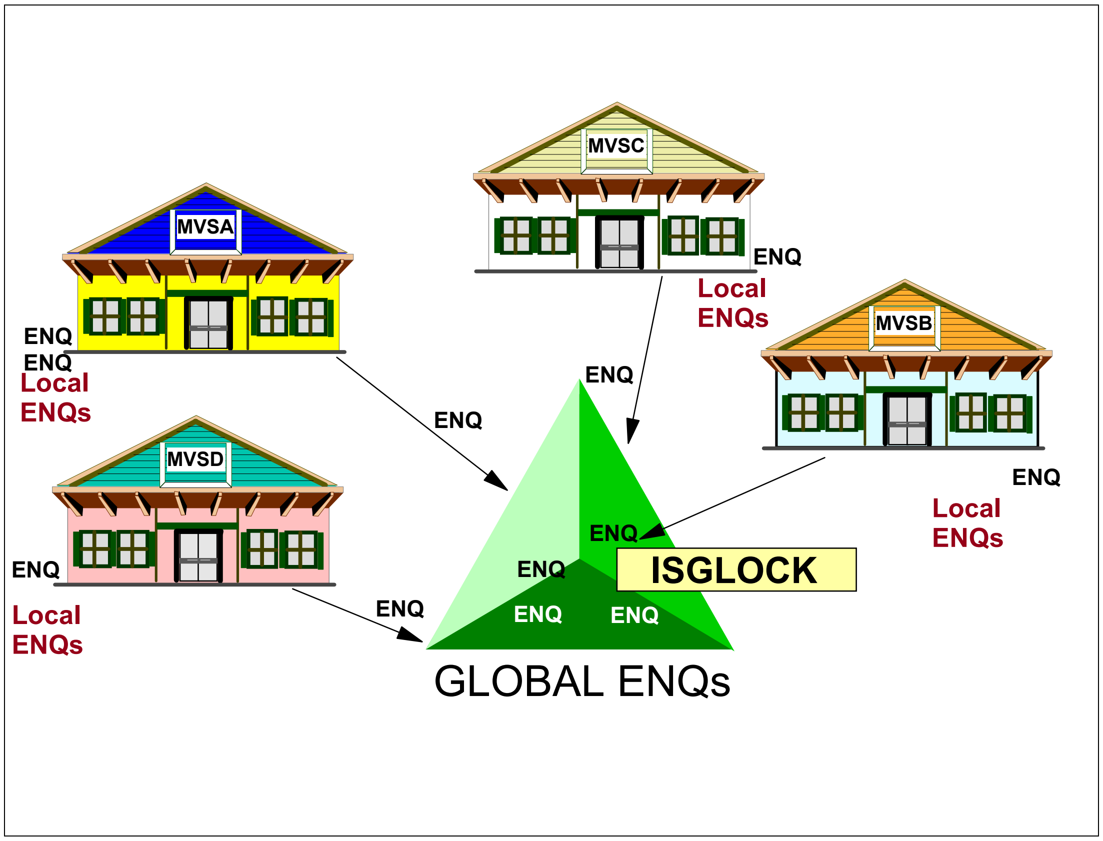


Figure 4-11 GRS star configuration

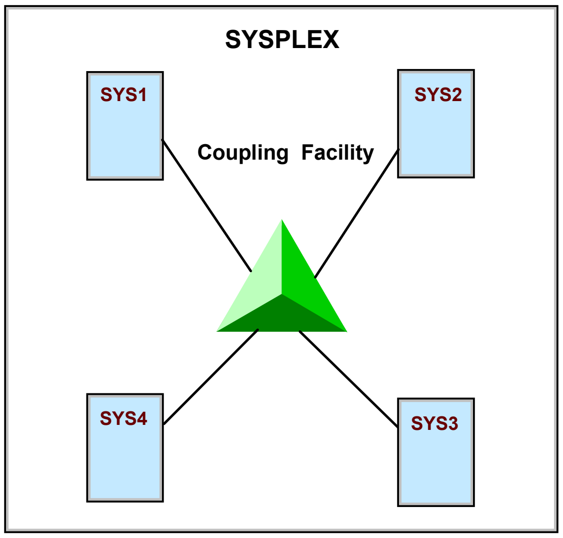
GRS star configuration

The star method of serializing global resources is built around a coupling facility in which the global resource serialization lock structure, ISGLOCK, resides. In a star complex, when an ISGENQ, ENQ, DEQ, or RESERVE macro is issued for a global resource, GRS uses information in the ISGLOCK structure to coordinate resource allocation across all z/OS systems in the sysplex.

In a star complex, GRS requires the use of the coupling facility for the serialization lock structure ISGLOCK. The coupling facility makes data sharing possible by allowing data to be accessed throughout the sysplex by ensuring the data remain consistent among each z/OS in the sysplex. The coupling facility provides the medium (global intelligent memory) in GRS to share information about requests for global resources between z/OS systems. Communication links (CF links) are used to connect the coupling facility with the GRS address spaces in the global resource serialization star sysplex.

Note: If you only have one coupling facility, using the star method is not recommended. Loss of that coupling facility would cause all systems in the complex to go into a wait state. In a production environment you should define at least two coupling facilities and implement GRS star instead of GRS ring.

4.12 GRS star topology



- ❑ Connected via coupling facility lock structure
- ❑ Does not use time slicing as GRS ring does
- ❑ Each system maintains local view of data
- ❑ Global view of data maintened in the lock structure in the CF

Figure 4-12 GRS star topology

GRS star topology

The star method of serializing global resources is built around a coupling facility in which the global resource serialization lock structure, ISGLOCK, resides. In a star complex, when an ISGENQ, ENQ, DEQ, or RESERVE macro is issued for a global resource, MVS uses information in the ISGLOCK structure to coordinate resource allocation across all systems in the sysplex.

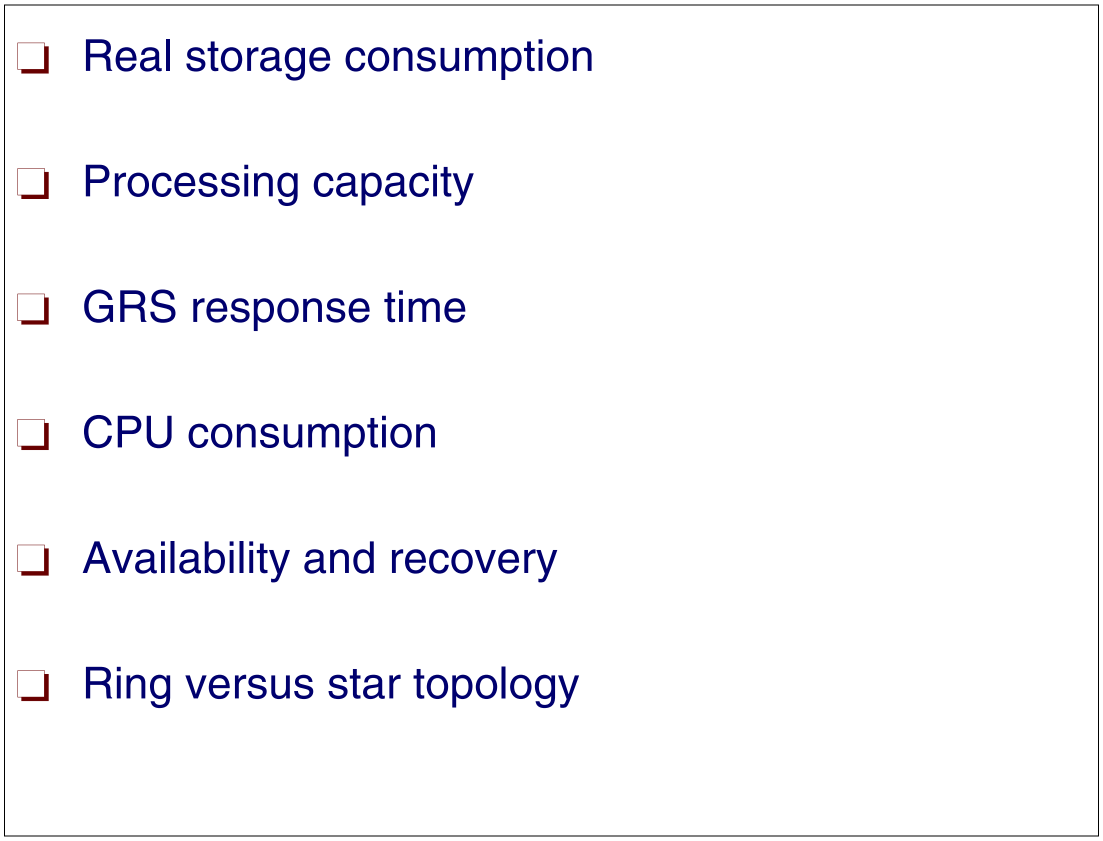
Coupling facility lock structures

In a star complex, global resource serialization requires the use of the coupling facility. The coupling facility makes data sharing possible by allowing data to be accessed throughout the sysplex by ensuring the data remain consistent among each system in the sysplex. The coupling facility provides the medium in global resource serialization to share information about requests for global resources between systems. Communication links are used to connect the coupling facility with the systems in the global resource serialization star sysplex.

GRS uses contention detection and management capability of the XES lock structure to determine and assign ownership of a particular global resource. Each system maintains only a local copy of its own global resources, and the GRS coupling facility lock structure has the overall image of all system global resources in use.

Figure 4-12 shows a GRS star complex, where the global ENQ/DEQ services use the coupling facility lock structure to manage and arbitrate the serialization requests, and the global GQSCAN service uses the XCF services to communicate across the sysplex.

4.13 GRS star highlights



- Real storage consumption
- Processing capacity
- GRS response time
- CPU consumption
- Availability and recovery
- Ring versus star topology

Figure 4-13 GRS star highlights

Central storage consumption

In a GRS ring configuration, each system in the ring maintains a queue of all global resource serialization requests for the entire sysplex. Because of the frequency of ENQ/DEQ processing, each system uses central storage that is proportional to the number of outstanding resource requests and the number of z/OS systems in the sysplex. Basically, in a GRS ring the amount of real storage consumed by GRS in each system goes up linearly with the number of z/OS systems in the sysplex. In the case of GRS star support, no GRS n the sysplex maintains a complete queue of all the global resource requests for the entire sysplex. Because of this approach, the central storage consumption by GRS for each system is governed only by number of requests from that z/OS.

Processing capacity

In a GRS ring, all global resource serialization requests are circulated around the ring in a single message buffer called the ring system authority (RSA). Basically, the RSA can be viewed as a floating data area that is shared by all systems in the sysplex, but owned by only one system at a time. To handle a global resource request, the originating system must place the request in the RSA, and pass it around the ring so all other systems are aware of the request. Clearly, since the RSA is a shared data area of limited size, the number of requests that each system can place in the RSA diminishes as the number of systems sharing the RSA increases. Also, as the number of z/OS systems in the ring increases, the length of time for the RSA to be passed around the ring increases. The net effect of this is that the GRS processing capacity goes down as the number of z/OS systems in the ring is increased (that is, fewer requests per second are handled by the ring).

In the case of the GRS star support, this problem is eliminated since all of the requests are handled by mapping ENQ/DEQ requests to XES lock structure requests. With this approach, there is no need to wait for the serialization mechanism, the RSA, to be held by this z/OS before processing the global resource request. Being able to process requests as they are received does improve processing capacity. For instance, there can never be a situation where the system will have to defer excess requests (when the RSA is full).

GRS response time

Very closely related to the capacity problem is the response time that it takes for GRS to handle a request. As can be seen from the capacity discussion, an increase in the number of z/OS systems in the GRS ring results in an increase in the average length of time a given z/OS must wait before it can receive control of the RSA to initiate a new request. Additionally, the length of time that it then takes to circulate the request around the ring before it can be processed is also increased. As a result, the GRS response time on a per request basis increases linearly as the number of z/OS systems in the ring is increased. In the case of the GRS star support, the same processing flow that addressed the capacity constraint also addresses the problem of responsiveness.

CPU consumption

In a GRS ring, each z/OS must maintain a view of the entire global resource queue. This means each z/OS must not only process the requests originating on it, but also must process the global requests originating on all the other z/OS systems in the ring. The effect of this is that each global ENQ/DEQ request generated in a ring consumes processor time on all the z/OS systems in the sysplex, and as the sysplex grows, the amount of processor time GRS ring consumes across the sysplex grows. In the case of the GRS star support, the overhead of processing an ENQ/DEQ request is limited to only the z/OS on which the request originates. Thus, the total processor time consumed across the sysplex is less than that consumed by a GRS ring. However, take into consideration that GRS star topology consumes coupling facility resources, such as: ICFs cycles, central storage and CF links bandwidth.

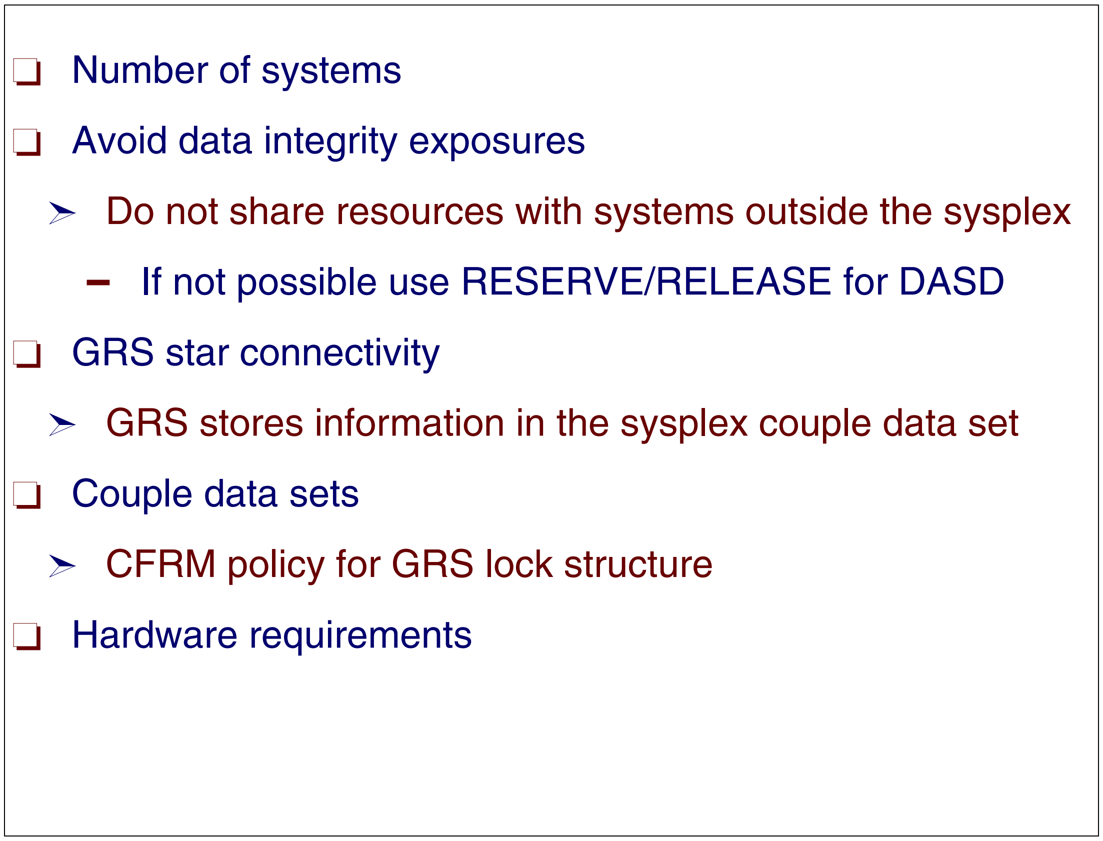
Availability and recovery

In a GRS ring, if one of the z/OS systems fails, all the other z/OS systems are affected during the ring reconstruction time. None of the remaining z/OS systems in the sysplex can process any further ENQ/DEQ requests until the ring is reconstructed. To rebuild the ring, each of the remaining z/OS systems must resynchronize and rebuild their view of the global resource queue. The amount of time this may take depends on the number of z/OS systems in the sysplex and the number of requests in the global resource queue. In the case of the GRS star support, availability is improved by the fact that there is less interdependency between the z/OS systems in the GRS complex. Whenever a z/OS fails, no action need be taken by the other z/OS systems in the sysplex, because there is no resource request data kept on any other z/OS for the requesters on the failing z/OS. All that needs to occur is the analysis of the contention for the resource, which could determine a new owner (or owners) of the resource, should the resource have been owned by the requester on the failing z/OS. This approach allows for a much quicker recovery time from the z/OS systems remaining in the sysplex, since there is no longer a need to clean up the resource queues or determine a new ring topology for the remaining z/OS systems.

Ring versus star topology

A GRS ring configuration can be used when you have no coupling facility available. Remember that GRS in ring mode can use CTCs to pass the RSA, without the need of XCF. The GRS star configuration is suggested for all Parallel Sysplex configurations. The GRS star configuration allows for sysplex growth, and also is of value to installations currently running a sysplex because of the improved responsiveness, the reduced consumption of processor and storage, and the better availability and recovery time.

4.14 GRS star configuration planning



- ❑ Number of systems
- ❑ Avoid data integrity exposures
  - ➢ Do not share resources with systems outside the sysplex
    - – If not possible use RESERVE/RELEASE for DASD
- ❑ GRS star connectivity
  - ➢ GRS stores information in the sysplex couple data set
- ❑ Couple data sets
  - ➢ CFRM policy for GRS lock structure
- ❑ Hardware requirements

Figure 4-14 GRS star configuration planning

Number of z/OS systems

The design of the GRS allows for an unlimited number of z/OS systems. The star topology for serializing global resources can support complexes of up to 32 z/OS systems due to parallel sysplex limitations. These 32 z/OS systems are supported efficiently and effectively.

Avoid data integrity exposures

To avoid a data integrity exposure, ensure that no z/OS outside the sysplex can access the same shared DASD as any z/OS in the sysplex. If that is unavoidable, as often happens, you must serialize the data on the shared DASD with the RESERVE macro and no using for such resource the conversion list. The sample GRS exit ISGGREX0 may help in managing DASD sharing among and outside sysplexes.

As stated before, in a star complex, GRS uses the lock services of the cross-system extended services (XES) component of z/OS to serialize access to global resources, and the GRS star method for processing global requests operates in a sysplex like any other z/OS component that uses the coupling facility. Therefore, a coupling facility with connectivity from all z/OS systems in the GRS complex is required.

GRS star connectivity

The overall design of GRS star makes the connectivity configuration relatively simple. GRS star use sysplex couple data sets, that for sure are shared by all z/OS systems in the sysplex. An alternate sysplex couple data set is recommended to be used to facilitate the migration from a ring to a star complex, and for availability reasons.

XCF stores information related to sysplex, systems, and XCF groups (such as global resource serialization), on the sysplex couple data set. GRS star stores RNL information into the sysplex couple data set.

Couple data sets

The following policies located in couple data sets are used to manage GRS star:

- Coupling facility resource management (CFRM) policy, which is required, defines how XES manages coupling facility structures (ISGLOCK in particular).
- Sysplex failure management (SFM) policy, which is optional, defines how system and signaling connectivity failures should be managed. This is the a recommended way of operating.

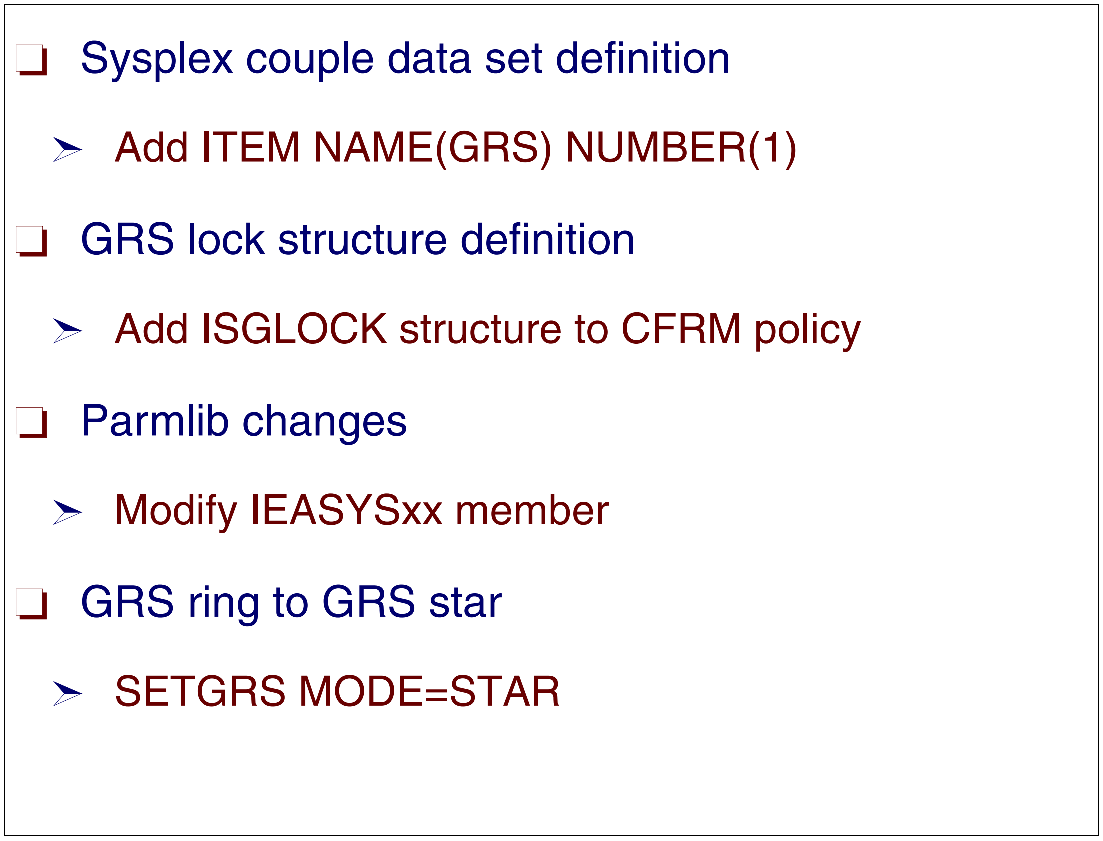
Other requirements

In designing a GRS star configuration, verify that the following requirements are met:

- Hardware requirements:

  A fully interconnected coupling facility (accessible by all z/OS systems)
- All z/OS systems in a GRS star complex must be in the same sysplex
- All z/OS systems in a star complex must be connected to a coupling facility containing the GRS lock structure whose name should be ISGLOCK. For availability reasons in case of coupling facility failure, a second coupling facility should be available to rebuild the structure used by GRS.
- A GRS star complex and a ring complex cannot be interconnected, and therefore they cannot coexist in a single GRS complex.
- GRS star does not support a mixed complex: all z/OS systems in the sysplex should participate in the star complex.
- All z/OS systems must be at z/OS Version 1 Release 2 level or above.

4.15 GRS star implementation



- ❑ Sysplex couple data set definition
  - ➢ Add ITEM NAME(GRS) NUMBER(1)
- ❑ GRS lock structure definition
  - ➢ Add ISGLOCK structure to CFRM policy
- ❑ Parmlib changes
  - ➢ Modify IEASYSxx member
- ❑ GRS ring to GRS star
  - ➢ SETGRS MODE=STAR

Figure 4-15 GRS star implementation

GRS star implementation

Before we start, it is important to mention that the huge majority of customers already migrate from ring to star. And this subject is here just completeness.

Figure 4-15 shows the steps necessary to implement the GRS star complex. It is assumed that a sysplex has already been implemented, that a coupling facility is operational, and that a GRS ring complex that matches the sysplex is working. This also means that the RNLs have been implemented according to the installation needs.

Sysplex couple data set definition

Add the GRS parameter to the sysplex couple data set for the star complex. The sysplex couple data set must be formatted with the IXCL1DSU utility. The GRS parameter is in the DEFINEDS statement. If this is not done, XCF issues the message:

```
IXC291I ITEM NAME NOT DEFINED, GRS
```

For more information on the IXCL1DSU utility, see: *z/OS MVS Setting Up a Sysplex,* SA22-7625

```
DEFINEDS SYSPLEX(WTSCPLX1)
         DSN(SYS1.XCF.CDS01) VOLSER(TOTDS1)
         MAXSYSTEM(16)
         CATALOG
       DATA TYPE(SYSPLEX)
       ITEM NAME(GRS) NUMBER(1)
       ITEM NAME(GROUP)  NUMBER(100)
       ITEM NAME(MEMBER) NUMBER(200)
```

Figure 4-16 Example of IXCL1DSU format utility statements

The ITEM NAME(GRS) NUMBER(1) statement allocates space in the couple data set for GRS usage. The support for star complex does not introduce changes in RNL functions or in the dynamic RNL changes.

Make the new CDS available to sysplex

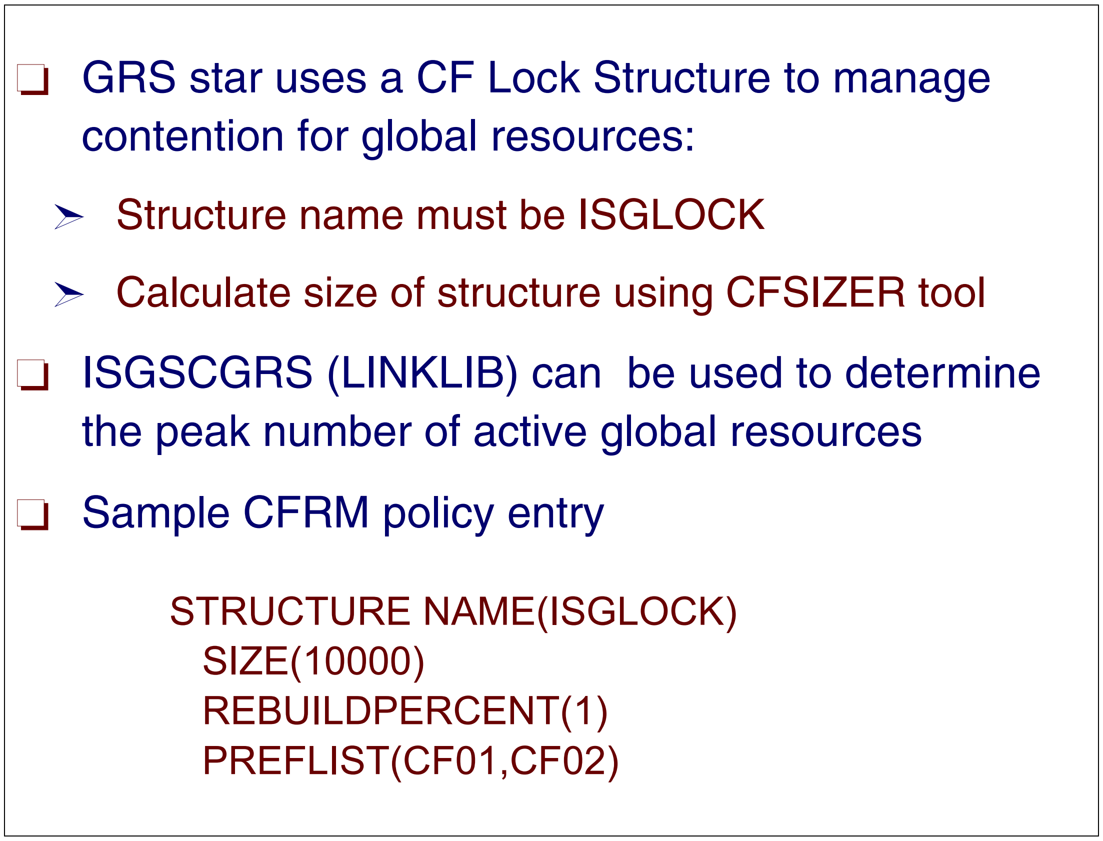
The **SETXCF** command allows to add the new formatted data sets dynamically:

SETXCF COUPLE,PSWITCH	Switches the current alternate sysplex CDS to primary. The primary old sysplex CDS is removed.
SETXCF COUPLE,ACOUPLE=dsname1	Specifies the new CDS to be used as an alternate sysplex CDS.
SETXCF COUPLE,PSWITCH	Switches the current alternate sysplex CDS to primary. This command removes the primary old sysplex CDS.
SETXCF COUPLE,ACOUPLE=dsname2	Add the second new formatted data set as alternate CDS.

The COUPLExx parmlib member must reflect the sysplex couple data set names for future IPLs. An example of COUPLExx member entry is:

```
COUPLE SYSPLEX(WTSCPLX1)
PCOUPLE(SYS1.XCF.CDS01)
ACOUPLE(SYS1.XCF.CDS02)
```

4.16 Define GRS lock structure



- GRS star uses a CF Lock Structure to manage contention for global resources:
  - Structure name must be ISGLOCK
  - Calculate size of structure using CFSIZER tool
- ISGSCGRS (LINKLIB) can be used to determine the peak number of active global resources
- Sample CFRM policy entry

```
STRUCTURE NAME(ISGLOCK)
  SIZE(10000)
  REBUILDPERCENT(1)
  PREFLIST(CF01,CF02)
```

Figure 4-17 Define the GRS lock structure

Define the GRS lock structure

Figure 4-17 shows an example of a GRS lock structure definition. Use the IXCMIAPU utility to define the structure in a new CFRM policy.

For more information on the IXCMIAPU utility, see *z/OS MVS Setting Up a Sysplex,* SA22-7625. GRS uses the XES lock structure to reflect a composite system level of interest for each global resource. The interest is recorded in the user data associated with the lock request.

The name of the lock structure must be ISGLOCK, and the size depends on the following factors:

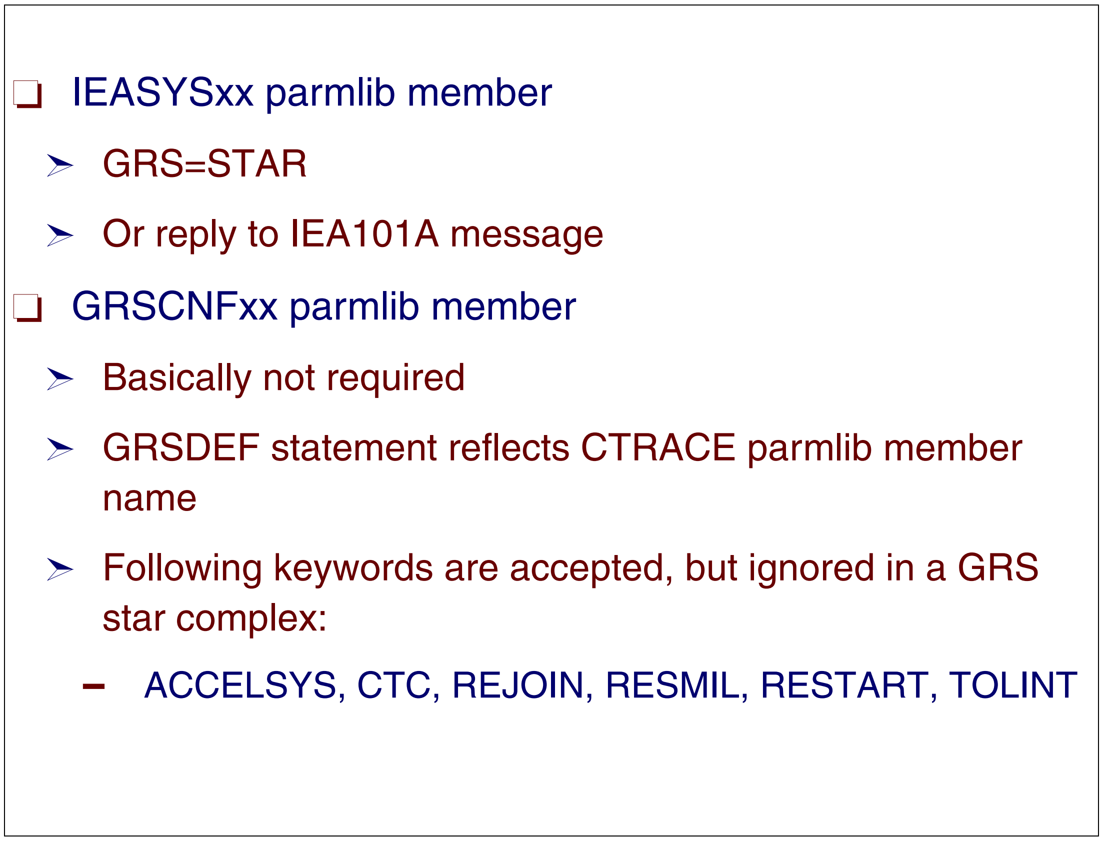
- The number of z/OS in the sysplex
- The type of workload being performed

Use the CFSIZER tool to determine the best size for your ISGLOCK lock structure. This tool is available via the Web page:

`http://www-1.ibm.com/servers/eserver/zseries/cfsizer/`

CFSIZER structure size calculations are always based on the highest available CFCC level. Input for the tool is the *peak global resources* value. The *GRS help* link of CFSIZER contains information on how to determine this value.

4.17 Parmlib changes



- IEASYSxx parmlib member
  - GRS=STAR
  - Or reply to IEA101A message
- GRSCNFxx parmlib member
  - Basically not required
  - GRSDEF statement reflects CTRACE parmlib member name
  - Following keywords are accepted, but ignored in a GRS star complex:
    - ACCELSYS, CTC, REJOIN, RESMIL, RESTART, TOLINT

Figure 4-18 Parmlib changes

IEASYSxx parmlib member

Initializing a star complex requires specifying STAR on the GRS= system parameter in the IEASYSxx parmlib member, or in response to message `IEA101A` during IPL. The START, JOIN, and TRYJOIN options apply to a ring complex only.

The PLEXCFG (also in IEASYSxx) parameter should be MULTISYSTEM; the relation between GRS= and PLEXCFG parameters are:

- Use GRS=NONE and PLEXCFG=XCFLOCAL or MONOPLEX when there is a single-system sysplex and no GRS complex.
- Use GRS=TRYJOIN (or GRS=START or JOIN) and PLEXCFG=MULTISYSTEM when there is a sysplex of two or more z/OS systems and the GRS ring complex uses XCF signalling services.
- Use GRS=STAR and PLEXCFG=MULTISYSTEM when there is a GRS star complex.

GRSCNFxx parmlib member

Basically, the GRSCNFxx member of parmlib is not required when initializing a star complex that uses the default CTRACE parmlib member, CTIGRS00, which is supplied with the z/OS. If you want to initialize a star complex using a CTRACE parmlib member other than the default, you must use the GRSDEF statement in the GRSCNFxx parmlib member. CTRACE describes the options for the GRS component trace, this trace is requested by IBM, when an error is suspected in the code, that is an extremely rare situation. See Figure 4-19 for a GRSCNFxx parmlib member example.

```
GRSDEF MATCHSYS(*)       /* GRSDEF FOR SYSTEMS SYS1, SYS2, SYS4
          CTRACE(CTIGRS01) /* PARMLIB MEMBER CTIGRS01 CONTAINS TRACE
                           /* OPTIONS
GRSDEF MATCHSYS(SYS3)   /* GRSDEF FOR SYSTEM SYS3
         CTRACE(CTIGRS03) /* PARMLIB MEMBER CTIGRS03 CONTAINS TRACE
                      /* OPTION
```

Figure 4-19 GRSCNFxx parmlib member example

GRS ring to star conversions

Apart from the two GRSDEF parameters listed in the example, all the other parameters on the GRSDEF statement (ACCELSYS, RESMIL, TOLINT, CTC, REJOIN, and RESTART) apply only to z/OS systems initializing in a ring complex. Although they can be specified on the GRSDEF statement and are parsed and syntax checked, they are not used when initializing z/OS systems into a GRS star complex. GRS ignores the parameters that are not applicable on the GRSDEF statement when initializing z/OS systems into a star complex, as well as when initializing z/OS systems into a ring complex.

If the ring parameters are left in the GRSCNFxx parmlib member, there is the potential, due to making an error in the specification of the GRS= parameter in IEASYSxx, to create two GRS complexes.

For this reason, even if it is possible for an installation to create a single GRSCNFxx parmlib member that can be used for the initialization of either a star or a ring complex, it is suggested you have different GRSCNFxx members, one for star and one for ring, and use the GRSCNF= keyword in IEASYSxx to select the proper one. This helps in the transition from a ring to a star complex if the installation elects to use the SETGRS MODE=STAR capability to make the transition.

4.18 GRS ring to GRS star

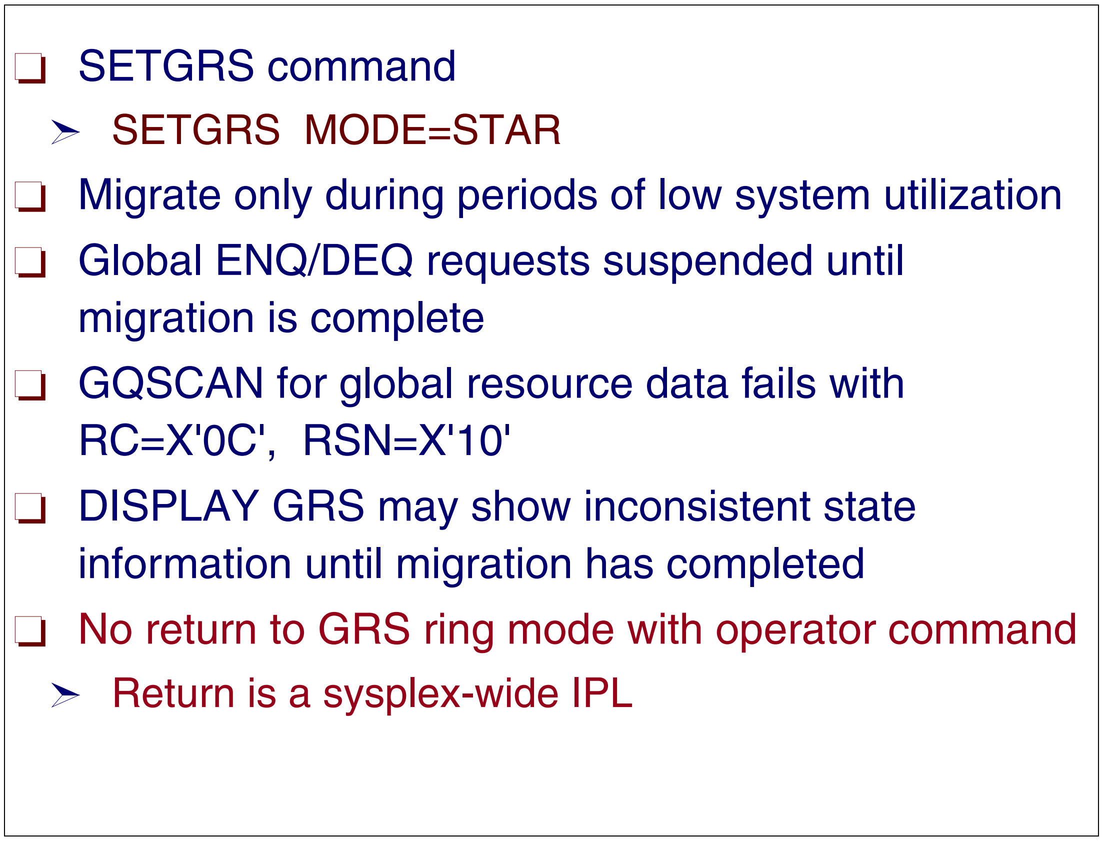

- SETGRS command
  - SETGRS MODE=STAR
- Migrate only during periods of low system utilization
- Global ENQ/DEQ requests suspended until migration is complete
- GQSCAN for global resource data fails with RC=X'0C', RSN=X'10'
- DISPLAY GRS may show inconsistent state information until migration has completed
- No return to GRS ring mode with operator command
  - Return is a sysplex-wide IPL

Figure 4-20 GRS ring to GRS star

GRS ring to GRS star

The **SETGRS** command is used to migrate an active ring to a star complex. There is no **SETGRS** option to migrate from a star to a ring complex. Returning to a ring complex requires an IPL of the entire sysplex.

The following command is used to migrate from a ring complex to a star complex:

```
SETGRS MODE=STAR
```

The **SETGRS** command can be issued from any z/OS in the complex and has sysplex scope.

While processing a **SETGRS MODE=STAR** command, processing is suspended for the GRS ENQ, DEQ, RESERVE, and GQSCAN for global resource data will fail with RC=X'0C', RSN=X'10'. The length of time GRS requesters are suspended may be a few minutes while the ISGLOCK lock structure and GRS sysplex couple data set records are initialized, and changes to the internal GRS control block structures are initialized as well.

Migration considerations

The migration should be invoked at a time when the amount of global resource request activity is likely to be minimal.

The **SETGRS MODE=STAR** request is valid under the following conditions:

- GRS is currently running a ring complex that exactly matches the sysplex.

- All z/OS systems in the ring complex support z/OS Version 1 Release 2 or later.
- All z/OS systems in the ring complex are connected to a coupling facility.
- All z/OS systems can access the ISGLOCK lock structure on the coupling facility.
- The GRS records are defined on the sysplex couple data sets.
- There are no active dynamic RNL changes in progress.

GRS commands

The **`DISPLAY GRS (D GRS)`** command shows the state of each z/OS in the complex. The **`D GRS`** shows z/OS status only as it relates to the GRS ring.

You can also use **`D GRS`** to display the local and global resources requested by the z/OS systems in the ring, contention information, the contents of the RNLs, and jobs that are delaying or suspended by a **`SET GRSRNL`** command.

You can issue **`D GRS`** from any z/OS 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.

4.19 GRS star complex overview

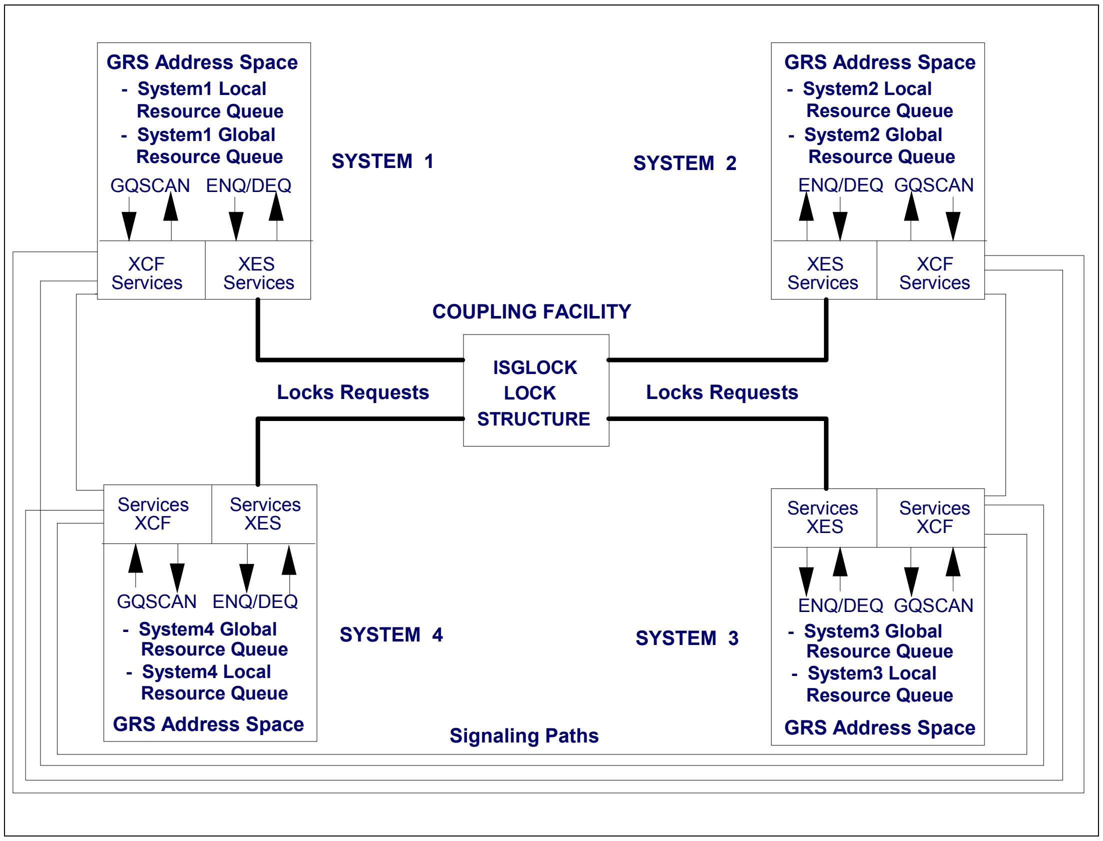


Figure 4-21 GRS star complex overview

GRS star complex overview

In a GRS star complex, when an ENQ, RESERVE, or DEQ request is issued for a global resource, the request is converted to an XES lock request. The XES lock structure coordinates the requests it receives to ensure proper resource serialization across all GRSs in the complex, and notifies the originating GRS about the status of each request. Based on the results of these lock requests, GRS responds to the requester (sometime placing it in wait) with the result of the global request.

As shown in Figure 4-21, each system in the GRS star complex has a *server task* (dispatchable unit) executing in the GRS address space. All ENQ, DEQ, RESERVE, and GQSCAN requests for global resources that are issued from any of the address spaces on a system are always first passed to the GRS address space on the requester's system. In the GRS address space, the server task performs the processing necessary to handle a given request and interfaces with the XES lock structure, if necessary, to process the request. In case of a GQSCAN request for global resource information, the server task packages the request for transmission and sends it to each of the other server tasks in the GRS complex using the XCF services. This is necessary because the status of global requests is maintained at the system level.

Request for a global resource

The request for global resource information is queued to the GQSCAN/ISGQUERY server on that system. Global resource serialization suspends the GQSCAN/ISGQUERY requester.

For GQSCAN/ISGQUERY users that do not need information about other systems and can not tolerate suspension. Global resource serialization processing will then package the request for transmission and send it to each of the other systems in the star complex.

Each system in the complex scans its system global resource queue for global resource requests that match the GQSCAN/ISGQUERY selection criteria, and respond to the originating system server with the requested global resource information, if any, and a return code.

The originating system waits for responses from all of the systems in the sysplex and builds a composite response, which is returned to the caller’s output area through the GQSCAN/ISGQUERY back-end processing.

Control is then returned to the caller with the appropriate return code.

4.20 Global ENQ processing

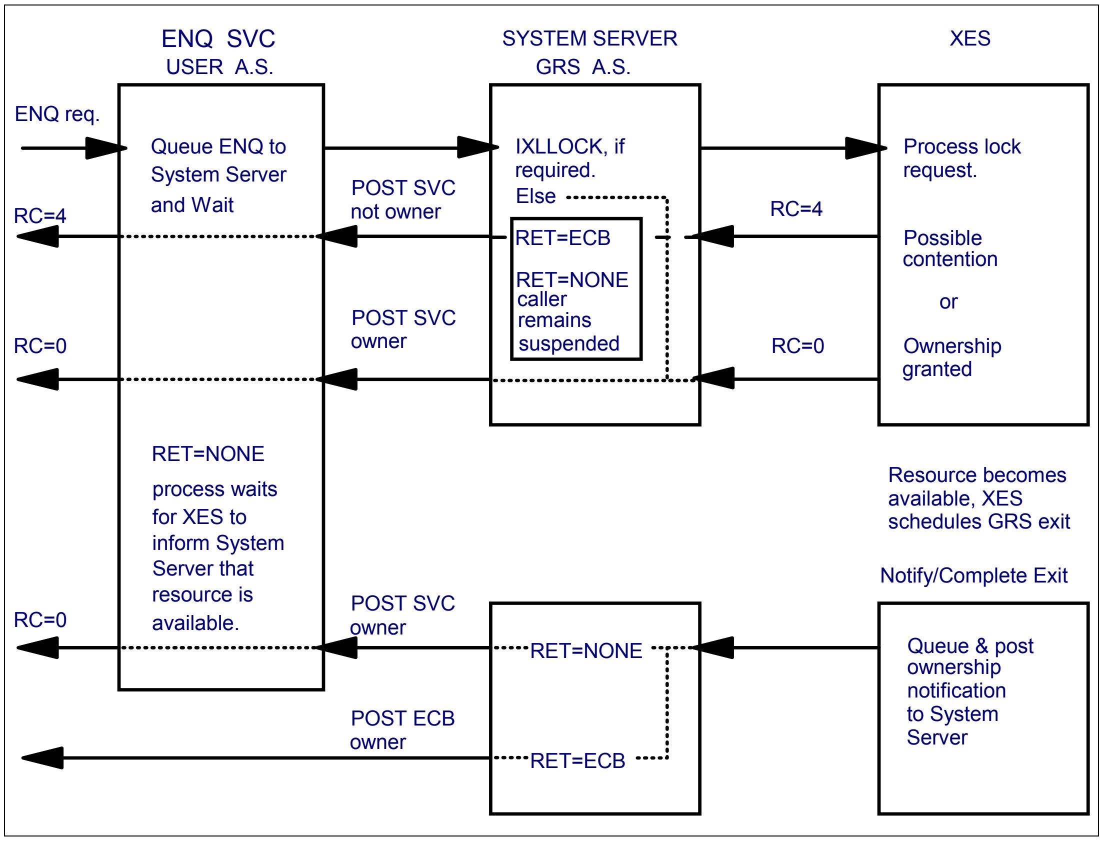


Figure 4-22 Global ENQ processing

Global ENQ processing

ENQ processing for the RET=NONE and RET=ECB keywords is shown at a high level in Figure 4-22. These operands mean that control of all the resources is unconditionally requested, if not the requesting task should be placed in a a wait state through an ECB. In a GRS star complex, no GRS maintains a complete view of the outstanding global resource requests, in contrast with the GRS ring philosophy of maintaining a complete view of the global resource requests on all systems in the complex. Instead, each system maintains a queue of each of the local requests, called the *system global resource queue*. Arbitrating requests for global resources from different systems in the complex is managed by putting a subset of the information from the system global resource queue into the user data associated with the lock request made to the XES lock structure for a particular resource.

Using lock structures

In a GRS star complex, requests for ownership of global resources are handled through a lock structure on a coupling facility that is fully interconnected with all the systems in the sysplex. GRS interfaces with the XES lock structure to reflect a composite system-level view of interest in each of the global resources for which there is at least one requester. This interest is recorded in the user data associated with the lock request.

Each time there is a change in the composite state of a resource, GRS updates the user data reflecting the new state of interest in the XES lock structure. In general, this composite state is altered each time a change is made to the set of owners of the resource or the set of waiters for the resource.

ENQ requests

Each time an ENQ request is received, the server task analyzes the state of the resource request queue for the resource. If the new request alters the composite state of the queue, an IXLLOCK request is made to reflect the changed state of the resource for the requesting system. If the resource is immediately available, the IXLLOCK request indicates that the system can grant ownership of the resource. If the XES lock structure cannot immediately grant ownership of the resource, the IXLLOCK request completes with the return code X' 04', and the server task holds the request pending and the requester in a wait state until the completion exit is driven for the request.

Only one IXLLOCK can ever be in progress from a particular system and for a particular global resource at one time. Requests for a resource that are received while a request is in progress are held pending until all preceding requests for the resource have been through IXLLOCK processing. The requests are processed in first-in first-out (FIFO) order. If contention exists for the resource, the requester is not granted ownership until some set of requesters dequeue from the resource. When the requester is to be assigned ownership of the resource, the contention exit, driven as the result of a DEQ request, drives the notify exit of the server task that is managing the requester.

Limits for ENQs

GRS allows flexibility and control over the system-wide maximums. The default values are:

- 16,384 for unauthorized requests
- 250,000 for authorized requests

This allows you to set the system-wide maximum values. Furthermore, it is now possible for an authorized caller to set its own address space-specific limits. These new values substantially increased STAR mode capacity. The majority of the persistent ENQs are global ENQs because they are data set related. There is only minimal relief for GRS=NONE, GRS=RING mode systems, and systems running some ISV serialization products.

ISGADMIN service

The ISGADMIN service allows you to programmatically change the maximum number of concurrent ENQ, ISGENQ, RESERVE, GQSCAN, and ISGQUERY requests in an address space. This is useful for subsystems such as CICS and DB2, which have large numbers of concurrently outstanding ENQs, query requests, or both. Using ISGADMIN, you can set the maximum limits of unauthorized and authorized concurrent requests. It is impossible to set the maximums lower than the system-wide default values.

New keywords in the GRSCNFxx parmlib member

To allow you to control the system-wide maximums, two new keywords are added to the GRSCNFxx parmlib member:

ENQMAXU(value)	Identifies the system-wide maximum of concurrent ENQ requests for unauthorized requesters. The ENQMAXU range is 16,384 to 99,999,999. The default is 16,384.
ENQMAXA(value)	Identifies the system-wide maximum of concurrent ENQ requests for authorized requesters. The ENQMAXA range is 250,000 to 99,999,999. The default is 250,000.

Use the **SETGRS ENQMAXU** command to set the system-wide maximum number of concurrent unauthorized requests, and the **SETGRS ENQMAXA** command to set the system-wide maximum number of concurrent authorized requests.

```
SETGRS ENQMAXU=nnnnnnnn[,NOPROMPT|NP]
SETGRS ENQMAXA=nnnnnnnn[,NOPROMPT|NP]
```

4.21 Global DEQ processing

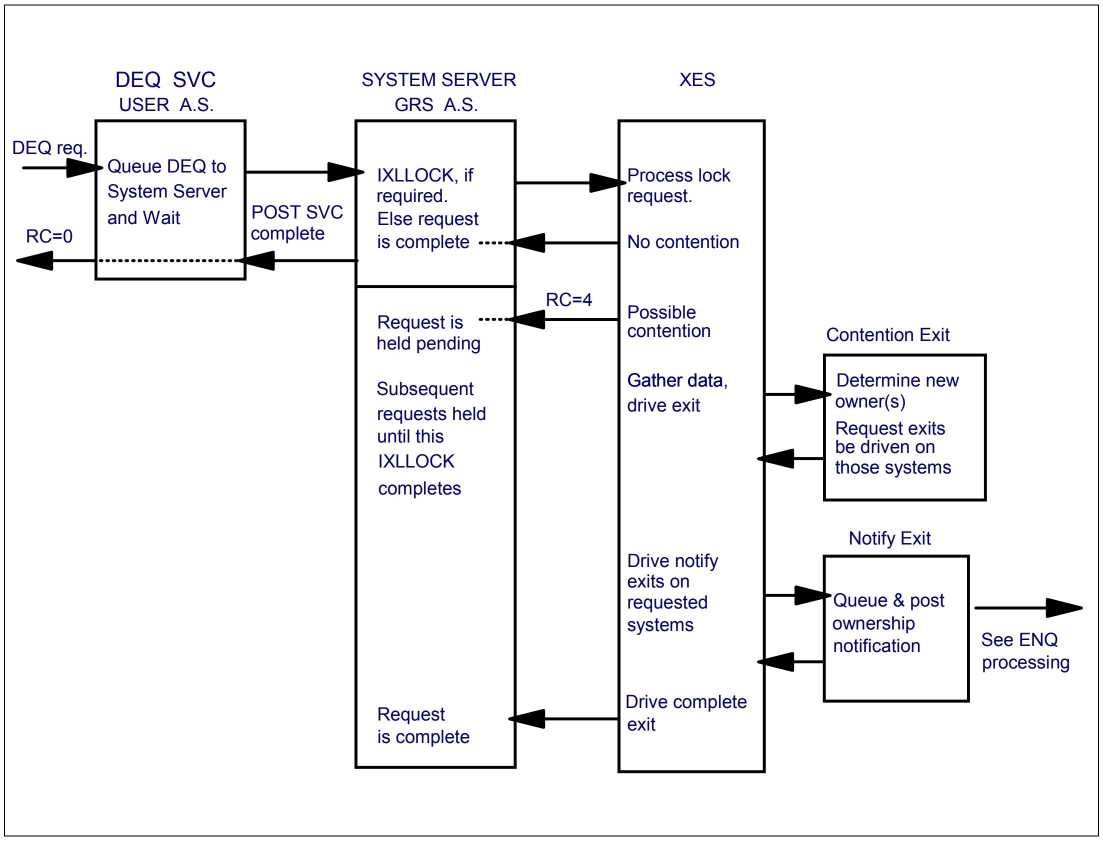


Figure 4-23 Global DEQ processing

Global DEQ processing

In a GRS star complex, releasing a resource has the effect of altering the system's interest in the resource. GRS alters the user data associated with the resource, and issues an IXLLOCK request to reflect the new interest. Figure 4-23 illustrates the global DEQ process.

After resuming the requesting task, the server task performs local processing and issues an IXLLOCK request if the DEQ results in an alteration of the system's composite state of the resource. If there is no global contention on the resource, XES returns from the IXLLOCK service with a return code of X'0', indicating that no asynchronous ownership processing is occurring.

When IXLLOCK completes with return code X'04', XES indicates that there may be contention for the resource and that asynchronous processing occurs for this resource. The server task places this resource into a pending state, keeping all subsequent requests queued in FIFO order until the asynchronous processing completes. See "Contention notification system (CNS)" on page 222

GRS contention exit

XES gathers the user data from each of the z/OS systems interested in the resource and presents it to the contention exit that GRS provides. The contention exit determines, based on each of the systems' composite states, which systems have owners of the resource. The contention exit causes the notify exit on each of the systems to be driven. The notify exit informs the server task of the change in ownership of the resource. The server task posts the new owner (or owners) of the resource.

GRS fast DEQ requests

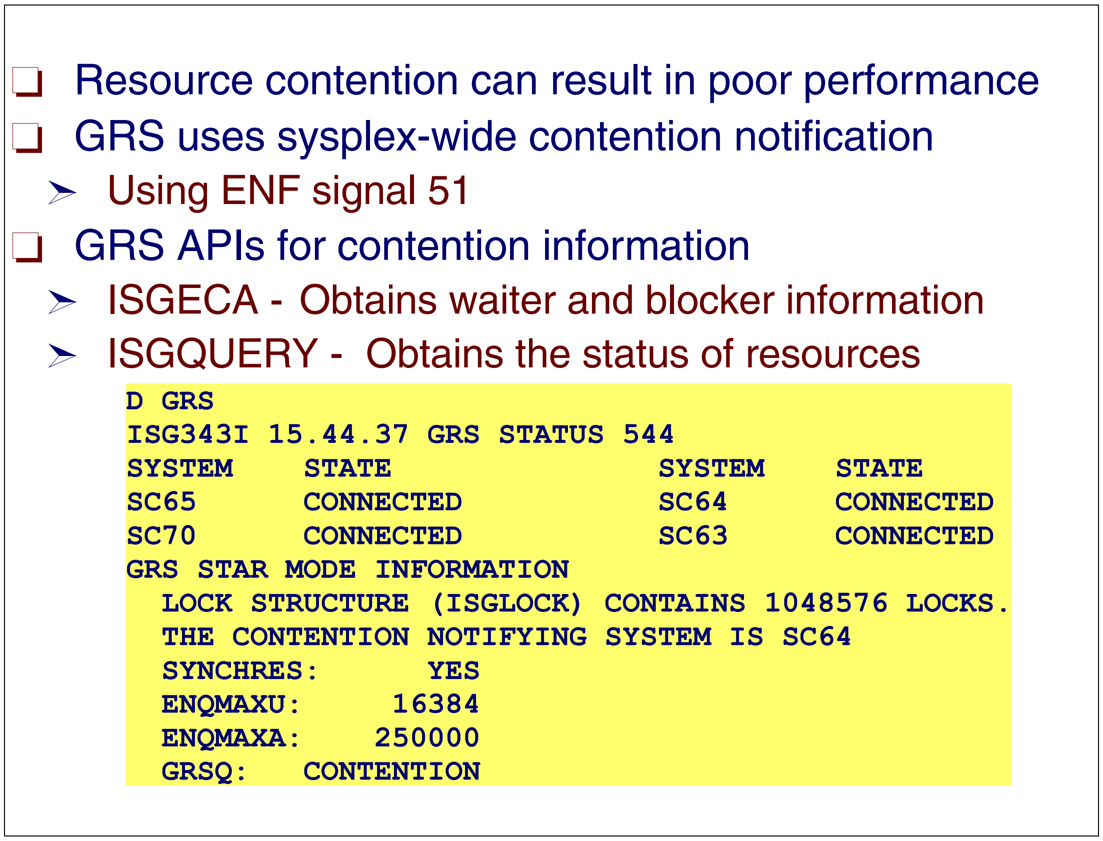
In a GRS star complex, a DEQ request is handled as a fast DEQ under the following conditions:

- The DEQ is for a single resource.
- The resource request (ENQ for the resource) has already been reflected in the GRS lock structure.
- The request for the resource is not the target of a MASID/MTCB request.

This is equivalent to the implementation of fast DEQ in a GRS ring complex.

If the resource request has not yet been reflected in the GRS lock structure, the DEQ requester waits until the IXLLOCK for that request has been completed. Following this, the request is resumed in parallel with the completion of the DEQ request.

4.22 Contention notification



- Resource contention can result in poor performance
- GRS uses sysplex-wide contention notification
  - Using ENF signal 51
- GRS APIs for contention information
  - ISGECA - Obtains waiter and blocker information
  - ISGQUERY - Obtains the status of resources

```
D GRS
ISG343I 15.44.37 GRS STATUS 544
SYSTEM    STATE               SYSTEM    STATE
SC65      CONNECTED           SC64      CONNECTED
SC70      CONNECTED           SC63      CONNECTED
GRS STAR MODE INFORMATION
  LOCK STRUCTURE (ISGLOCK) CONTAINS 1048576 LOCKS.
  THE CONTENTION NOTIFYING SYSTEM IS SC64
  SYNCHRES:      YES
  ENQMAXU:     16384
  ENQMAXA:    250000
  GRSQ:   CONTENTION
```

Figure 4-24 Contention notification

Contention notification system (CNS)

In ring mode, each system in the global resource serialization complex knows about the complex-wide SYSTEMS scope ENQs that allow each system to issue the appropriate event notification facility (ENF) signal 51 contention notification event. However, in star mode, each system only knows about the ENQs that are issued by the system.

Resource contention

Resource contention can result in poor system performance. When resource contention lasts over a long period of time, it can result in program starvation or deadlock conditions. When running in ring mode, each system in the GRS complex is aware of the complex-wide ENQs, which allows each system to issue the appropriate ENF 51 contention notification event. However, when running in star mode, each system only knows the ENQs that are issued by itself.

ENF 51 signal

To ensure that the ENF 51 contention notification event is issued on all systems in the GRS complex in a proper sequential order, one system in the complex is appointed as the sysplex-wide contention notifying system (CNS). All ENQ contention events are sent to the CNS, which then issues a sysplex-wide ENF 51. GRS provides two APIs to query for contention information:

ISGECA	Obtains waiter and blocker information for GRS-managed resources.
ISGQUERY	Obtains the status of resources and requester of those resources.

GRS issues an ENF 51 to notify monitoring programs to track resource contention. This is useful in determining contention bottlenecks, preventing bottlenecks, and potentially automating correction of these conditions.

During an IPL, or if the CNS can no longer perform its duties, any system in the complex can act as the CNS. You can determine which system is the current CNS with the **`DISPLAY GRS`** command. In the example shown in Figure 4-25, you can see that the CNS is SC64. Figure 4-26 shows an example of changing the CNS from system SC64 to system SC70.

```
D GRS
ISG343I 15.44.37 GRS STATUS 544
SYSTEM    STATE               SYSTEM    STATE
SC65      CONNECTED           SC64      CONNECTED
SC70      CONNECTED           SC63      CONNECTED
GRS STAR MODE INFORMATION
  LOCK STRUCTURE (ISGLOCK) CONTAINS 1048576 LOCKS.
  THE CONTENTION NOTIFYING SYSTEM IS SC64
  SYNCHRES:      YES
  ENQMAXU:     16384
  ENQMAXA:    250000
  GRSQ:   CONTENTION
```

Figure 4-25 Output of the DISPLAY GRS command

```
SETGRS CNS=SC70
*092 ISG366D CONFIRM REQUEST TO MIGRATE THE CNS TO SC70. REPLY CNS=SC70
  TO CONFIRM OR C TO CANCEL.
092CNS=SC70
 IEE600I REPLY TO 092 IS;CNS=SC70
 IEE712I SETGRS   PROCESSING COMPLETE
 ISG364I CONTENTION NOTIFYING SYSTEM MOVED FROM SYSTEM SC64 TO SYSTEM SC70.
         OPERATOR COMMAND INITIATED.
D GRS
 ISG343I 15.54.47 GRS STATUS 565
 SYSTEM    STATE               SYSTEM    STATE
 SC65      CONNECTED           SC64      CONNECTED
 SC70      CONNECTED           SC63      CONNECTED
 GRS STAR MODE INFORMATION
   LOCK STRUCTURE (ISGLOCK) CONTAINS 1048576 LOCKS.
   THE CONTENTION NOTIFYING SYSTEM IS SC70
   SYNCHRES:      YES
   ENQMAXU:     16384
   ENQMAXA:    250000
   GRSQ:   CONTENTION
```

Figure 4-26 Setting the CNS using SETGRS command

4.23 GQSCAN request for global resource data

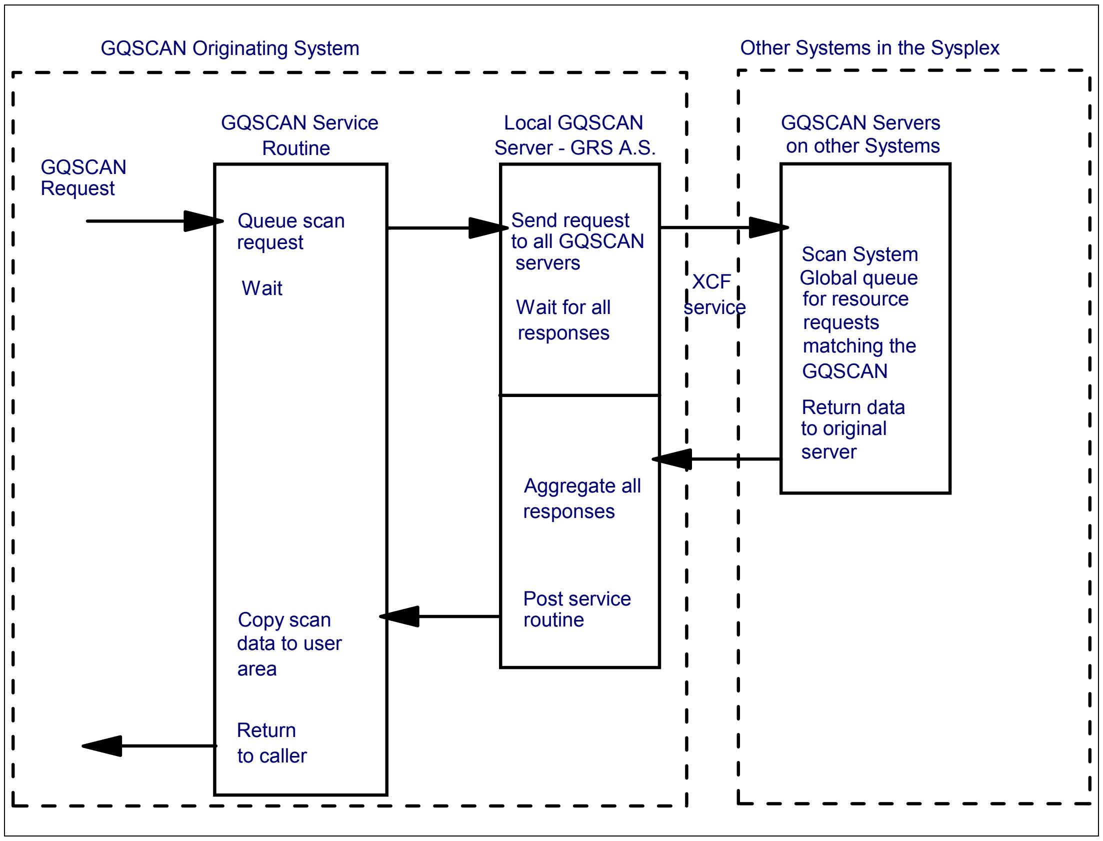

GQSCAN Originating System
Other Systems in the Sysplex
GQSCAN Request
GQSCAN Service Routine
Queue scan request
Wait
Copy scan data to user area
Return to caller
Local GQSCAN Server - GRS A.S.
Send request to all GQSCAN servers
Wait for all responses
Aggregate all responses
Post service routine
XCF service
GQSCAN Servers on other Systems
Scan System Global queue for resource requests matching the GQSCAN
Return data to original server

Figure 4-27 GQSCAN request for global resource data

GQSCAN request for global resource data

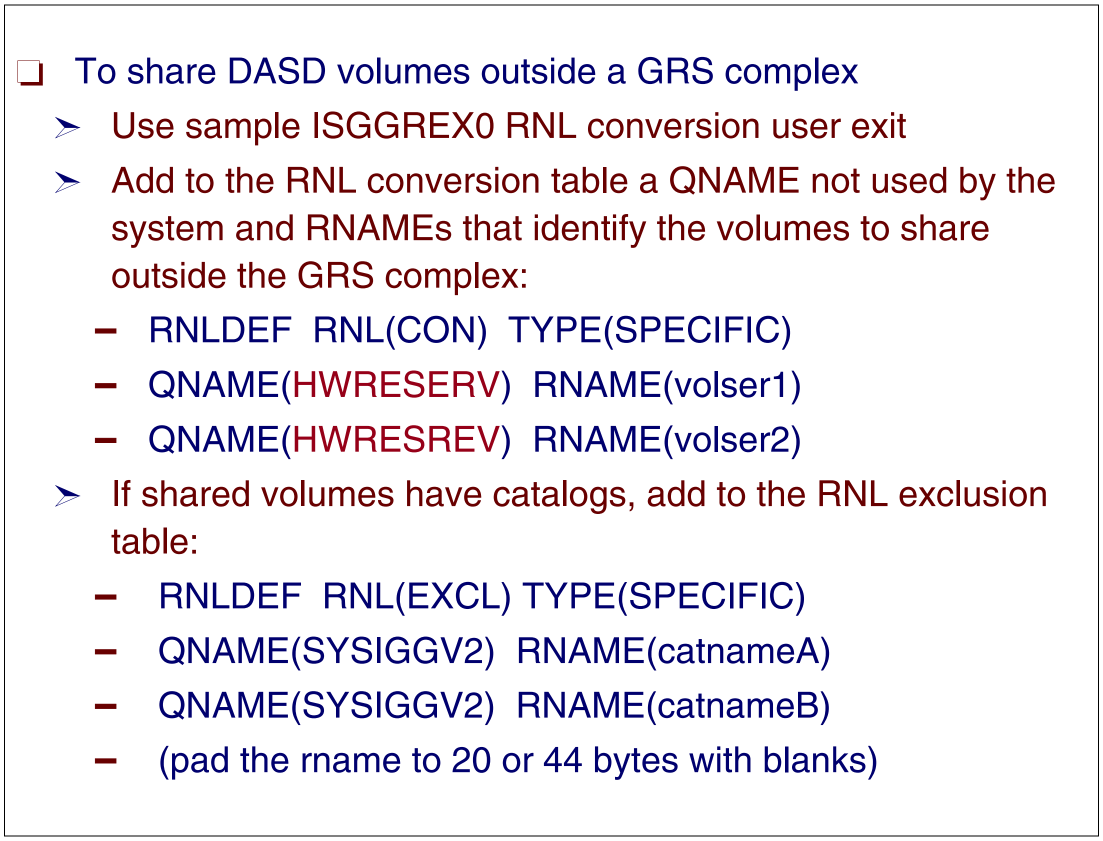
The basic flow of a GQSCAN request for global resource in a star complex is shown in Figure 4-27. In the case of a GQSCAN request for global resource information, the request is passed from the issuing address space to the GRS server task on that z/OS system, and the GQSCAN request is *waited*. The server task then packages the request for transmission and sends it to each of the server tasks in the GRS star complex.

Each server task in the complex scans its system global resource queue that matches the GQSCAN selection criteria, responding to the originating system with the requested global resource information (if any) and a return code. The originating server task waits for responses from all of the server tasks in the complex and builds a composite response to be returned to the caller.

Waiting the issuer of GQSCAN when global resource information is requested is a significant change from the way GQSCAN works in a GRS ring complex. In a GRS ring complex, GQSCAN processing is performed synchronously with respect to the caller's task.

A GQSCAN macro option is provided to allow the issuer to indicate whether or not cross-system processing for global resource information should be performed. The *no cross-system* option (XSYS=NO) is provided primarily for the GQSCAN that cannot be put into a wait, and does not require that data about requesters on other systems in the complex. XSYS=YES is the default option.

4.24 ISGGREX0 RNL conversion exit



❑ To share DASD volumes outside a GRS complex

- ➢ Use sample ISGGREX0 RNL conversion user exit
- ➢ Add to the RNL conversion table a QNAME not used by the system and RNAMEs that identify the volumes to share outside the GRS complex:
  - RNLDEF RNL(CON) TYPE(SPECIFIC)
  - QNAME(HWRESERV) RNAME(volser1)
  - QNAME(HWRESREV) RNAME(volser2)
- ➢ If shared volumes have catalogs, add to the RNL exclusion table:
  - RNLDEF RNL(EXCL) TYPE(SPECIFIC)
  - QNAME(SYSIGGV2) RNAME(catnameA)
  - QNAME(SYSIGGV2) RNAME(catnameB)
  - (pad the rname to 20 or 44 bytes with blanks)

Figure 4-28 ISGGREX0 RNL conversion exit

ISGGREX0 RNL conversion exit

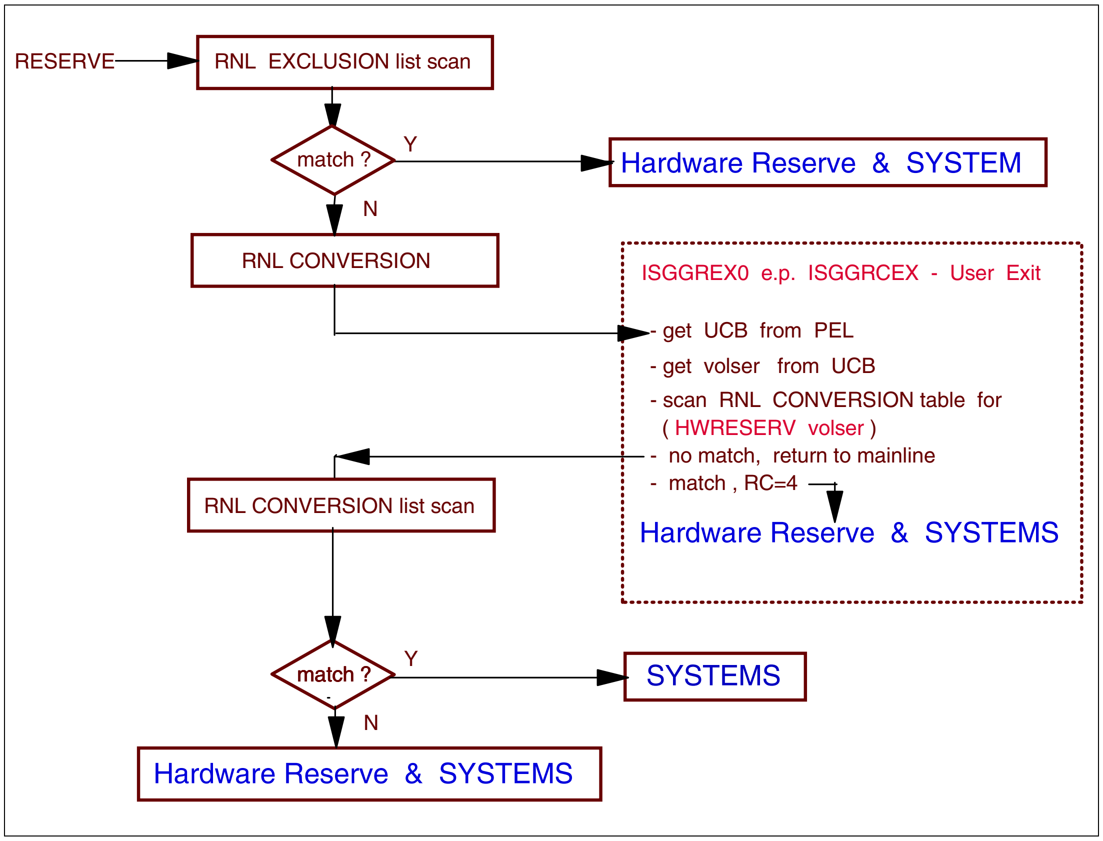
When GRS receives a ENQ/ISNENQ/RESERVE/DEQ request for a resource with a scope of SYSTEM or SYSTEMS, the installation-replaceable scan RNL exit (ISGGREX0) is invoked to search the appropriate RNL. However, this exit is not invoked, if the request specifies RNL=NO or you have specified GRSRNL=EXCLUDE on the IEASYSxx parmlib member. A return code from ISGGREX0 indicates whether or not the input resource name exists in the RNL.

Input to the ISGGREX0 consists of the QNAME, the RNAME, and the RNAME length of the resource. The IBM default ISGGREX0 exit search routine finds a matching RNL entry when:

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

  The length of the specific RNAME is important. A specific entry does not match a resource name that is padded with blank characters to the right.
- ▸ A generic QNAME entry in the RNL matches the QNAME of the search argument.
- ▸ A generic QNAME entry in the RNL matches the corresponding portion of the resource name in the search argument.

4.25 ISGGREX0 conversion exit flow



RESERVE
RNL EXCLUSION list scan
match ?
Y
N
Hardware Reserve & SYSTEM
RNL CONVERSION
ISGGREX0 e.p. ISGGRCEX - User Exit
- get UCB from PEL
- get volser from UCB
- scan RNL CONVERSION table for (HWRESERV volser)
- no match, return to mainline
- match , RC=4
Hardware Reserve & SYSTEMS
RNL CONVERSION list scan
match ?
Y
N
SYSTEMS
Hardware Reserve & SYSTEMS

Figure 4-29 ISGGREX0 conversion exit flow

ISGGREX0 conversion exit flow

GRS has a user exit, ISGGREX0, that receives control whenever an ENQ/DEQ/RESERVE request is issued for a resource. ISGGREX0 scans the input resource name list (RNL) for the resource name specified in the input parameter element list (PEL). A return code from the exit routine indicates whether or not the input resource name appears in the RNL.

You can use ISGGREX0 to determine whether the input resource name exists in the RNL. Replacing ISGGREX0 changes the technique that GRS normally uses to scan an RNL. Changing the search technique can have an adverse effect on system performance, especially when the RNLs contain many specific entries.

The routine has three external entry points:

- ISGGSIEX scans the SYSTEM inclusion RNL.
- ISGGSEEX scans the SYSTEM exclusion RNL.
- ISGGRCEX scans the RESERVE conversion RNL.

Invoking the exit routine

Depending on the RNL the request requires, the exit routine is invoked at the appropriate entry point for the SYSTEM inclusion RNL, the SYSTEM exclusion RNL, or the RESERVE conversion RNL.

You can modify the IBM-supplied exit routine to perform the processing necessary for your system. Use the source version of the exit routine provided as member ISGGREXS in SYS1.SAMPLIB.

For example, ISGGRCEX can be used to avoid converting reserves against a volume on a system outside the global resource serialization complex.

The ISGGREX0 exit routine in each system belonging to the same GRS complex must yield the same scan results; otherwise, resource integrity cannot be guaranteed. For the same reason, the RNLs themselves must be the same. The exit routine resides in the nucleus and to change or activate it, a sysplex-wide IPL is required. See *z/OS MVS: Installation Exits*, SA22-7593, for details.

4.26 Shared DASD between sysplexes

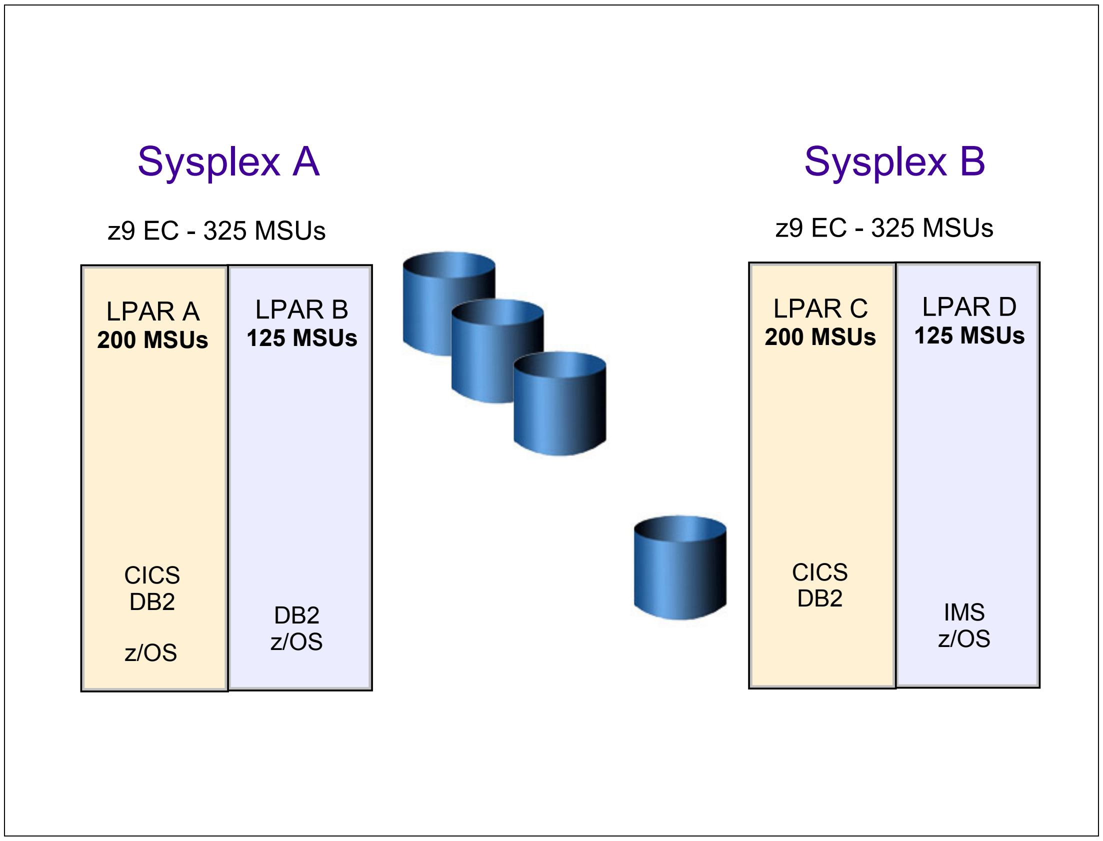


Figure 4-30 Shared DASD between sysplexes

Shared DASD between sysplexes

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.

It is possible to support shared DASD with z/OS systems across GRSplexes, as shown in Figure 4-30, by coding a special exit and placing it in the GRS ISGNQXIT exit. Since it is possible for a single installation to have two or more global resource serialization complexes, each operating independently. However, the independent complexes cannot share resources. Also, be certain there are no common links 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. If that is unavoidable, however, you must serialize the data on the shared DASD with the RESERVE macro. You need to decide how many of these systems to combine in one complex.

ISGNQXIT exit

This exit code allows all RESERVE requests for specified volumes in the RNL list to result in an HW RESERVE whenever the resource name is specified in the RNL list. By using the new ISGNQXIT exit, and adding an RNL definition to the conversion table with a QNAME not used by the system and RNAMEs that identify the volumes to share outside of GRS, the RESERVE requests for the volumes included in the RNL definitions always result in an HW RESERVE.

```
RNLDEF RNL(CON) TYPE(SPECIFIC|GENERIC)
QNAME(HWRESERV)
RNAME(VOLSER|volser-prefix)
```

The same RESERVE resource requests, which address other volumes, should have the possibility to be filtered through the conversion RNLs to have the HW RESERVE eliminated.

Prior to RNL processing, your system can implement the ISGNQXIT or the ISGNQXITFAST installation exits. These exit routines will affect the way ISGENQ, ENQ, DEQ, and RESERVE requests are handled. The following attributes can be altered (note that for certain types of requests, the exits can change the return/reason code):

- QName
- RName
- Scope
- Device UCB address
- Convert Reserve
- Bypass RNLs.

4.27 ISGNQXITFAST and ISGNQXIT exits

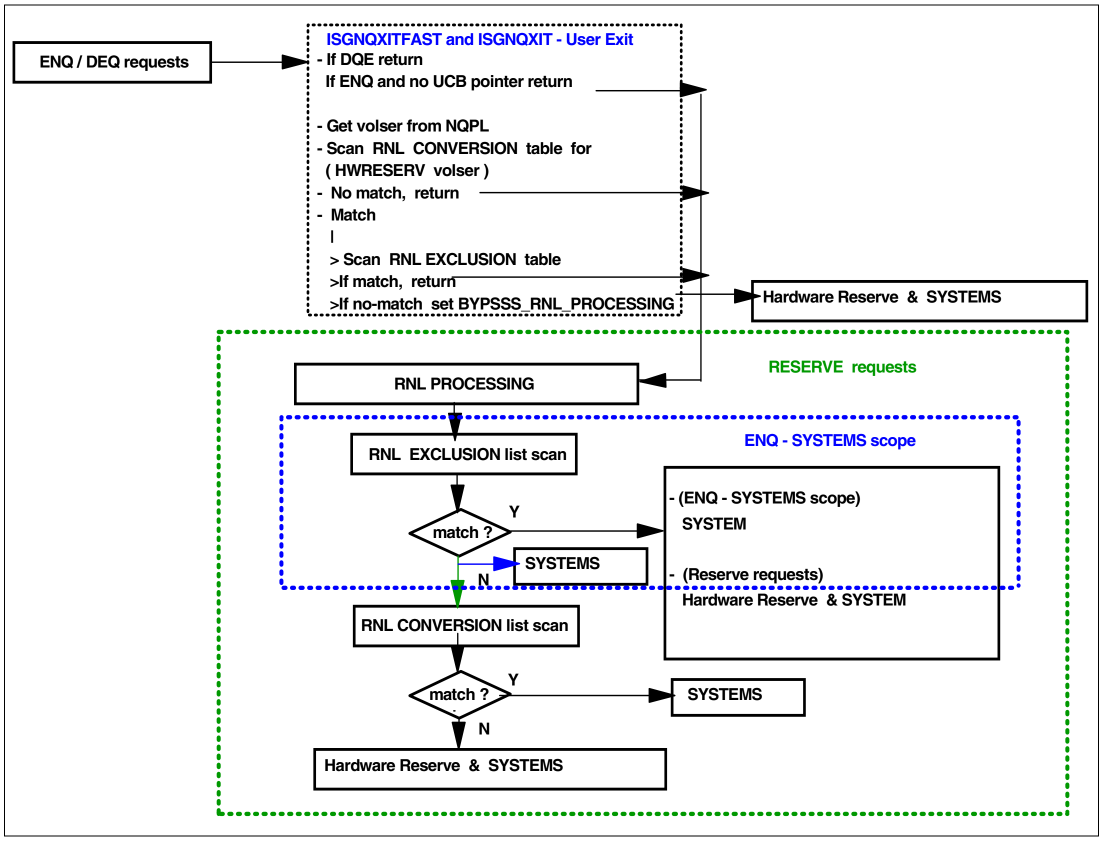


Figure 4-31 ISGNQXITFAST and ISGNQXIT exits

ISGNQXITFAST and ISGNQXIT exits

For each ENQ/DEQ/RESERVE request with SCOPE=SYSTEM or SCOPE=SYSTEMS, the system invokes the ENQ/DEQ installation exit points.

For this purpose, GRS with wildcard support provides exit point ISGNQXIT. GRS also provides the exit point ISGNQXITFAST, which is intended to offer a higher performance alternative to ISGNQXIT.

Both installation exits can modify attributes of the ENQ/DEQ/RESERVE request prior to Resource Names List (RNL) processing; the only difference is the environment with which the exits receive control. For additional information, refer to *z/OS MVS: Installation Exits*, SA22-7593.

For the purpose of cross-sysplex DASD sharing, only one of the provided sample installation exits ISGNQXITFAST and ISGNQXIT should be active. Both installation exits provide the same functionality and have the same external parameters.

Note: System programmers only use RNLs and either the ISGNQXITFAST or ISGNQXIT installation exit. All of the other exits are intended for the independent software vendors (ISVs). As of z/OS V1R9 and later, the installation exits that are used by the ISVs and the ISGNQXITFAST exit (not the ISGNQXIT exit) can get control in a cross memory environment.

Whenever global resource serialization encounters a request for a resource with a scope of SYSTEM or SYSTEMS, global resource serialization searches the appropriate RNL to determine the scope of the requested resource. Global resource serialization will not scan the RNL if one of the following is true:

- The request specifies RNL=NO.
- During the system IPL, GRSRNL=EXCLUDE was specified.
- During the system IPL, GRS=NONE in IEASYSxx was specified.
- The ISGNQXIT or ISGNQXITFAST installation exit routines are installed and active on the system.

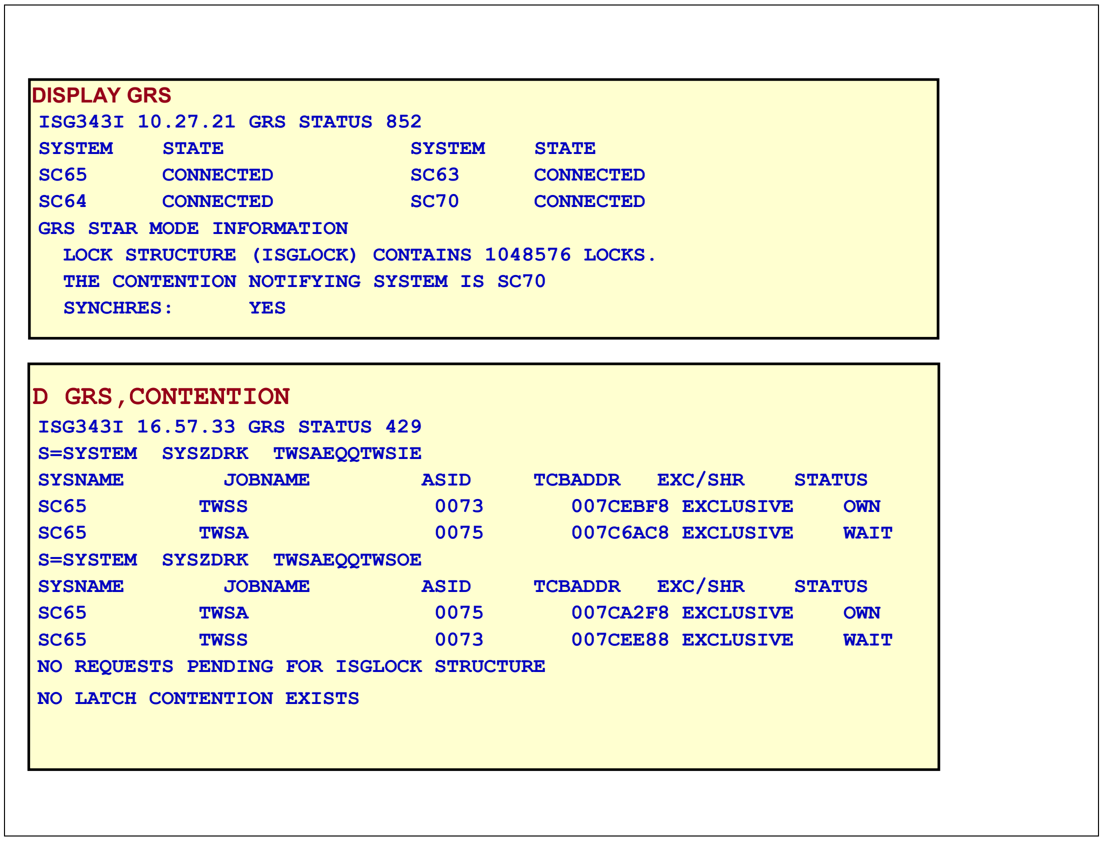
GRS enhancements for dynamic exits

All major internal GRS control blocks are moved above the bar. This caused some problems for others' serialization products to extend their monitoring software through external available interfaces. The dynamic GRS exits was enhanced so that the alternate serialization products can fully support the (ENQ, DEQ, RESERVE for both PCs & SVCs) interface in their resource monitoring.

These are the GRS dynamic relevant exits that alternate serialization products can use (ISGCNFXITSYSTEM and ISGCNFXITSYSPLEX were updated and ISGNQXITQUEUED2 is completely new), as follows:

- ISGNXITPREBATCH performance-oriented exit used to determine if the ISGNQXITBATCHCND should be called for a particular ENQ request.
- ISGNQXITBATCHCND details of a particular ENQ request prior to GRS queuing it up.
- ISGCNFXITSYSTEM filtering exit for contention seen for scope=SYSTEM ENQs.
- ISGENDOFLQCB notification of the last requester of a resource DEQing.
- ISGCNFXITSYSPLEX filtering exit for contention seen for scope=SYSTEMS ENQs.
- ISGNQXITQUEUED1 follow-up from the batch exit where local resources have been queued.
- ISGNQXITQUEUED2 follow-up from the batch and queued2 exits where global resources have been queued.

4.28 GRS star operating (1)



```
DISPLAY GRS
ISG343I 10.27.21 GRS STATUS 852
SYSTEM     STATE                SYSTEM     STATE
SC65       CONNECTED            SC63       CONNECTED
SC64       CONNECTED            SC70       CONNECTED
GRS STAR MODE INFORMATION
  LOCK STRUCTURE (ISGLOCK) CONTAINS 1048576 LOCKS.
  THE CONTENTION NOTIFYING SYSTEM IS SC70
  SYNCHRES:      YES
```

```
D GRS,CONTENTION
ISG343I 16.57.33 GRS STATUS 429
S=SYSTEM  SYSZDRK  TWSAEQQTWSIE
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65          TWSS               0073       007CEBF8 EXCLUSIVE    OWN
SC65          TWSA               0075       007C6AC8 EXCLUSIVE    WAIT
S=SYSTEM  SYSZDRK  TWSAEQQTWSOE
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65          TWSA               0075       007CA2F8 EXCLUSIVE    OWN
SC65          TWSS               0073       007CEE88 EXCLUSIVE    WAIT
NO REQUESTS PENDING FOR ISGLOCK STRUCTURE
NO LATCH CONTENTION EXISTS
```

Figure 4-32 GRS star operating (1)

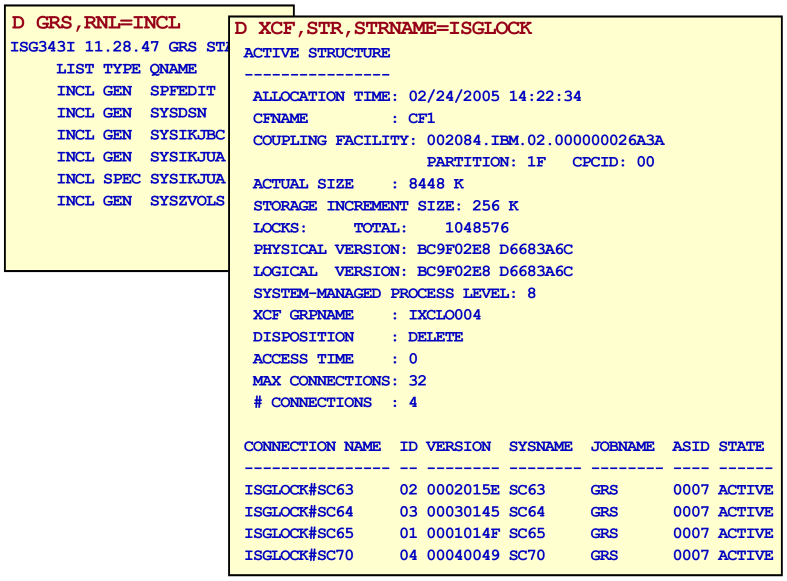
GRS star operating (1)

The **`DISPLAY GRS`** command shows the state of each system in the complex. The system status is not an indication of how well a system is currently running. There are four systems in the complex, all in GRS state connected. The ISGLOCK structure contains 1048576 locks and the GRS contention notifying system is SC70.

`DISPLAY GRS,CONTENTION` is useful for analyzing whether a resource contention exists in the complex. In Figure 4-32 sample outputs ASID 0073 and ASID 0075 both hold *exclusive a* resource with the same ASIDs waiting for *exclusive* access. The problem looks like a contention deadlock.

To get more details, **`DISPLAY GRS,ANALYZE,BLOCKER`** or **`ANALYZE,WAITER`** or **`ANALYZE,DEPENDENCY`** can be used to find out why a resource is blocked and which job or ASID is responsible for holding the resource.

4.29 GRS star operating (2)



```
D GRS,RNL=INCL
ISG343I 11.28.47 GRS ST
     LIST TYPE QNAME
     INCL GEN  SPFEDIT
     INCL GEN  SYSDSN
     INCL GEN  SYSIKJBC
     INCL GEN  SYSIKJUA
     INCL SPEC SYSIKJUA
     INCL GEN  SYSZVOLS
```

```
D XCF,STR,STRNAME=ISGLOCK
 ACTIVE STRUCTURE
 ----------------
  ALLOCATION TIME: 02/24/2005 14:22:34
  CFNAME         : CF1
  COUPLING FACILITY: 002084.IBM.02.000000026A3A
                     PARTITION: 1F   CPCID: 00
  ACTUAL SIZE    : 8448 K
  STORAGE INCREMENT SIZE: 256 K
  LOCKS:      TOTAL:    1048576
  PHYSICAL VERSION: BC9F02E8 D6683A6C
  LOGICAL  VERSION: BC9F02E8 D6683A6C
  SYSTEM-MANAGED PROCESS LEVEL: 8
  XCF GRPNAME    : IXCLO004
  DISPOSITION    : DELETE
  ACCESS TIME    : 0
  MAX CONNECTIONS: 32
  # CONNECTIONS  : 4

 CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME  ASID STATE
 ---------------- -- -------- -------- -------- ---- ------
 ISGLOCK#SC63     02 0002015E SC63     GRS      0007 ACTIVE
 ISGLOCK#SC64     03 00030145 SC64     GRS      0007 ACTIVE
 ISGLOCK#SC65     01 0001014F SC65     GRS      0007 ACTIVE
 ISGLOCK#SC70     04 00040049 SC70     GRS      0007 ACTIVE
```

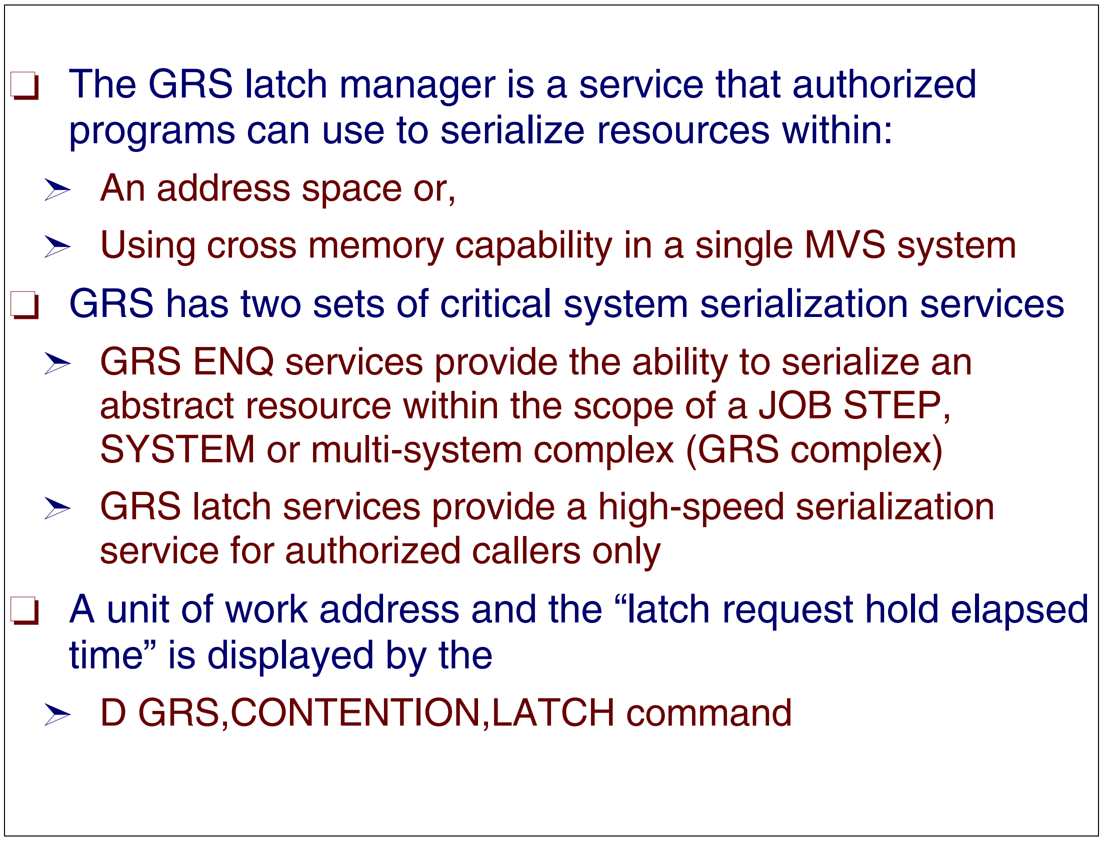
Figure 4-33 GRS star operating (2)

GRS star operating (2)

`DISPLAY GRS,RNL=INCL` shows the contents of the inclusion RNL. See 4.6, “Resource name list (RNL)” on page 195 for more details.

Figure 4-33 shows a partial display of the ISGLOCK structure details gathered with the **`D XCF,STR,STRNAME=ISGLOCK`** command. This display is useful for seeing the ISGLOCK structure CFRM policy definition parameters. The output also shows the systems connected to the structure and their connection names.

4.30 Global resource serialization latch manager



- The GRS latch manager is a service that authorized programs can use to serialize resources within:
  - An address space or,
  - Using cross memory capability in a single MVS system
- GRS has two sets of critical system serialization services
  - GRS ENQ services provide the ability to serialize an abstract resource within the scope of a JOB STEP, SYSTEM or multi-system complex (GRS complex)
  - GRS latch services provide a high-speed serialization service for authorized callers only
- A unit of work address and the “latch request hold elapsed time” is displayed by the
  - D GRS,CONTENTION,LATCH command

Figure 4-34 Global resource serialization latch manager

Global resource serialization latch manager

The global resource serialization latch manager is a service that authorized programs can use to serialize resources within an address space or, using cross memory capability, within a single MVS system. Programs can call the latch manager services while running in task or service request block (SRB) mode.

GRS ENQ and latch services

GRS provides two sets of critical system serialization services. The GRS ENQ services provide the ability to serialize an abstract resource within the scope of a JOB STEP, SYSTEM, or multi-system complex (GRS complex). The GRS complex is usually equal to the sysplex. Via the HW reserve function, DASD volumes can be shared between different systems that are not in the same GRS complex or even in the same operating system, for example, between z/VM and z/OS. Enq/Reserve services can be used by authorized or unauthorized users. Almost every component, subsystem, and many applications use ENQ in some shape or form.

The GRS latch services provide a high-speed serialization service for authorized callers only. User-provided storage is used to manage a lock table that is indexed by a user-defined lock number. GRS latch is also widely used. Very large users are USS, System Logger, RRS, MVS, and many others.

New latch request information

A unit of work address and the "latch request hold elapsed time" is displayed by the **`D GRS,CONTENTION,LATCH`** command output. This solves a long term problem determination issue related to which units of work within an ASID are affected and if displayed, contention is for new or old instances of contention. With current releases, this is impossible to determine.

The difference is between long-term contention and new instances of short-term contention. For example, every time you look, the same players are in contention but you do not know whether something is moving or not. Some latches can be in frequent contention.

The new information provided is as follows:

- The holding and waiting units of work TCB or SRB within an ASID
- The amount of time that the latch is in contention

Not having this information makes it impossible to determine whether a latch was in contention continuously between intervals. For example, it has gone in and out of contention, but every time you look the same players are in contention.

Command examples

If latch contention exists, the system displays the messages shown in Figure 4-35.

```
D GRS,LATCH,CONTENTION
ISG343I 23.00.04 GRS LATCH STATUS 886
LATCH SET NAME: MY.FIRST.LATCHSET
 CREATOR JOBNAME: APPINITJ CREATOR ASID: 0011
  LATCH NUMBER: 1
    REQUESTOR  ASID  EXC/SHR     OWN/WAIT WORKUNIT TCB   ELAPSED TIME
    MYJOB1     001   EXCLUSIVE   OWN      006E6CF0  Y    00:00:40.003
    DATACHG    0019  EXCLUSIVE   WAIT     006E6B58  Y    00:00:28.001
    DBREC      0019  SHARED      WAIT     006E6CF0  Y    00:00:27.003
  LATCH NUMBER: 2
    REQUESTOR  ASID  EXC/SHR     OWN/WAIT WORKUNIT TCB   ELAPSED TIME
    PEEKDAT1   0011  SHARED      OWN      007E6CF0  Y    00:00:32.040
    PEEKDAT2   0019  SHARED      OWN      007F6CF0  Y    00:00:32.040
    CHGDAT     0019  EXCLUSIVE   WAIT     007D6CF0  Y    00:00:07.020
  LATCH SET NAME: SYS1.FIRST.LATCHSET
 CREATOR JOBNAME: INITJOB2 CREATOR ASID: 0019
  LATCH NUMBER: 1
    REQUESTOR  ASID  EXC/SHR     OWN/WAIT WORKUNIT TCB   ELAPSED TIME
    MYJOB2     0019  SHARED      OWN      006E6CF0  Y    00:01:59.030
  LATCH NUMBER: 2
    REQUESTOR  ASID  EXC/SHR     OWN/WAIT WORKUNIT TCB   ELAPSED TIME
    TRANJOB1   0019  SHARED      OWN      006E7B58  Y    01:05:06.020
    TRANJOB2   0019  EXCLUSIVE   WAIT     006E9B58  Y    00:01:05.003
```

Figure 4-35 D GRS,LATCH,CONTENTION command output

Note: Refer to *z/OS MVS Diagnosis: Reference*, GA22-7588 and component-specific documentation for additional information on what specific latch contention could mean and what steps should be taken for different circumstances.

5

z/OS system operations

To help you understand the z/OS environment operation, we present here an overview of the following:

- Operation goals
- Connecting consoles to a sysplex
- Multisystem consoles
- Consoles in a sysplex environment
- Message processing
- Message flow in a sysplex environment
- Command flow in a sysplex environment
- Console-related PARMLIB members
- System commands to display and change console characteristics
- Hardcopy log
- SYSLOG and OPERLOG
- msys for operations

5.1 Planning z/OS operations

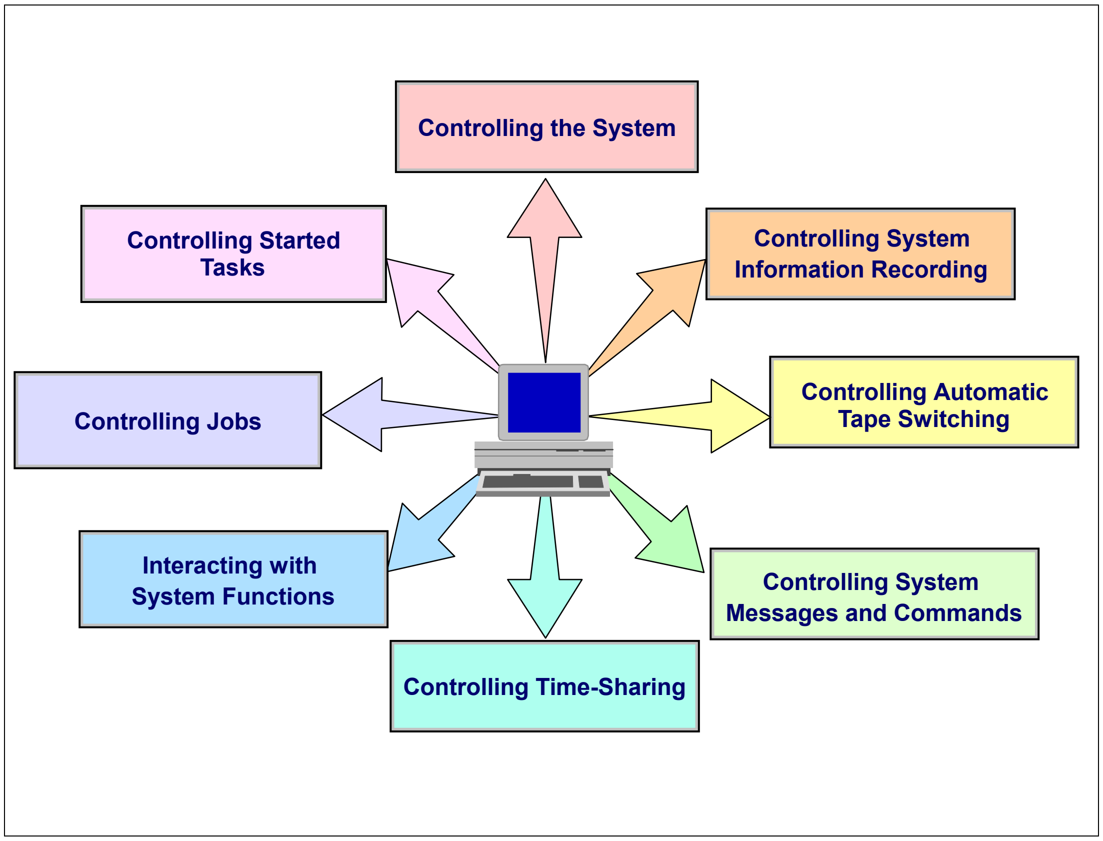

Controlling the System
Controlling Started Tasks
Controlling System Information Recording
Controlling Jobs
Controlling Automatic Tape Switching
Interacting with System Functions
Controlling System Messages and Commands
Controlling Time-Sharing

Figure 5-1 z/OS operations

Planning z/OS operations

Operating a z/OS environment is complex and critical. The operation affects not only application performance and availability, but also the operating system itself. Because of its complexity, planning z/OS operation in a Parallel Sysplex is vital for application availability.

z/OS environment operations involves managing the following:

- Hardware such as processors and peripheral devices, including the consoles where the operators do their work
- Software such as the z/OS operating system, the job entry subsystem (JES), and subsystems like NetView that can control automated operations
- Applications that run on z/OS

Planning z/OS operations must take into account how operators use consoles to do their work and how you want to manage messages and commands. Because messages are also the basis of automated operations, understanding message processing in the z/OS system can help you to plan your automation tasks.

Planning z/OS operations also requires thinking about the particular needs of the installation, for example:

- The business goals
- Policies established by the installation to allow the installation to grow and handle work efficiently

z/OS operations goals

The goals vary from installation to installation, but they are important when planning z/OS operations. However, any installation might consider the following goals:

- Increasing system availability to meet system-level agreements (SLAs).
- Controlling operating activities and functions.
- Simplifying operators' tasks - The complexity of operating z/OS has increased, and you need to think about the tasks and skills of your operators—how operators respond to messages at their consoles—and how you can reduce or simplify their actions. Also, you need to plan z/OS operator tasks in relation to any automation scripts currently implemented.
- Streamlining message flow and command processing - Planning z/OS operations for a system must take into account how operators use consoles to do their work and how you want to manage messages and commands. Because messages are also the basis of automated operations, understanding message processing in an z/OS system can help you plan z/OS automation. Consider how to manage messages and commands:
  - Operators need to respond to messages
  - Routing messages to operator consoles
  - Suppressing messages to help your operators manage
- A single system image allows the operator, for certain tasks, to interact with several images of a product as though they were one image. For example, the operator can issue a single command to all z/OS systems in the sysplex instead of repeating the command for each system.
- A single point of control allows the operator to interact with a suite of products from a single workstation, thereby reducing the number of consoles the operator has to manage.

Planning the operations environment

The z/OS environment at an installation can affect how you plan to meet your operations goals. Your z/OS operating environment might be a single z/OS system or a multisystem environment. Depending on the environment, operating z/OS can involve different approaches to your planning tasks. For example, planning console security for a multisystem environment requires more coordination than for a single z/OS system. But much of the planning you do for a single system can serve as the basis for planning z/OS operations in a multisystem environment.

5.2 Operating a z/OS environment

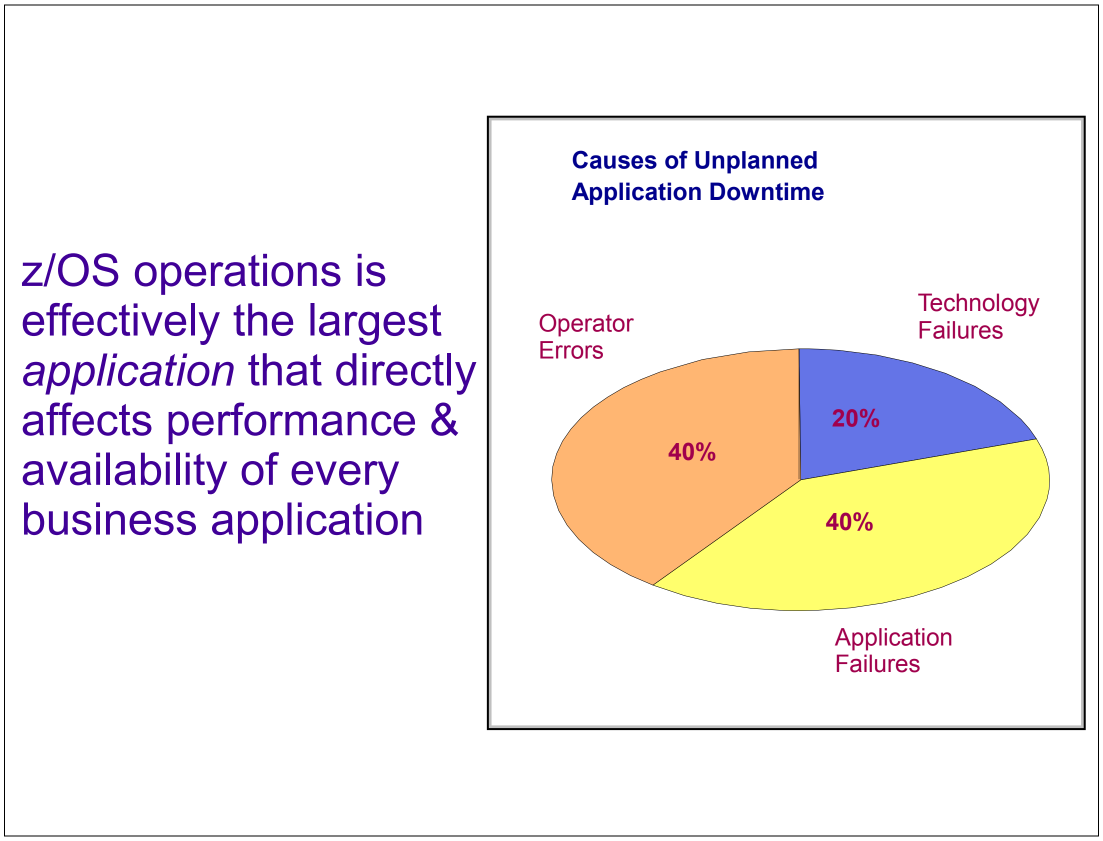

z/OS operations is effectively the largest *application* that directly affects performance & availability of every business application

Figure 5-2 Operating a z/OS environment

Operating a z/OS environment

Controlling the operating system effectively involves operator tasks such as:

- Displaying current system status, such as the number of active jobs and teleprocessing functions, so you can take appropriate actions to operate the system efficiently and to correct potential problems.
- Displaying the status of devices and the availability of paths.
- Communicating among several consoles.
- Communicating within a sysplex. In a sysplex, several z/OS systems interact to process work.
- Setting the time and changing the system parameters.
- Using the system restart function to control certain system functions.
- Responding to messages.
- Activating a WLM service policy for a sysplex.

z/OS command structure

z/OS provides system and subsystem commands that display job and system status either when requested or continually at regular intervals. Other commands route status information to one or more consoles and provide communication among operators in a multiple-console environment, as well as communication with time-sharing users. Many commands let you display information about all the systems in a sysplex, and some commands allow you to control any target system in the sysplex.

5.3 z/OS console types

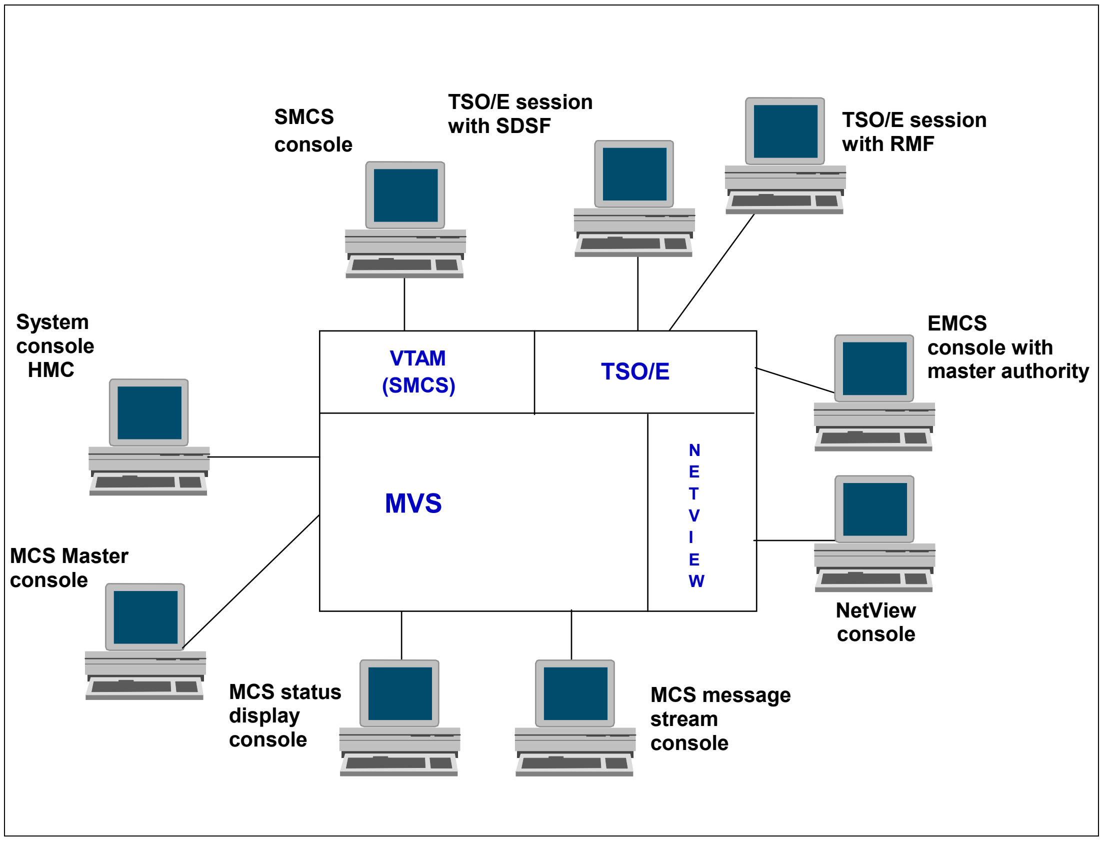


Figure 5-3 z/OS console types

z/OS console types

Figure 5-3 shows the four types of consoles:

MCS MCS consoles are devices that are locally attached to an z/OS system and provide the basic communication between operators and z/OS. MCS consoles are attached to control devices that do not support SNA protocols. You can define up to 99 consoles through the CONSOLxx PARMLIB member. In the past, MCS consoles were connected to non-SNA 3174 terminal controllers. Today MCS consoles are mostly PCs with TN3270E terminal emulation, which are connected to an IBM 2074 console support controller. If you have a z990 or a z890 and as a minimum z/OS R3 installed, you can use an OSA-Express 1000Base-T Ethernet adapter as an OSA integrated console controller. In this configuration you also use PCs with TN3270E terminal emulation.

SMCS SNA Multiple Console Support (SMCS) is a console that uses z/OS Communications Server to provide communication between operators and z/OS instead of direct I/O to the console device. It is implemented as a VTAM application. SMCS consoles can be real 3270-type devices, but usually they are 3270 emulators such as IBM Personal Communications software. Since this type of console uses VTAM for communication, it is not available as an NIP console.

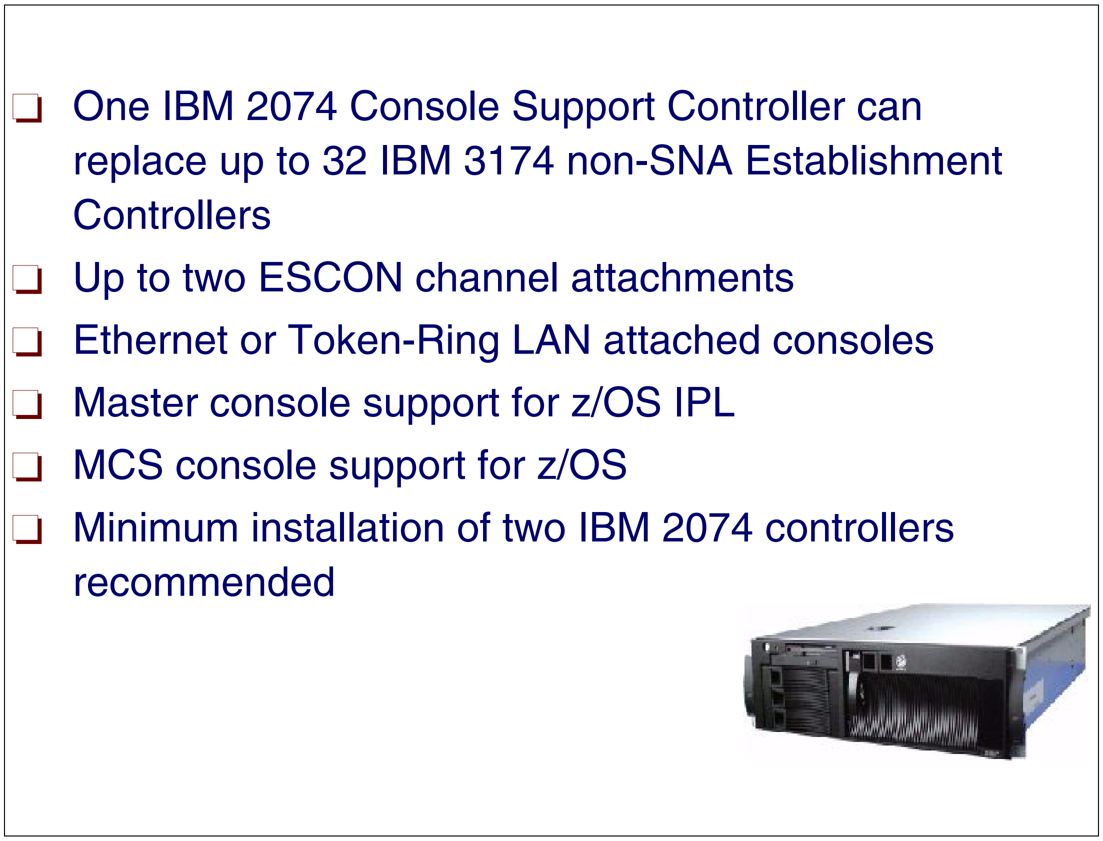
EMCS Extended MCS (EMCS) consoles are programmable consoles defined and activated through an intended programming interface (the MCSOPER macro with REQUEST=ACTIVATE). The number of active EMCS consoles is not restricted.

Programs that use EMCS consoles are the TSO/E CONSOLE command, SDSF (with JES2), (E)JES (with JES3), or user-written programs.

System Console integration is supported by z/OS and allows the hardware management console to support both hardware functions and the operating system IPL, recovery, and problem analysis. The system console in problem determination mode is automatically defined by z/OS during system initialization. The name of this type of console is SYSCONS.

Note: The system console function is provided as part of the Hardware Management Console (HMC). An operator can use the system console to initialize MVS and other system software, and during recovery situations when other consoles are unavailable.

5.4 IBM 2074 console support controller



- ❑ One IBM 2074 Console Support Controller can replace up to 32 IBM 3174 non-SNA Establishment Controllers
- ❑ Up to two ESCON channel attachments
- ❑ Ethernet or Token-Ring LAN attached consoles
- ❑ Master console support for z/OS IPL
- ❑ MCS console support for z/OS
- ❑ Minimum installation of two IBM 2074 controllers recommended

Figure 5-4 IBM 2074 console support controller

One 2074 controller can replace up to 32 3174 non-SNA controllers

Since IBM no longer produces 3174 controllers and most of them were built with parallel channel attachments, there was a need for a new generation of console controllers. The IBM 2074 console support controller is the successor to the 3174s. It is based on an IBM Netfinity® PC in TN3270E mode, which fits into standard 19-inch racks. You are able to connect your consoles to 32 z/OS images over one 2074. That helps you save a lot of cabling in your data center.

Note: The 2074 controller is still supported but can no longer be ordered.

Up to two ESCON channel attachments

Channel attachments between the 2074 and the CPUs can be done in three ways:

- Switched ESCON
- Switched FICON bridge
- Non-switched ESCON

Ethernet or Token Ring LAN-attached consoles

With the IBM 2074 you have console PCs with Personal Communications running TN3270E emulation instead of 3278 terminals. These PCs connect via TCP/IP protocol to the 2074 controller.

Master console support for z/OS IPL

Because you need one master console for IPL in the sysplex, the 2074 delivers this support.

MCS console support for z/OS

MCS consoles are the devices normally used as console types. The IBM 2074 console support controller can also handle this type. You can find a description in 5.3, “z/OS console types” on page 241.

Minimum of two IBM 2074 controllers is recommended

From an availability point of view, we recommend that you install a minimum of two 2074 console support controllers. To eliminate single points of failure, they should also be configured with two ESCON ports.

Refer to *Introducing the IBM 2074 Control Unit,* SG24-5966 for more details.

5.5 2074 console support controller configuration

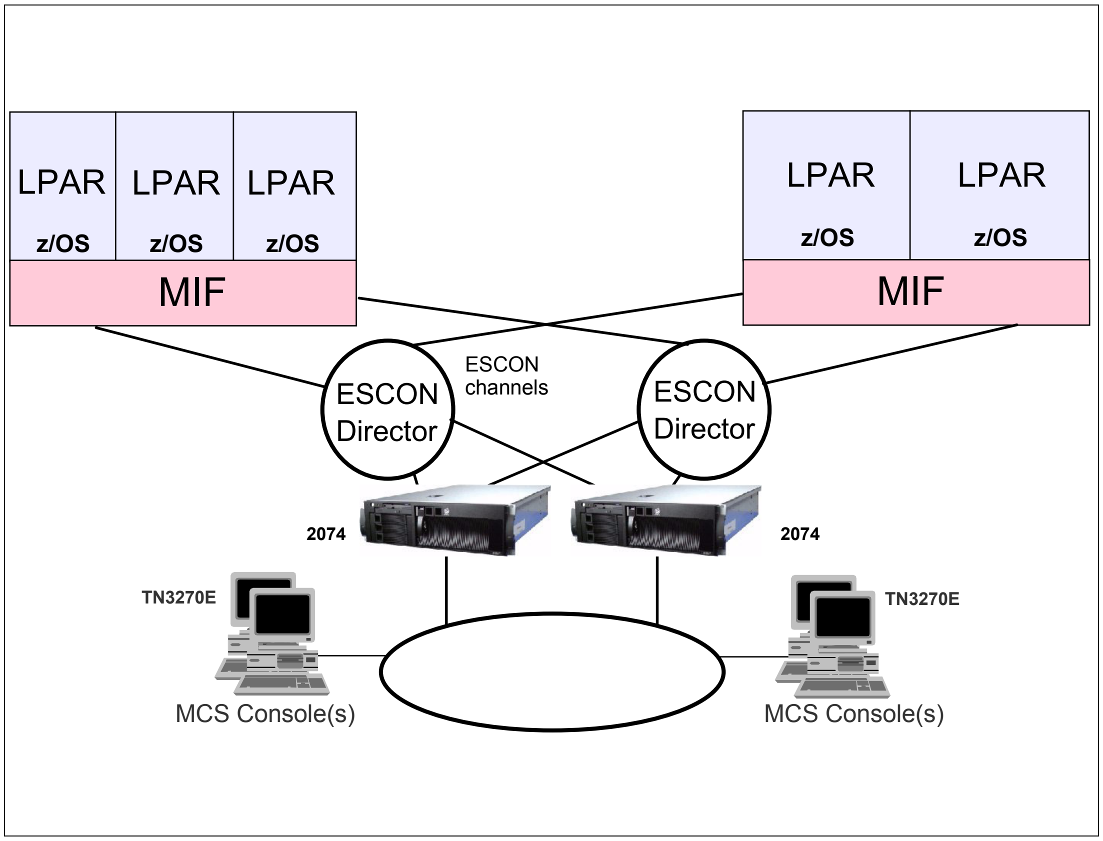


Figure 5-5 2074 configuration

2074 configuration

Figure 5-5 shows a configuration with multiple LPARs on two CECs. With the use of ESCON Multiple Image Facility (EMIF) you can share the 2074s between several LPARs. You can either connect the 2074 directly to the CPU or, as shown here, using ESCON directors. MCS consoles are attached to the 2074 via TCP/IP protocol and TN3270E terminal emulation. These consoles are PCs. Token Ring (4/16 Mb/sec) and Fast Ethernet (10/100 Mb/sec) are supported. The 2074 converts the TN3270E sessions so that they appear as local non-SNA terminals to the z/OS operating systems.

5.6 OSA integrated console controller

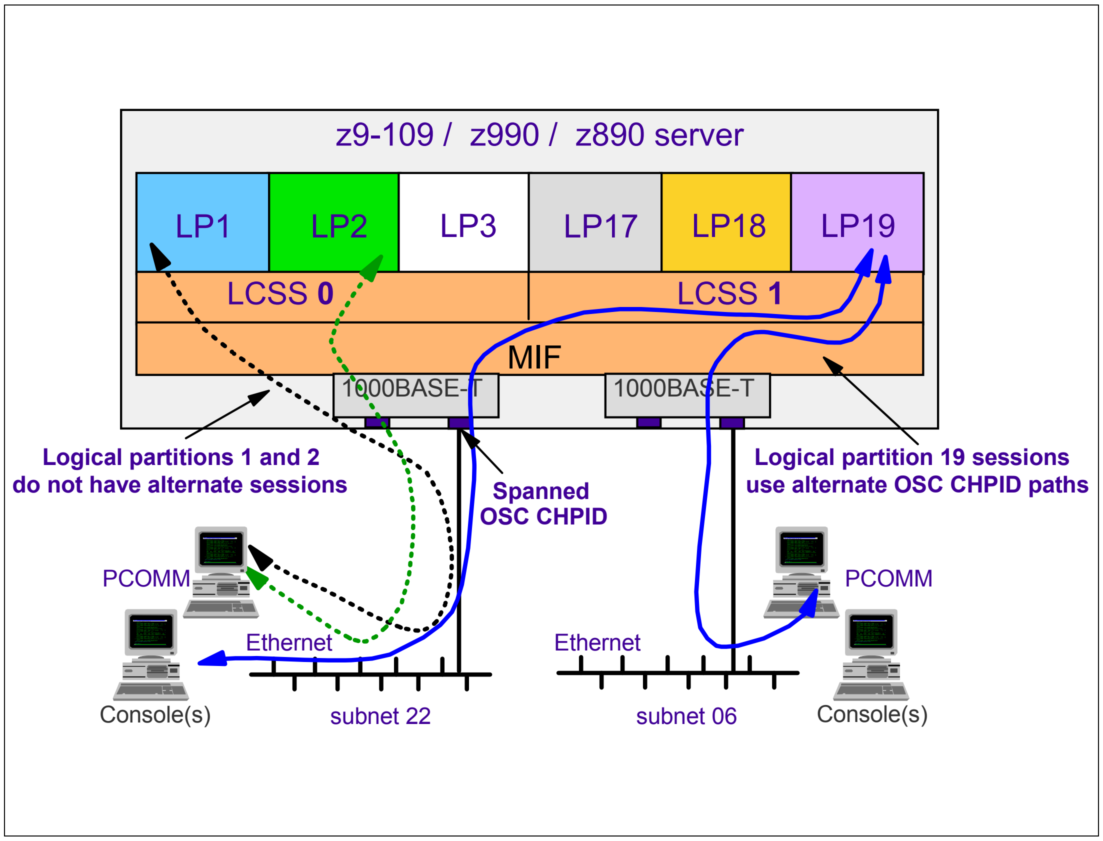

z9-109 / z990 / z890 server
LP1
LP2
LP3
LP17
LP18
LP19
LCSS 0
LCSS 1
MIF
1000BASE-T
1000BASE-T
Logical partitions 1 and 2 do not have alternate sessions
Spanned OSC CHPID
Logical partition 19 sessions use alternate OSC CHPID paths
PCOMM
PCOMM
Ethernet
Ethernet
Console(s)
Console(s)
subnet 22
subnet 06

Figure 5-6 OSA integrated console controller

OSA integrated console controller

The OSA-Express Integrated Console Controller (OSA-ICC) support is a no-charge function included in Licensed Internal Code (LIC) on z9-109, z990, and z890 servers. It is available via the OSA-Express2 and OSA-Express 1000BASE-T Ethernet features, and supports Ethernet-attached TN3270E consoles.

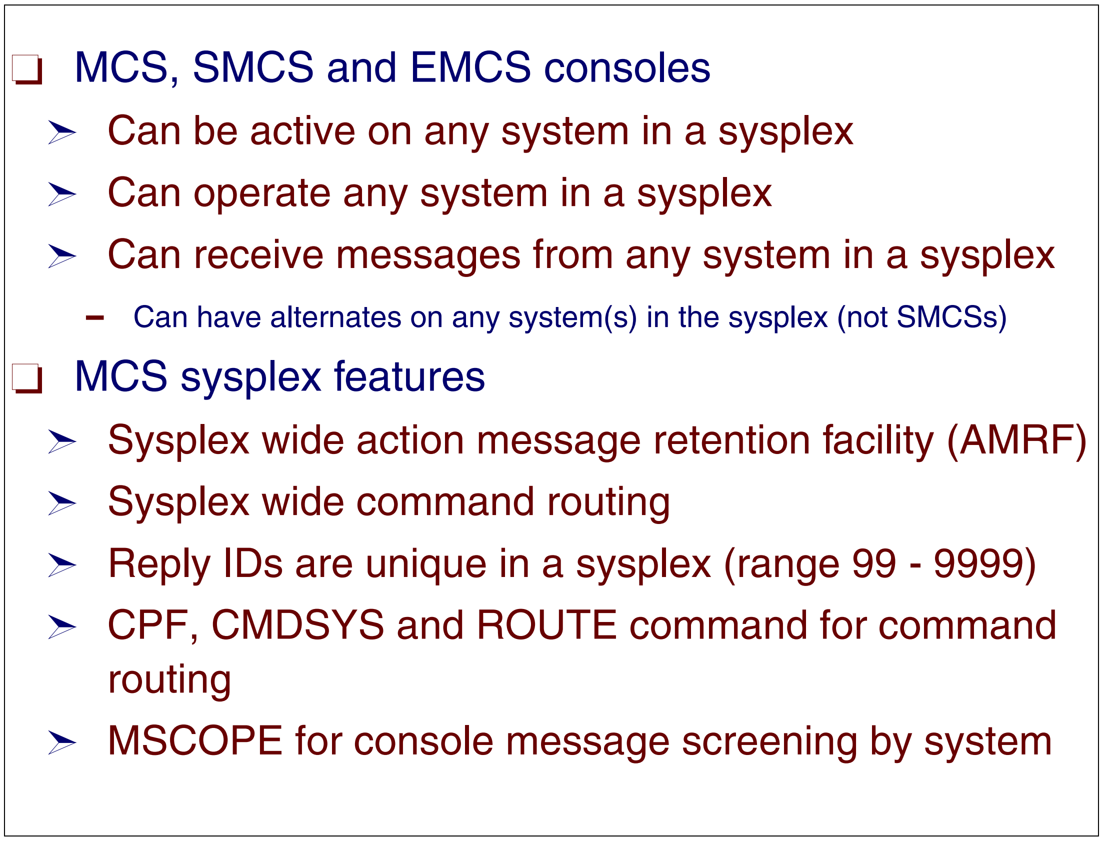
The OSA-ICC provides a system console function at IPL time and operating systems support for multiple logical partitions. Console support can be used by z/OS, z/OS.e, z/VM, zVSE, and TPF. The OSA-ICC also supports local non-SNA DFT 3270 and 328x printer emulation for TSO/E, CICS, IMS, or any other 3270 application that communicates through VTAM.

With the OSA-Express2 and OSA-Express 1000BASE-T Ethernet features, the OSA-ICC is configured on a port-by-port basis, using Channel Path Identifier (CHPID) type OSC. Each port can support up to 120 console session connections, can be shared among logical partitions using Multiple Image Facility (MIF), and can be spanned across multiple Logical Channel Subsystems (LCSSs).

Figure 5-5 shows an example of the OSA-Express Integrated Console Controller in a single system configuration.

Refer to *OSA-Express Integrated Console Controller Implementation Guide,* SG24-6364 for more details.

5.7 Multisystem consoles in a sysplex



- MCS, SMCS and EMCS consoles
  - Can be active on any system in a sysplex
  - Can operate any system in a sysplex
  - Can receive messages from any system in a sysplex
    - Can have alternates on any system(s) in the sysplex (not SMCSs)
- MCS sysplex features
  - Sysplex wide action message retention facility (AMRF)
  - Sysplex wide command routing
  - Reply IDs are unique in a sysplex (range 99 - 9999)
  - CPF, CMDSYS and ROUTE command for command routing
  - MSCOPE for console message screening by system

Figure 5-7 Multisystem consoles in a sysplex

MCS, SMCS, and EMCS consoles

Consoles can be active on any system in a sysplex and can provide sysplex-wide control. MCS uses XCF services for command and message transportation between systems and thus provides a single system image for the operators. Normal message presentation is controlled by a console's ROUTCDE and MSCOPE settings, and the UD (undeliverable message) attribute.

MCS sysplex features

MCS multisystem support features:

- Sysplex-wide action message retention facility (AMRF)
- Sysplex-wide unique reply IDs
- Sysplex-wide command routing through:
  - ROUTE operator command
  - Command prefix facility (CPF)
  - CMDSYS setting for a console (through the CONSOLE statement in the CONSOLxx parmlib member or CONTROL V,CMDSYS= operator command)

SMCS consoles support most of the same functions that MCS consoles do. There are a few differences:

- Synchronous WTO/R, also known as Disabled Console Communication Facility (DCCF), is not supported for SMCS consoles. The system console (HMC) or an MCS console must be used instead.
- SMCS consoles are not available during NIP. The system console or an MCS console must be used instead.
- SMCS consoles require VTAM to be active. If VTAM is not active for any reason, then SMCS consoles will not be available. The system console and MCS consoles do not rely on VTAM, and these could be used instead.
- SMCS consoles must be activated differently than MCS consoles. The activation process depends on the console definitions, but in all cases, VARY CONSOLE and VARY CN,ONLINE do not work for SMCS.
- SMCS does not support output-only (message stream and status display) consoles. SMCS consoles must always be full-capability consoles. MCS supports output-only consoles.
- SMCS does not support printer consoles, and cannot be used as hardcopy devices. MCS supports printer consoles and hardcopy devices.

5.8 Sysplex operating environment

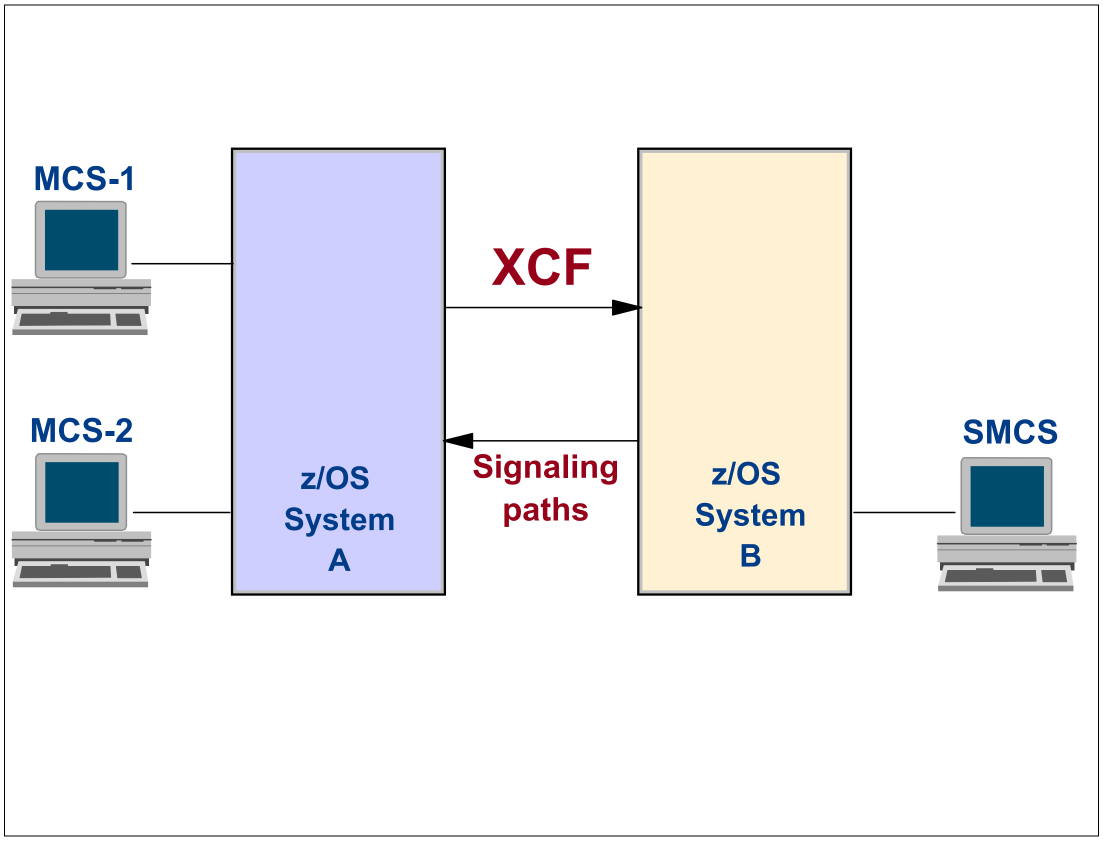


Figure 5-8 Sysplex operating environment

Sysplex operating environment

The sysplex operating environment provides a simpler and more flexible way to operate consoles in a multisystem environment. Many changes were introduced into multiple console support (MCS) to support the sysplex environment.

In a sysplex, MCS or SMCS consoles can:

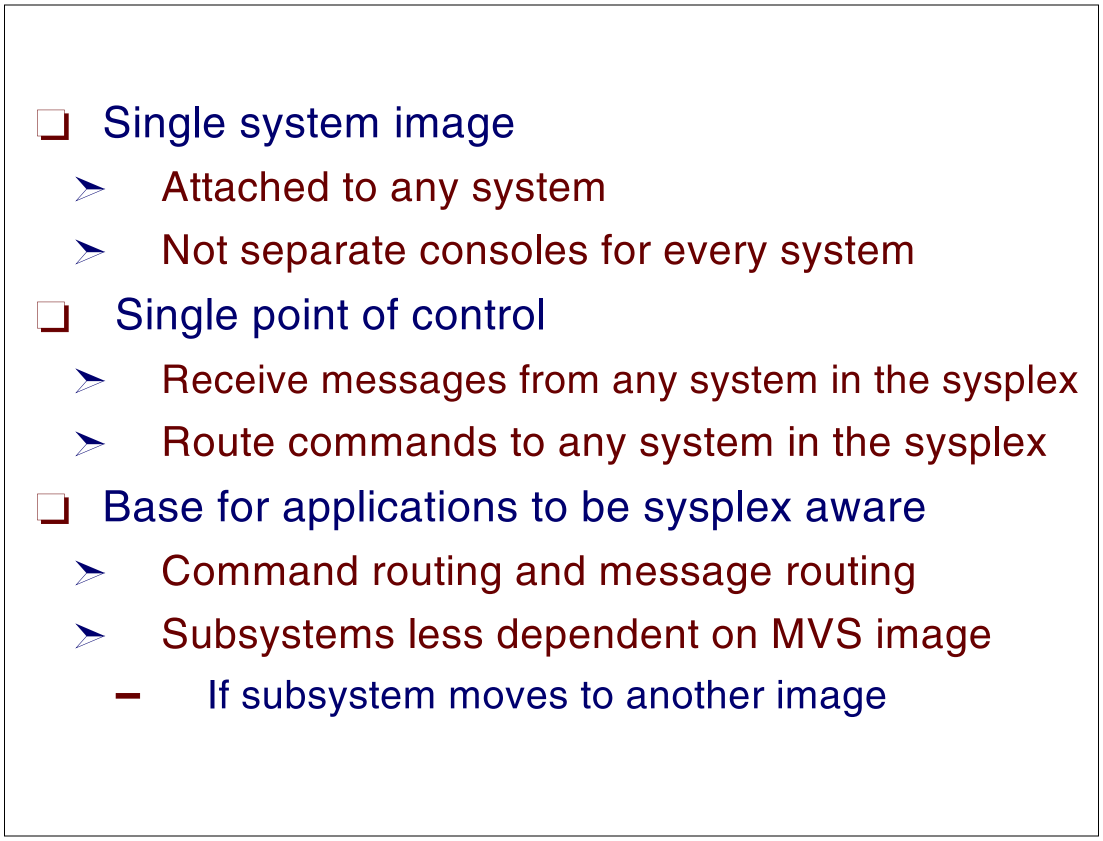
- Be attached to any system
- Receive messages from any system in the sysplex
- Route commands to any system in the sysplex

Therefore, new considerations are necessary when defining MCS consoles in this environment, such as:

- There is no requirement that each system have consoles attached.
- The 99 console limit for the sysplex is *extended* with the use of extended MCS consoles (EMCS). This adds greater flexibility when attaching consoles.
- A sysplex, which can have up to 32 systems, can be operated from a single console.
- Multiple consoles can have master command authority.

With MCS consoles in a sysplex, no matter where they are attached, it is possible to control any system in the sysplex. The ability to assign each console a unique name and unique characteristics greatly eases the task of system management.

5.9 Support for multisystem management



- ❑ Single system image
  - ➢ Attached to any system
  - ➢ Not separate consoles for every system
- ❑ Single point of control
  - ➢ Receive messages from any system in the sysplex
  - ➢ Route commands to any system in the sysplex
- ❑ Base for applications to be sysplex aware
  - ➢ Command routing and message routing
  - ➢ Subsystems less dependent on MVS image
    - – If subsystem moves to another image

Figure 5-9 Support for multisystem management

Support for multisystem management

In a sysplex, the major tasks of operating an individual z/OS image do not change very much. Consoles are used to receive messages and issue commands to accomplish system tasks. With the existence of multiple z/OS images and multiple subsystems, the potential exists for the operator to receive an enormous number of messages, since there is the capability to route all messages from all z/OS systems to a single console.

Single system image

A single system image allows the operator, for certain tasks, to interact with several images of a product as though they were one image. The operator can issue a single command to all z/OS systems in the sysplex instead of repeating the command for each system.

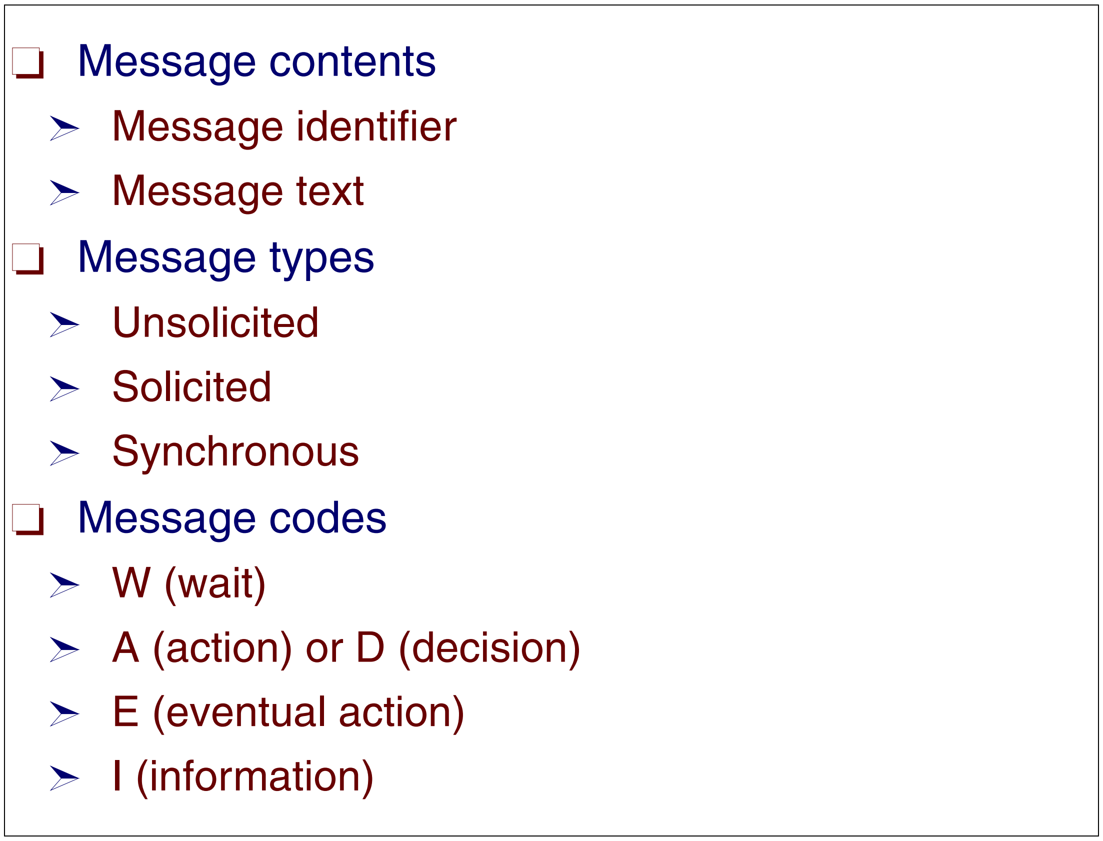
Single point of control

A single point of control allows an operator to interact with a suite of products from a single workstation. An operator can accomplish a set of tasks from a single workstation, thereby reducing the number of consoles the operator has to manage.

Base for applications to be sysplex aware

Multisystem consoles can be used by applications to be sysplex aware due to the fact that no matter on which system the applications exist, command routing and message routing can be done. The application does not have to exist on the same system as a console used to communicate with it.

5.10 Message processing



- Message contents
  - Message identifier
  - Message text
- Message types
  - Unsolicited
  - Solicited
  - Synchronous
- Message codes
  - W (wait)
  - A (action) or D (decision)
  - E (eventual action)
  - I (information)

Figure 5-10 Message processing

Message processing and contents

When the z/OS system and any program running under the z/OS system require communication with operators, they issue messages. The z/OS write to operator (WTO) and the write to operator with reply (WTOR) macro services cause messages to be routed to the operators and hardcopy log.

A message contains text to provide information, describe an error, or request an operator action. A message contains an identifier, which is usually a three-letter prefix to identify the system component that produced the message, and a message serial number to identify the individual message. The identifier might contain other information, for example, a message type code.

```
IOS351I DYNAMIC CHANNEL PATH MANAGEMENT NOT ACTIVE
....
IEA989I SLIP TRAP ID=X33E MATCHED
....
*IXC585E STRUCTURE SYSTEM_OPERLOG IN COUPLING FACILITY CF1
PHYSICAL STRUCTURE VERSION BC7E4AB5 792253EC,
IS AT OR ABOVE STRUCTURE FULL MONITORING THRESHOLD OF  80%.
ENTRIES:  IN-USE:      26771 TOTAL:      34220,  78% FULL
ELEMENTS: IN-USE:      27416 TOTAL:      34270,  80% FULL
```

Figure 5-11 Message examples

Message types

There are three types of messages:

Unsolicited	A message routed by a routing code; that is, the message is issued by the system and it is not a response to a command.
Solicited	These messages are responses to commands issued by an operator and normally routed back to the console where the command was issued. Some operator commands support the L= operand, which specifies the display area, console or console name, or both, of the console where the command response is to be routed.
Synchronous	There are situations in which system operations cannot continue until the system operator takes some external action. An example might be an authorized application detecting a critical problem that warrants stopping the entire system to correct. Using the LOADWAIT macro and the WTO macro with the WSPARM and SYNCH=YES parameters stops the system so the operator can correct a problem, if possible. By using LOADWAIT and WTO, you issue a synchronous message to the operator and place the system into a restartable or non restart able wait state. A multiple line WTOR can be used only when SYNCH=YES is also specified. SYNCH=YES indicates the request is to be processed synchronously. This type of WTOR is used in error and recovery environments, when normal message processing cannot be used. The message is sent to the console, and the reply is obtained immediately, before control is returned to the caller.

The essential difference between solicited and unsolicited messages is that solicited messages are (normally) routed by the console ID that issued the command; while unsolicited messages go to consoles that are receiving the routing codes or other general routing attributes used for the message.

Message codes

The IBM message type codes, specified by a letter following the message serial number, are associated with message descriptor codes as shown in Table 5-1.

Table 5-1 Message type and descriptor codes

Type Code	Descriptor Code
W (wait)	1
A (action) or D (decision)	2
E (eventual action)	3
I (information)	4 through 10
E (critical eventual action)	11
I (information)	12 and 13

The descriptor code identifies the significance of the message. Descriptor codes are specified in the DESC parameter of the WTO or WTOR macro. Messages can be routed to a specific console through the CONSID or CONSNAME parameter of the WTO or WTOR macro. Messages can also be routed to a group of consoles by assigning routing codes through the ROUTCDE parameter of the WTO or WTOR macro. The routing code identifies where a message is displayed. A console definition includes the set of routing codes it should receive. More than one routing code can be assigned to a message.

Refer to *z/OS MVS Routing and Descriptor Codes,* SA22-7624 for more details.

5.11 Message flow in a sysplex environment

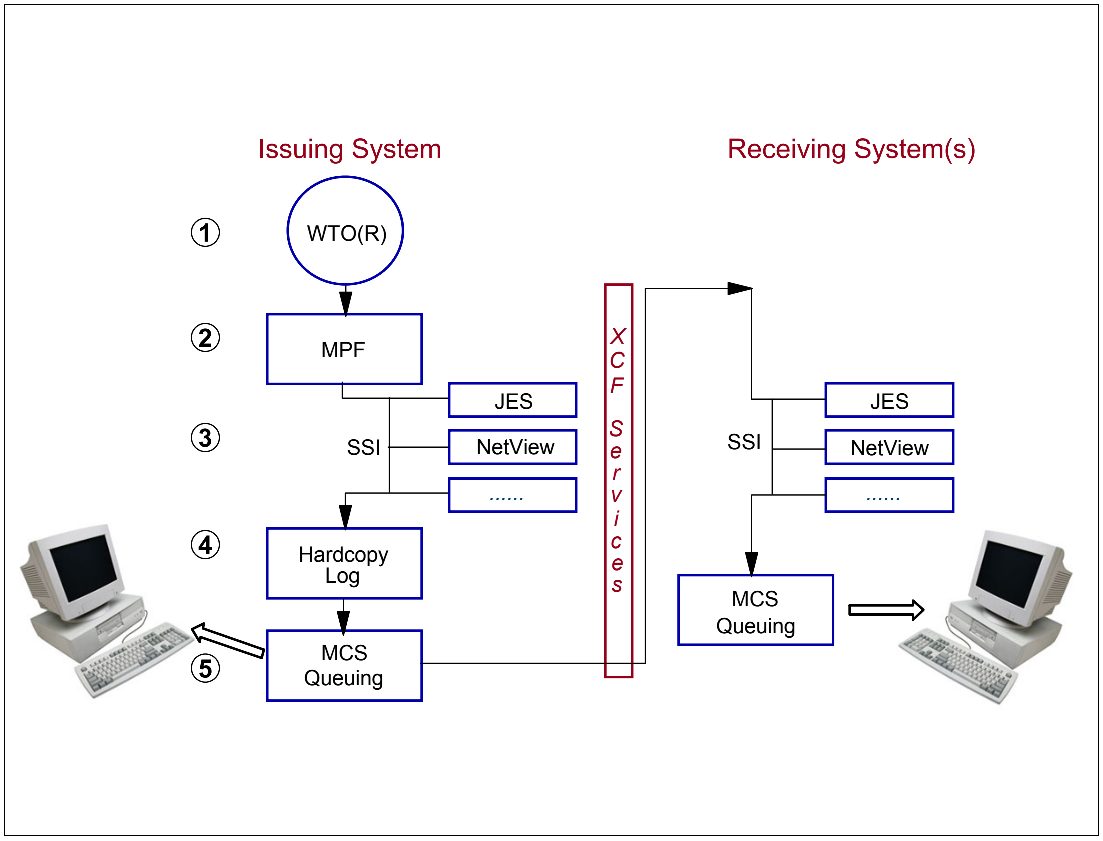


Figure 5-12 Message flow in a sysplex environment

Message flow in a sysplex environment

In a sysplex, a message is routed to all active consoles on all systems that are eligible to receive that particular message.

Figure 5-12 shows the message flow in a sysplex. It is important to understand this flow because in a sysplex environment, message flow is changed to send messages issued on one system to other systems in the sysplex using XCF services.

1. The z/OS write to operator (WTO) and the write to operator with reply (WTOR) macro services cause messages to be routed to the consoles and the hardcopy log.
2. First on the issuing system, the message is processed by the message processing facility (MPF). This processing is based on entries in the MPFLSTxx PARMLIB member. MPF processing allows an installation to influence how WTO and WTOR messages are to be processed. Through the MPFLSTxx member, you can specify some processing options for a message.
3. Following MPF processing, the message is broadcast to all active subsystems that request to receive control for the WTO subsystem interface (SSI) function code 9. The subsystem must use the IEAVG700 interface to indicate that all WTOs and WTORs are to be broadcast. The message is presented to each subsystem in turn. Each subsystem may inspect the message and process it as appropriate. A subsystem can alter write-to-operator queue element (WQE) fields, in which case later subsystems on the SSI will see the changed WQE. A WQE is an internal control block that contains the message

text and all related information for that message. The IHAWQE macro maps the WQE fields.

For example, when NetView is using the SSI rather than an extended MCS console for z/OS communication, NetView on the SSI inspects all messages to see whether they are marked by MPF as eligible for automation. NetView intercepts automation message text and selected attributes from the WQE and sends the data to the NetView address space for further processing. NetView does not modify the actual WQE.

4. After the message has been inspected by all active subsystems, it is written to the hardcopy log (usually the SYSLOG data set, the operations log (OPERLOG), or both) unless hardcopy logging is suppressed by an exit. OPERLOG is a log stream maintained in a Coupling Facility that uses the System Logger to record and merge communications about programs and system functions from each system in a sysplex. The messages are logged using message data blocks (MDBs), which provide more data than is recorded in the SYSLOG.
5. Finally, the message is routed for display on the appropriate MCS and extended MCS consoles. The routing may require message transportation using XCF services to other systems in the sysplex because some receiving consoles may not be physically attached to the system where the message was issued.

XCF services

After the XCF transportation on the receiving system, the message goes through the SSI loop, but it is not logged, and finally the message is processed by the message queueing tasks to be displayed on the consoles.

If a message is destined for a specific console that is not active in the sysplex, it is logged and discarded unless it is an action message or WTOR message, in which case it is processed as an undelivered message. It is sent to all active consoles receiving UD messages. The master console is a UD receiver by default.

MCS queueing of messages

Messages that are already *delivered* to an active extended MCS console, but not yet *retrieved*, are purged from the MCS queues when the console is deactivated; that is, unprocessed queued messages are not rerouted.

MSCOPE specifications

The MSCOPE specification on the CONSOLE statement in the CONSOLxx PARMLIB member allows you to screen those systems in the sysplex from which a console is to receive messages not explicitly routed to the console.

5.12 Command flow in a sysplex environment

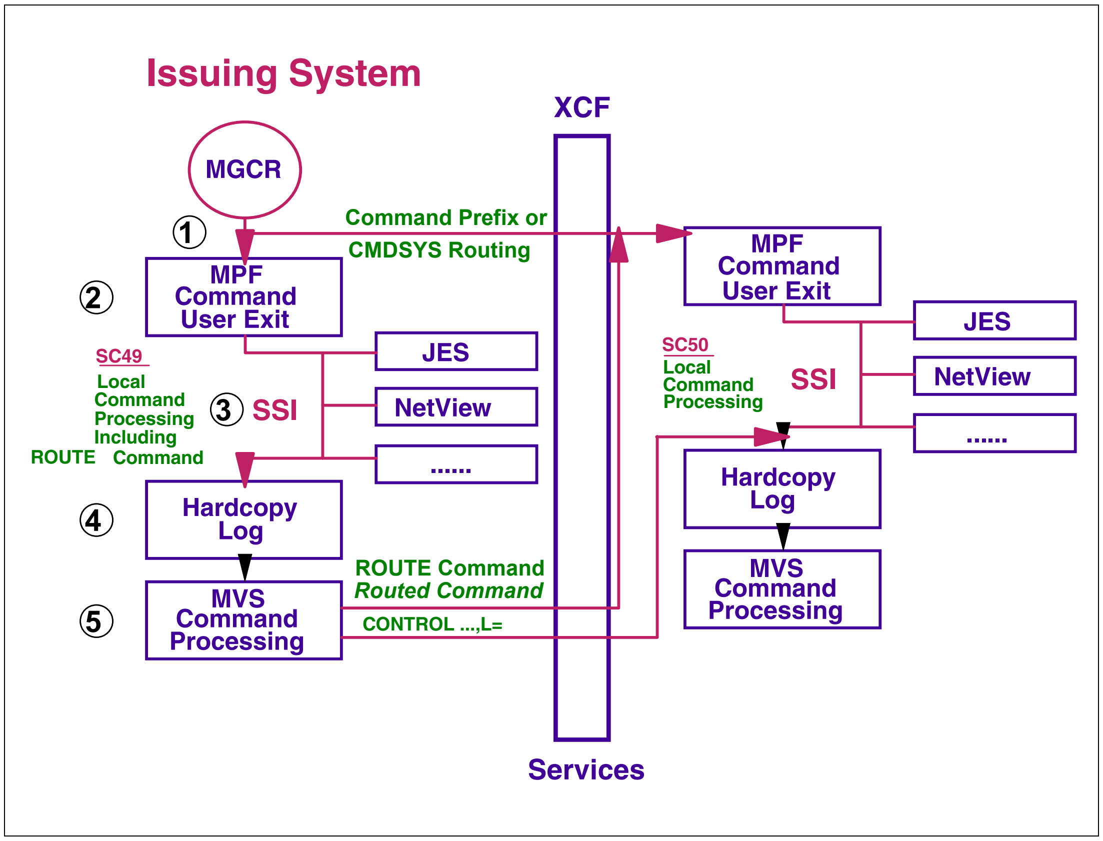


Figure 5-13 Command flow in a sysplex environment

Command flow in a sysplex environment

When an operator command is entered through the MGCR(E) macro service, the following processing flow takes place:

1. If the command contains a prefix or the console has a CMDSYS specification that directs the command to another system, the command is immediately transmitted using XCF services to the processing system.

   For command prefix and CMDSYS routing, the command is first transported to the receiving system before any system symbol substitution takes place.

   A REPLY command is sent to the system where the WTOR was issued. If the REPLY command text contains system symbols, substitution occurs on the receiving system.

2. If the command is not transmitted to another processing system, it is processed on the issuing system by the installation MPF command exit routines. The exits are specified using the CMD statement in the MPFLSTxx PARMLIB member. These exits can perform authority checking, modifying the command text or the command processing.

   For commands containing system symbols, substitution has occurred before the exit is entered.

3. The command is then broadcast on the subsystem interface (SSI) to all active subsystems. Each subsystem inspects the command and decides whether to process it. The subsystems base the decision on the command prefix characters of the command string. For example, by default, NetView looks for a percent sign (%) and processes the commands starting with the % sign.

Subsystem selects message

When a subsystem decides to process the command, the command is passed to subsystem processing, and a return code is set to indicate that the command was processed by a subsystem.

At this point in processing, all system symbol substitution has occurred. The original command text is also available.

4. After the command has been examined by all active subsystems, it is logged to the hardcopy log (usually SYSLOG or OPERLOG).

   The hardcopy log contains the command before any system symbols contained in the command have been substituted, and it also contains the command after substitution has occurred.

5. If none of the subsystems have marked the command as having been processed, it is assumed to be an z/OS command and is passed to the appropriate z/OS command processor. If a command processor does not exist, an error message is issued stating that the command is invalid.

5.13 Console-related parmlib members

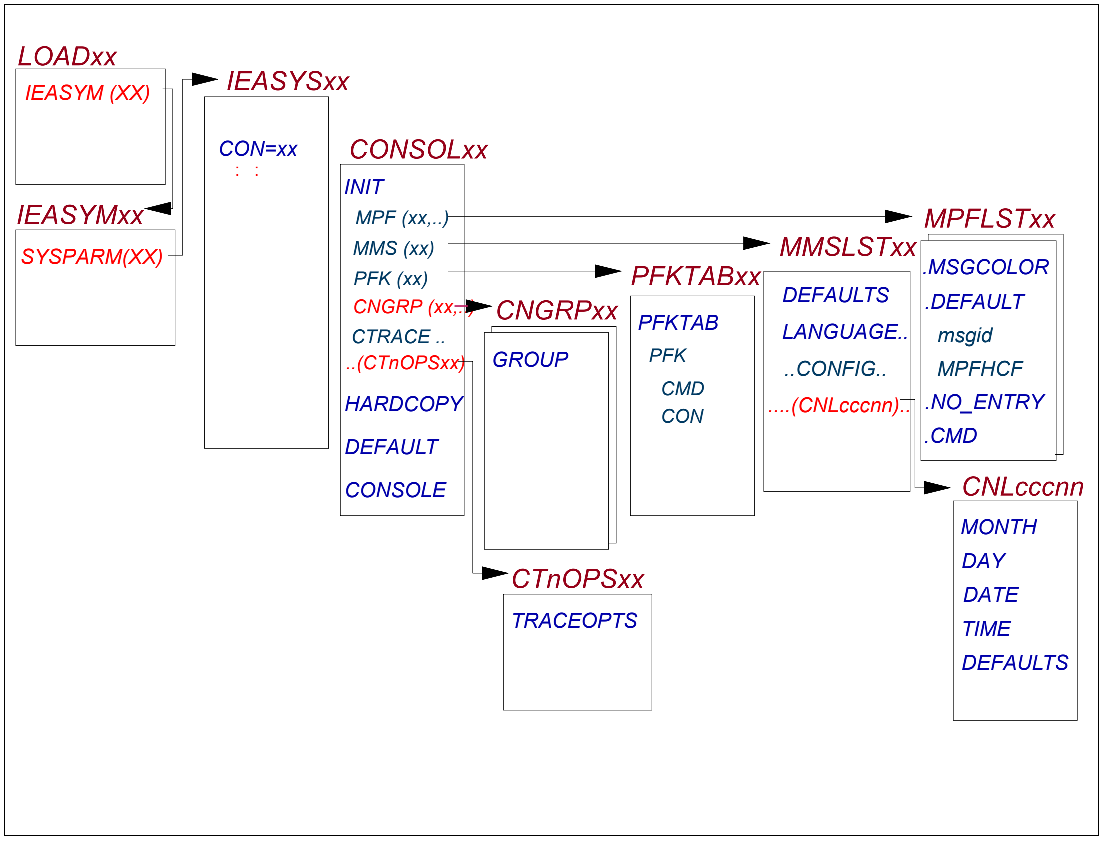


Figure 5-14 Console-related PARMLIB members

Console-related PARMLIB members

Figure 5-14 summarizes the console-related PARMLIB members, their *chaining*, and also lists the various statements in each member.

Besides the CONSOLxx PARMLIB member, several other related PARMLIB members affect console operations and are shown in Figure 5-14. Within the CONSOLxx member, you may reference:

CNGRPxx	This member defines console groups for alternate console switching.
CT*n*OPSxx	This member specifies the component tracing options for the operation services (OPS) component.
MMSLSTxx	This member is used to display translated U.S. English messages into another language that your installation has provided.

This member references the message configuration member CNLcccxx, which defines how translated messages are to be displayed at your installation.

The primary objective of the MMS is to provide a programmable interface for message translation. The z/OS message service allows components of, or products running on, z/OS to translate U.S. English messages into the user's native language.

MCS messages displayed on locally attached MCS consoles are not translated. When the z/OS message services (MMS) is active, the TSO/E

	CONSOLE command attempts to translate console messages to the primary language specified in the user's profile (UPT).
MPFLSTxx	This member is used for message processing control.
PFKTABxx	This member is used to define any PFK tables for the MCS consoles.
CNLcccxx	This member is used to specify the time and date format for translated messages by using the MONTH, DAY, DATE, and TIME statements.

Refer to *z/OS MVS Initialization and Tuning Reference,* SA22-7592 for more details.

5.14 Display console status information

```
DISPLAY CONSOLES,ACTIVE
IEE889I 16.36.19 CONSOLE DISPLAY 994
MSG: CURR=0    LIM=1500 RPLY:CURR=2    LIM=999  SYS=SC63      PFK=00
 CONSOLE            ID   --------------- SPECIFICATIONS ---------------
 SYSLOG                  COND=H      AUTH=CMDS         NBUF=N/A
                         ROUTCDE=ALL
 OPERLOG                 COND=H      AUTH=CMDS         NBUF=N/A
                         ROUTCDE=ALL
NO CONSOLES MEET SPECIFIED CRITERIA

 DISPLAY CONSOLES,BACKLOG
 IEE889I 16.45.53 CONSOLE DISPLAY 000
 MSG: CURR=0    LIM=1500 RPLY:CURR=2    LIM=999  SYS=SC63      PFK=00
  CONSOLE            ID   --------------- SPECIFICATIONS ---------------
 NO CONSOLES MEET SPECIFIED CRITERIA
  WTO BUFFERS IN CONSOLE BACKUP STORAGE =        0
  ADDRESS SPACE WTO BUFFER USAGE
  NO ADDRESS SPACES ARE USING MORE THAN   500 WTO BUFFERS
  MESSAGES COMING FROM OTHER SYSTEMS - WTO BUFFER USAGE
  NO WTO BUFFERS ARE IN USE FOR MESSAGES FROM OTHER SYSTEMS
```

Figure 5-15 Display console status information

Display console status information

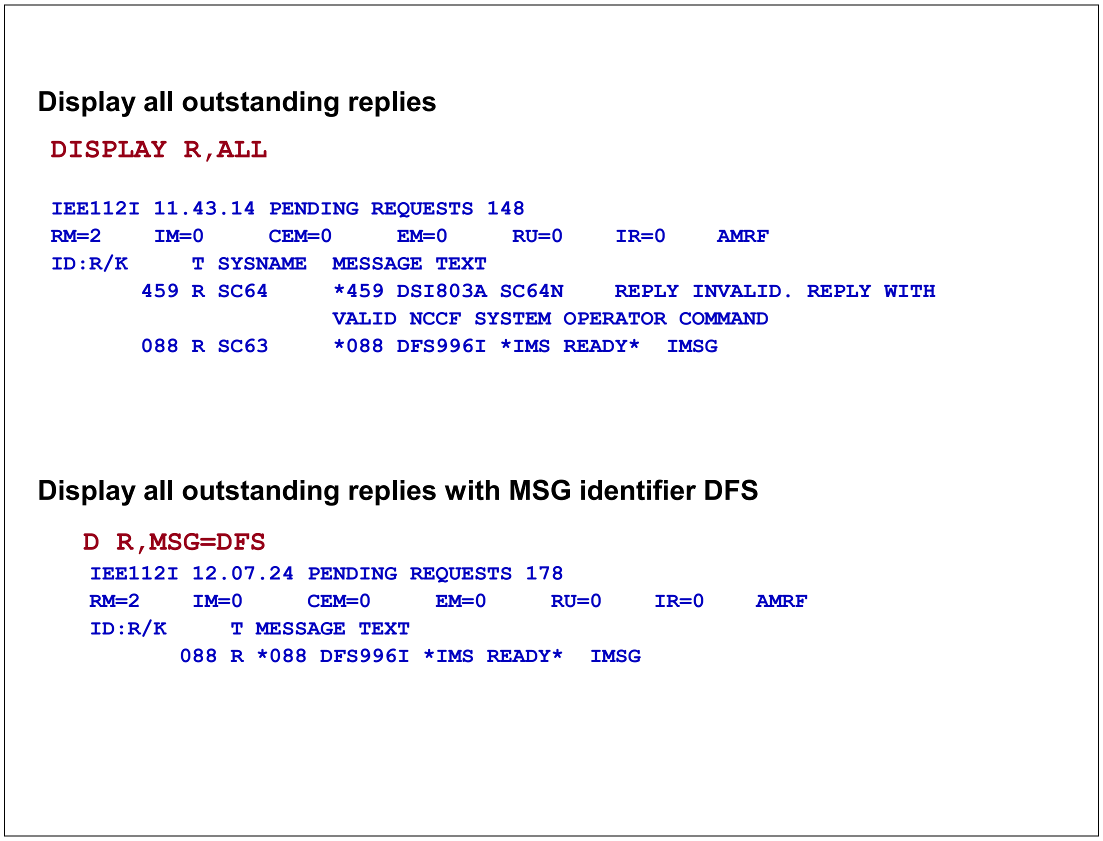
Use the DISPLAY CONSOLES,ACTIVE command to display the status of all consoles in the sysplex including SMCS. The fields are defined as follows:

MSG: CURR=0	Current use of WTO buffers
LIM=1500	Specified limit of max WTO buffers
RPLY: CURR=2	Number of WTOR messages displayed and waiting for reply
LIM=999	Specified limit of max WTOR buffers
SYS=SC63	SC63 is the system the console is defined to
PFK=00	Active PFK table defined in CONSOLxx parmlib member
CONSOLE ID	Console identifier
AUTH=CMDS	Command authority for the console
ROUTCDE=ALL	All routing codes established for the console

The **`DISPLAY CONSOLES,BACKLOG`** command is useful if there is a WTO flood coming from another system.

If you need information about EMCS consoles, use the **`DISPLAY EMCS`** command. See 5.16, “Display all defined EMCS” on page 261 for additional command examples for displaying EMCS information.

5.15 Display system requests



Display all outstanding replies

```
DISPLAY R,ALL

IEE112I 11.43.14 PENDING REQUESTS 148
RM=2    IM=0     CEM=0     EM=0     RU=0    IR=0    AMRF
ID:R/K     T SYSNAME  MESSAGE TEXT
       459 R SC64     *459 DSI803A SC64N    REPLY INVALID. REPLY WITH
                      VALID NCCF SYSTEM OPERATOR COMMAND
       088 R SC63     *088 DFS996I *IMS READY*  IMSG
```

Display all outstanding replies with MSG identifier DFS

```
D R,MSG=DFS
IEE112I 12.07.24 PENDING REQUESTS 178
RM=2    IM=0     CEM=0     EM=0     RU=0    IR=0    AMRF
ID:R/K     T MESSAGE TEXT
       088 R *088 DFS996I *IMS READY*  IMSG
```

Figure 5-16 Display system requests

Display system requests

`DISPLAY R,ALL` informs you about outstanding system requests requiring operator actions. You can display the following:

- WTOR messages
- Action messages saved by AMRF
- Action messages issued by the communication task
- Action messages that where not displayed on all necessary consoles

Use the following form of the **`DISPLAY`** command to display outstanding messages requiring operator action. You can request that the system display:

- The immediate action messages (descriptor codes 1 or 2), eventual action messages (descriptor code 3), and critical eventual action messages (descriptor code 11)
- The device numbers of devices waiting for mount requests to be fulfilled
- The device numbers of devices waiting for operator intervention
- The status of the action message retention facility
- An alphabetical list of keynames of outstanding action messages
- The messages issued by a specified system
- The messages that await operator response at a specified console
- The messages that have specific routing codes

5.16 Display all defined EMCS

```
D EMCS,S,ST=L
IEE129I 14.47.50 DISPLAY EMCS 916
DISPLAY EMCS,S,ST=L
NUMBER OF CONSOLES MATCHING CRITERIA: 114
SC63     *ROUT091 SC64     *ROUT092 SC65     *ROUT093 SYSJ3N01
SC70     *ROUT094 BZZOSC63 BZZOSC65 PAOLOR2  BART     HERING
TWSRES1  TROWELL  VAINI    CBDHSS00 RC63     KYNEF    BARTR1
*ROUT095 TWSRES6  RCONWAY  *ROUT096 *ROUT097 *ROUT098 *ROUT099
*ROUT100 SYSJ3N02 AUTGSS63 AHW00263 AAUTO164 MBEAL    AUTSYS64
AUTREC64 AUTRPC64 RC65     TWSRES7  KEYES    SCHOEN   EMCSCON
*SYSLG63 *OPLOG01 *SYSLG64 *OPLOG02 *SYSLG65 *OPLOG03 SYSJ3D01
*SYSLG70 *OPLOG04 BOZOBOB  DSNTWR   ROGERS   PAOLOR3  VBUDI
TIVO01   TWSRES2  KMT2     TWSRES3  TWSRES5  *ROUTE70 VAIN
RONN     MSO6RV5  AUTXCF63 ARONN63  ASRES464 WHITE    AUTMON64
MERONI   AUTGSS64 *ROUTE65 SYSIOSRS HAIMO65  UNOB     STOECKE
PIPPO    DAV      *DICNS63 *DICNS64 HAIMO    *DICNS65 SYSJ3R01
BOZO     *DICNS70 BZZOSC64 BZZOSC70 PAOLOR1  PEGGYR   VBUDI64
PLUGH    GTMTBSM  TWSRES4  PAOLOR4  DONNAS   PAOLOR6  RC64
SYSJ3R02 AAUTO163 AUTSYS63 AUTREC63 SC64R64  AUTXCF64 AHW00264
PAOLOR8  SC64T64  MORON    PATRICK  XYZ      HORN     XCFCONS
GREGGL   HERICONS
```

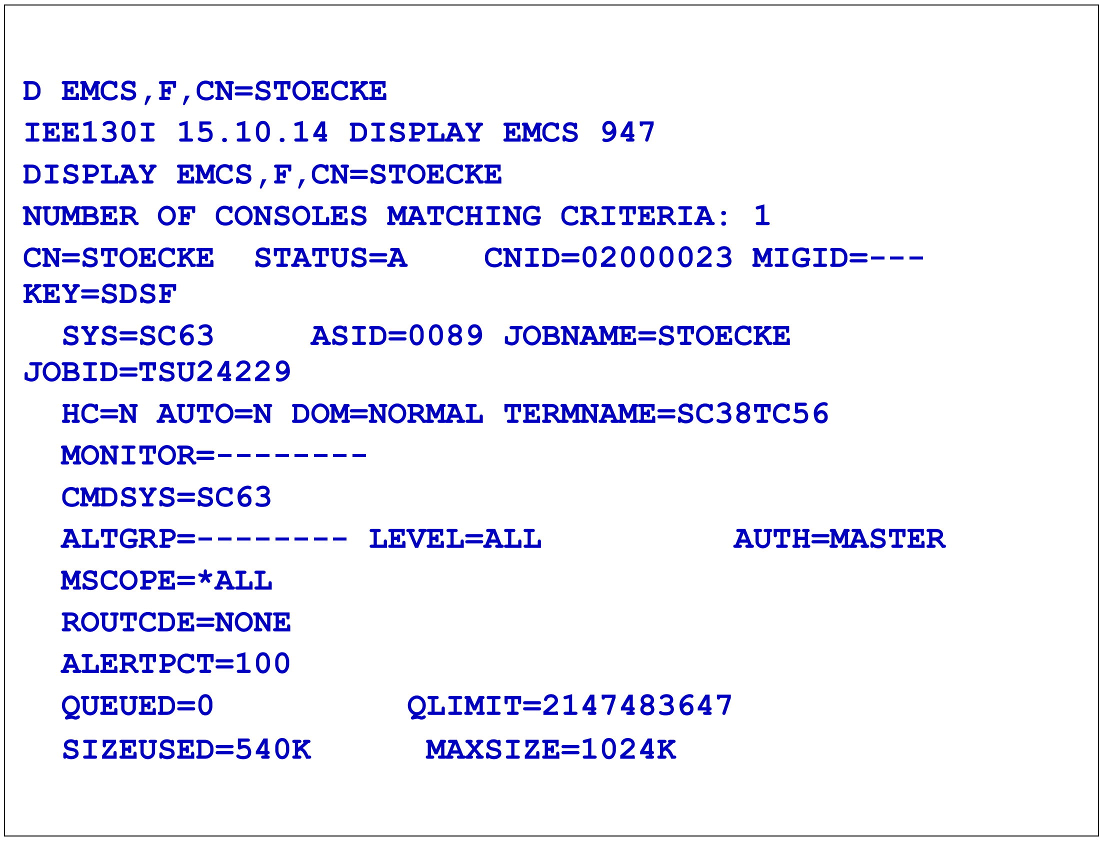
Figure 5-17 Display all defined EMCS

Display all defined EMCS

Use the **DISPLAY EMCS** command (instead of the **DISPLAY CONSOLES** command) to display information about extended MCS (EMCS) consoles. When the system searches for any consoles you specify, it allows wildcard matching. CN, SYS, and KEY can include wildcard characters (* and ?) that allow a single parameter to match many different actual conditions. For example, CN=AD? matches console names like AD1 or AD2 but not ADD1. CN=A* matches A1 or AD1 or ADD1.

With the **DISPLAY EMCS,S,STATUS=L** command you can display and determine the total number of EMCS consoles defined in the sysplex. STATUS=L displays both the active and inactive extended MCS consoles.

5.17 Display information about an EMCS



```
D EMCS,F,CN=STOECKE
IEE130I 15.10.14 DISPLAY EMCS 947
DISPLAY EMCS,F,CN=STOECKE
NUMBER OF CONSOLES MATCHING CRITERIA: 1
CN=STOECKE  STATUS=A    CNID=02000023 MIGID=---
KEY=SDSF
  SYS=SC63     ASID=0089 JOBNAME=STOECKE
JOBID=TSU24229
  HC=N AUTO=N DOM=NORMAL TERMNAME=SC38TC56
  MONITOR=--------
  CMDSYS=SC63
  ALTGRP=-------- LEVEL=ALL          AUTH=MASTER
  MSCOPE=*ALL
  ROUTCDE=NONE
  ALERTPCT=100
  QUEUED=0          QLIMIT=2147483647
  SIZEUSED=540K      MAXSIZE=1024K
```

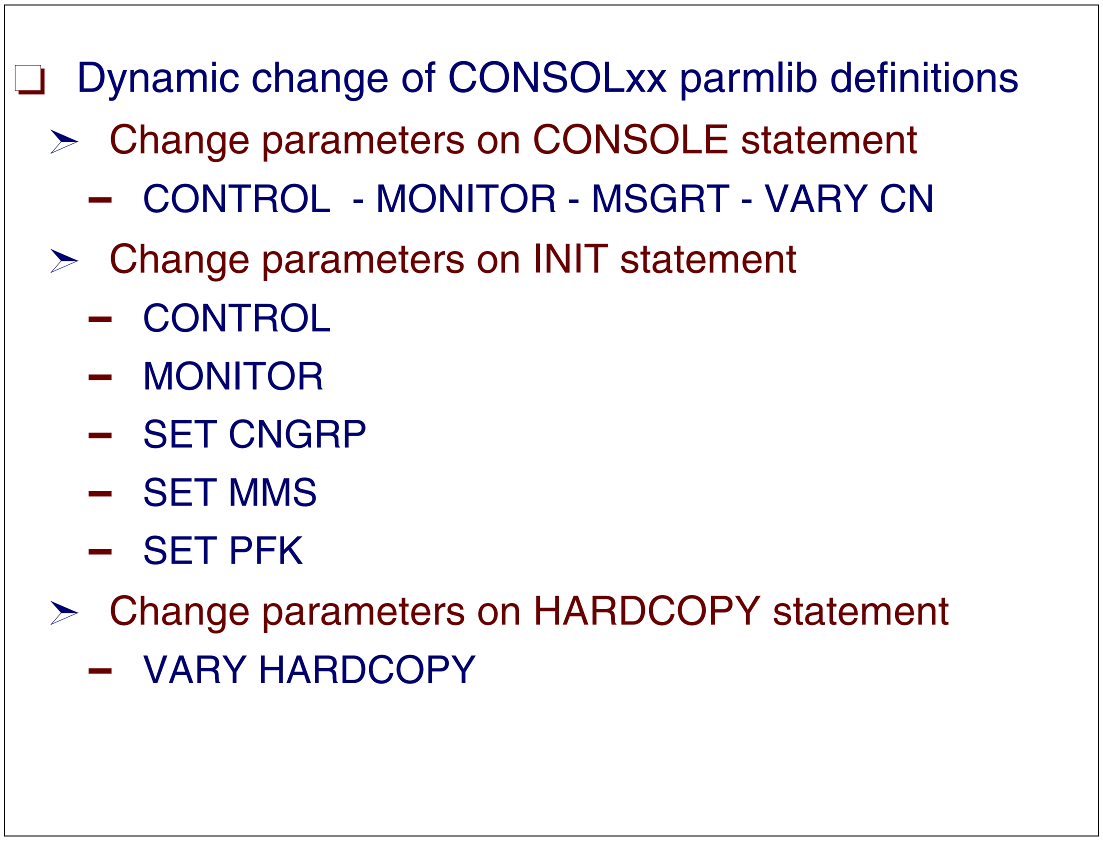
Figure 5-18 Display information about an EMCS

Display information about an EMCS

The `DISPLAY EMCS,FULL,CN=consolename` command displays all available information about the console selected with the CN criteria. The displayed fields have the following meanings:

CN=	Console name
STATUS=A	EMCS is in status active
CNID=	Console identifier
KEY=	Console key name assigned to a console group, which makes it possible to group consoles by function
SYS=	System the console is defined to - SC63 in this example
AUTO=N	No indicates that the console does not receive messages specified for automation through MPF
DOM=NORMAL	Indicates that the system directs all appropriate DOMs to the console
TERMNAME=	Terminal name
CMDSYS=	Commands processed on the system where the console is attached
LEVEL=ALL	All levels of messages sent to the console
AUTH=MASTER	Master authority for the console
MSCOPE=*ALL	Displays messages from all systems in the sysplex on the console
ROUTCODE=NONE	No routing codes established for the console

5.18 Defining and changing console characteristics



- Dynamic change of CONSOLxx parmlib definitions
  - Change parameters on CONSOLE statement
    - CONTROL - MONITOR - MSGRT - VARY CN
  - Change parameters on INIT statement
    - CONTROL
    - MONITOR
    - SET CNGRP
    - SET MMS
    - SET PFK
  - Change parameters on HARDCOPY statement
    - VARY HARDCOPY

Figure 5-19 Defining and changing console characteristics

Defining and changing console characteristics

The CONSOLxx parmlib member lets you define certain devices as consoles and specify attributes that determine how your operators can use MCS or SMCS consoles. CONSOLxx contains four statements that define and control consoles for an MVS system:

- CONSOLE
- INIT
- DEFAULT
- HARDCOPY

CONSOLE statement

The CONSOLE statement lets you define each console by device number on the CONSOLE statement. You must have a CONSOLE statement for each device that you want to use as a console. Use the Add Device panel in hardware configuration definition (HCD) to specify the device number and the unit type for MCS consoles only. SMCS consoles require the terminals to be defined for use via z/OS Communications Server. A CONSOLxx member can include multiple CONSOLE statements: one for each console in the configuration.

You use the CONSOLE statement to define a device as a console. You define each console device with one CONSOLE statement. CONSOLE also lets you specify console attributes that control the following for an MCS or SMCS console:

- Console security by assigning command authority levels
- Message routing and message formatting

- Certain console screen functions (console mode, methods for deleting messages from the screen, ways to control display areas on the screen, and how to set up the PFKs for the console)
- Console operation in a sysplex

INIT, DEFAULT, and HARDCOPY statements

The INIT, DEFAULT, and HARDCOPY statements define general characteristics for all MCS and SMCS consoles in the system or sysplex. The INIT statement is used to control basic initialization values for all MCS or SMCS consoles in the configuration.

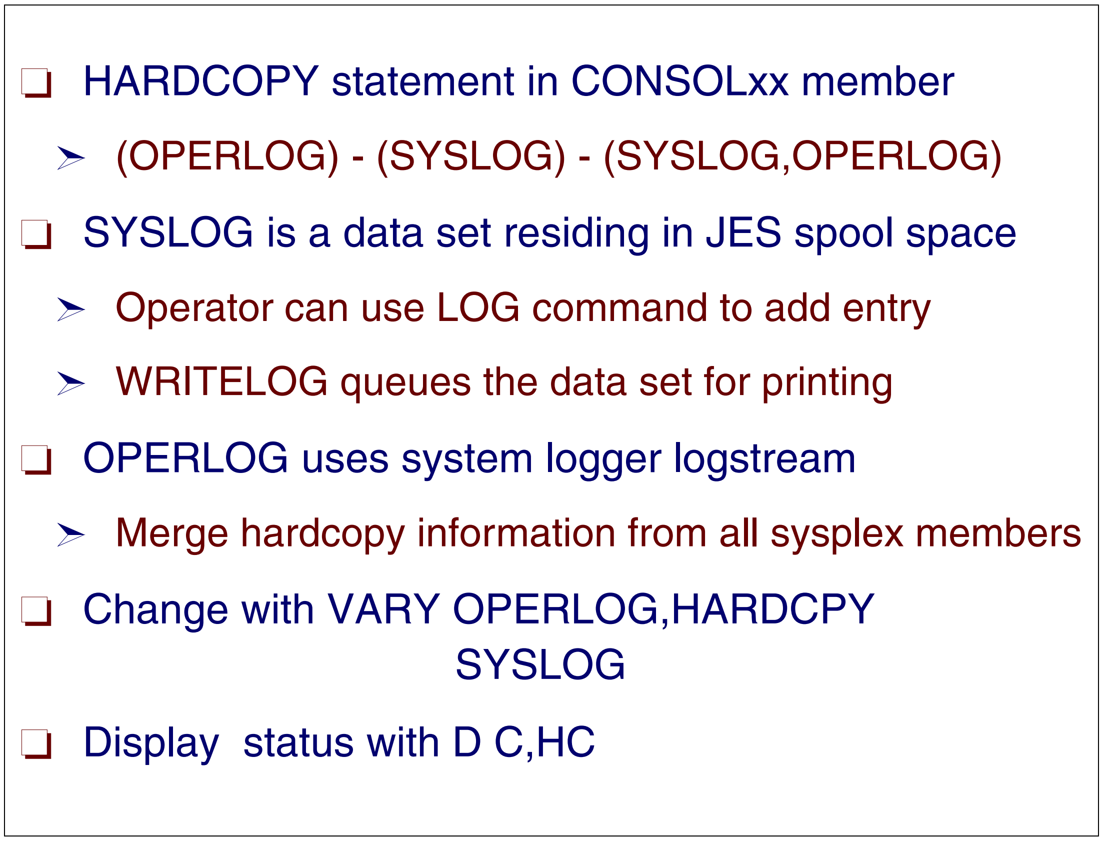
You use the DEFAULT statement to control certain default values for MCS and SMCS consoles in the configuration.

You can use the optional HARDCOPY statement to define the characteristics of the hardcopy message set and specify the hardcopy medium. You can control how to record messages and commands for the system. After IPL, operators can use the **VARY** command to do the following:

- Change the set of messages included in the hardcopy message set
- Assign SYSLOG or OPERLOG, or both, as the hardcopy medium

You can use several system commands to change the console characteristics dynamically. Each command has an equivalent statement in the CONSOLxx parmlib member. The changes are only temporary; for permanent changes the CONSOLxx member should be updated. This enables you to keep the definitions over the next IPL. For more details, refer to *z/OS System Commands,* SA22-7627. Chapter 3 shows all the details of the dynamic change capabilities.

5.19 The hardcopy medium



- HARDCOPY statement in CONSOLxx member
  - (OPERLOG) - (SYSLOG) - (SYSLOG,OPERLOG)
- SYSLOG is a data set residing in JES spool space
  - Operator can use LOG command to add entry
  - WRITELOG queues the data set for printing
- OPERLOG uses system logger logstream
  - Merge hardcopy information from all sysplex members
- Change with VARY OPERLOG,HARDCPY
  SYSLOG
- Display status with D C,HC

Figure 5-20 The hardcopy medium

The hardcopy medium

Hardcopy processing allows your installation to have a permanent record of system activity and helps you audit the use of operator commands. The group of messages and commands that are recorded is called the *hardcopy message set*. The system log, operations log, or MCS printer that receive messages are called the hardcopy medium.

HARDCOPY statement in the CONSOLxx member

You can specify whether the hardcopy medium is the system log (SYSLOG) or the operations log (OPERLOG) at system initialization using the HARDCOPY statement in the CONSOLxx member of parmlib.

SYSLOG is a data set residing in the JES spool space

The system log (SYSLOG) is a data set residing in the primary job entry subsystem’s (JES) spool space. It can be used by application and system programmers to record communications about problem programs and system functions. The operator can use the LOG command to add an entry to the system log. SYSLOG is queued for printing when the number of messages recorded reaches a threshold specified at system initialization. The operator can force the system log data set to be queued for printing before the threshold is reached by issuing the WRITELOG command.

OPERLOG uses the System Logger log stream

The operations log (OPERLOG) is a log stream that uses the System Logger to record and merge communications about programs and system functions from each system in a sysplex. Only the systems in a sysplex that have specified and activated the operations log will have their records sent to OPERLOG. For example, if a sysplex has three systems, SYS A, SYS B and SYS C, but only SYS A and SYS B activate the operations log, then only SYS A and SYS B will have their information recorded in the operations log.

Change with VARY HARDCPY

Once the system has been initialized, operators can use the VARY HARDCPY command to redefine the hardcopy medium.

Display status with D C,HC

Use the D C,HC commands to display the hardcopy medium status; see Figure 5-21.

```
D C,HC
IEE889I 15.33.13 CONSOLE DISPLAY 495
MSG: CURR=0    LIM=1500 RPLY:CURR=2    LIM=999  SYS=SC65      PFK=00
 CONSOLE           ID  --------------- SPECIFICATIONS ---------------
 SYSLOG                COND=H      AUTH=CMDS         NBUF=N/A
                       ROUTCDE=ALL
 OPERLOG               COND=H      AUTH=CMDS         NBUF=N/A
                       ROUTCDE=ALL
```

Figure 5-21 D C,HC command example

5.20 z/OS operations log (OPERLOG)

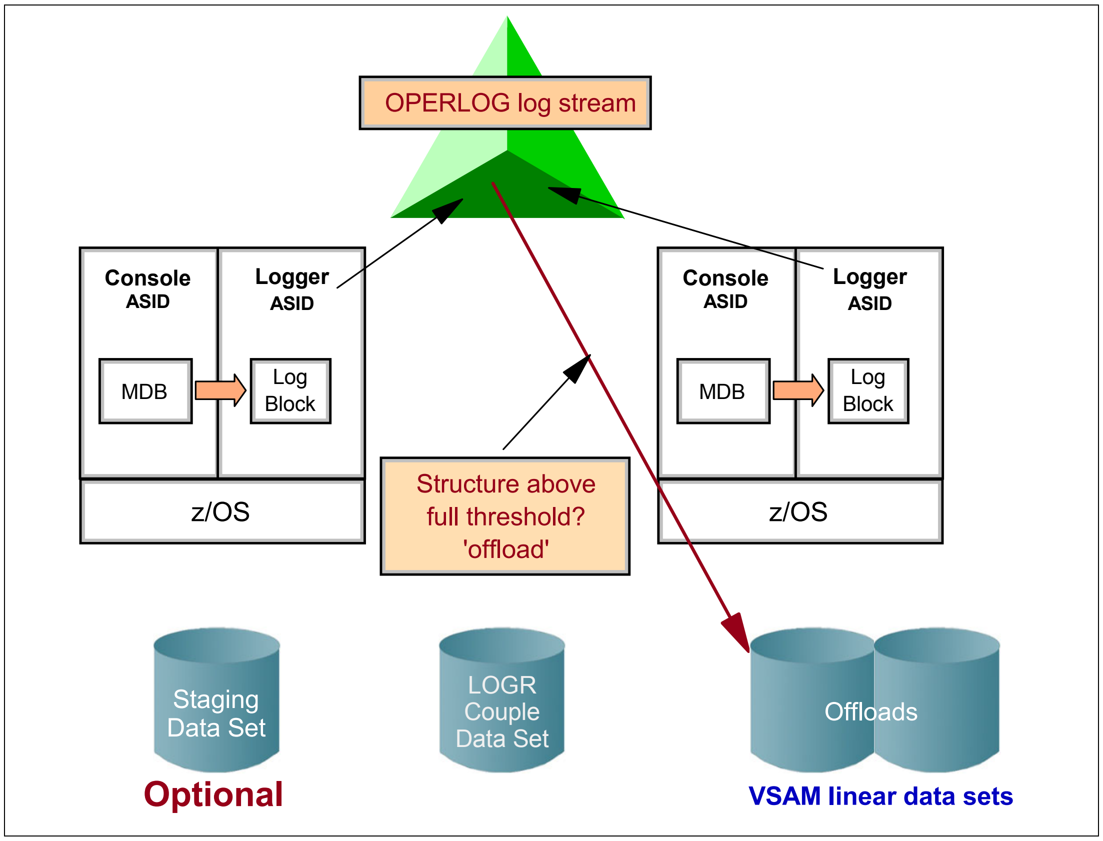


Figure 5-22 z/OS operations log (OPERLOG)

z/OS operations log (OPERLOG)

The OPERLOG uses System Logger services to record and merge communications about programs and system functions from each system in a sysplex, providing a sysplex-wide merged and chronologically ordered message log.

The messages are logged using message data blocks (MDB), which provide more data than is recorded in the SYSLOG. You can use member IEAMDBLG, in SYS1.SAMPLIB, to convert OPERLOG records into SYSLOG format.

Only the systems in the sysplex that have specified and activated the operations log will have their records sent to OPERLOG.

There is only one OPERLOG log stream within the entire sysplex, but the recording of the messages into the log stream can be activated on a system basis. You might have systems that need to merge in the OPERLOG log stream and other systems that can continue to record independently on their SYSLOG. Only the systems in a sysplex that have specified and activated the operations log have their records sent to OPERLOG.

Defining OPERLOG as a DASD-only log stream is suitable *only* for a *single system sysplex*, because a DASD-only log stream is single-sysplex in scope and you can only have one OPERLOG log stream per sysplex. This means that if you make OPERLOG a DASD-only log stream, only one system can access it.

Browse OPERLOG with the SDSF LOG option

You can browse OPERLOG or SYSLOG with the system display and search facility (SDSF) LOG option. The log you are browsing, OPERLOG or SYSLOG, is shown on the upper left corner of the log display. SDSF displays which media is currently browsing and you can switch from OPERLOG to SYSLOG by using the LOG OPERLOG or LOG SYSLOG command at any time.

If you are browsing the OPERLOG while structure rebuild takes place either due to a failure or to a maintenance procedure, the log stream data will be unavailable for the time it takes to rebuild the log stream in the alternate coupling facility. For the duration of this interval, you are notified that data is not available with an error message in your upper right corner.

Accessing OPERLOG log streams

SYS1.SAMPLIB contains the sample program IEAMDBLG to read log blocks from the OPERLOG log stream and convert them to SYSLOG format. The program is an example of how to use the services of the System Logger to retrieve and delete records from the OPERLOG stream.

IEAMDBLG reads the records created in a given time span, converts them from Message Data Block (MDB) format to Hard-copy Log format (HCL or JES2 SYSLOG), and writes the SYSLOG-format records to a file.

IEAMDBLG also has an option to delete from the log stream all the records created prior to a given date. When you use the delete option, a suggestion is to first copy the records to an alternate media and then conditionally delete them in a separate JCL step. This ensures that you have a copy of the data before deleting. If you do not execute the commands in two separate conditional steps, deletion occurs simultaneously with copy without any guarantee that the copy process was successful.

Note: More details about setting up and using OPERLOG are in:

- *z/OS MVS Planning: Operations,* SA22-7601
- *z/OS MVS Setting Up a Sysplex,* SA22-7625
- *Systems Programmer's Guide to z/OS System Logger*, SG24-6898

6

Automatic restart management

Automatic restart management (ARM) is key to automating the restarting of subsystems and applications (referred to collectively as applications) so they can recover work they were doing at the time of an application or system failure and release resources, such as locks, that they were holding. With the automatic restart management policy, you can optionally control the way restarts are done. You can use the policy defaults that IBM provides, or you can override them.

In a sysplex environment, a program can enhance its recovery potential by registering as an element of automatic restart management. Automatic restart management can reduce the impact of an unexpected error to an element because z/OS can restart it automatically, without operator intervention. In general, z/OS restarts an element when:

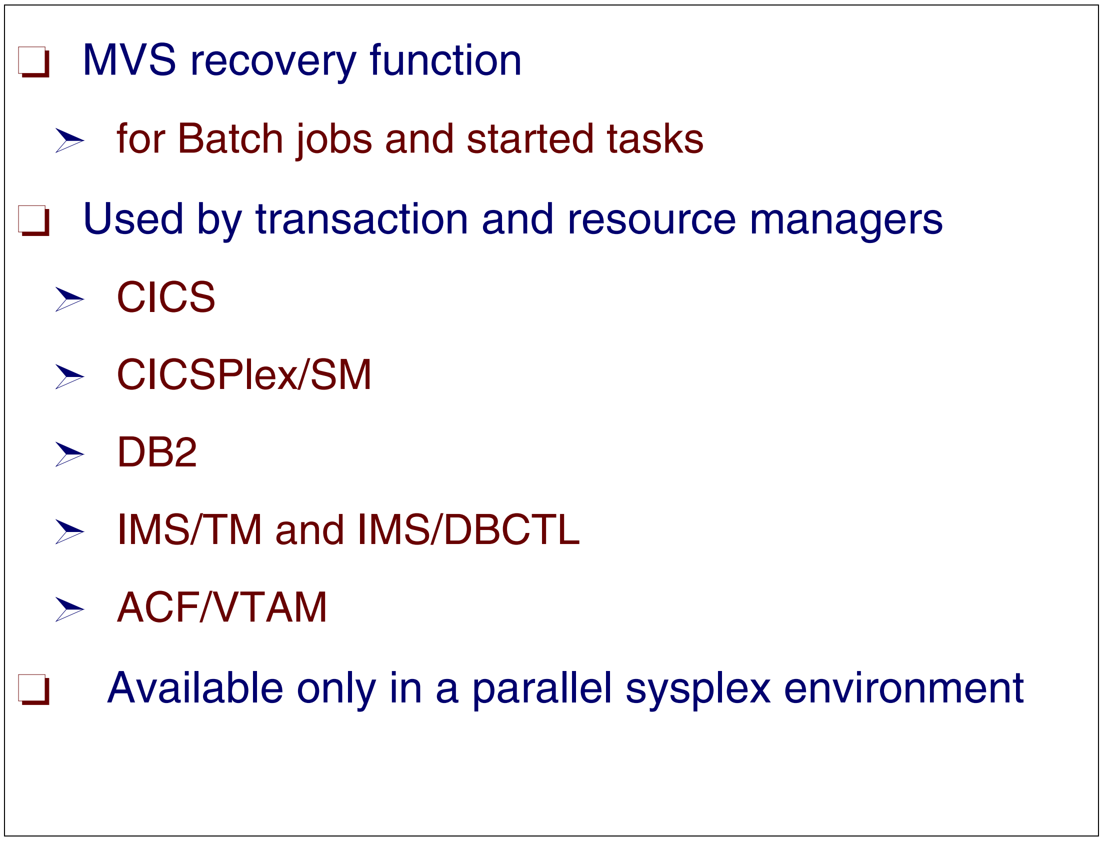
- The element itself fails. In this case, z/OS restarts the element on the same system.
- The system on which the element was running unexpectedly fails or leaves the sysplex. In this case, z/OS restarts the element on another system in the sysplex; this is called a cross-system restart.

In general, your installation can use automatic restart management in two ways:

- To control the restarts of applications (such as CICS) that already use automatic restart management as part of their recovery.
- To write or modify installation applications to use automatic restart management as part of recovery.

During planning before installation of the z/OS systems in the sysplex, you can define a sysplex failure management (SFM) policy, and an automatic restart management policy. The goals of SFM and automatic restart management are complementary. SFM keeps the sysplex up and running; automatic restart management keeps specific work in the sysplex (batch jobs or started tasks) up and running. SFM manages system-level problems and partitions systems out of the sysplex in response to those problems. Automatic restart management manages subsystems and restarts them appropriately after either subsystem-level or system-level failures occur.

6.1 Automatic restart management



- MVS recovery function
  - for Batch jobs and started tasks
- Used by transaction and resource managers
  - CICS
  - CICSPlex/SM
  - DB2
  - IMS/TM and IMS/DBCTL
  - ACF/VTAM
- Available only in a parallel sysplex environment

Figure 6-1 Automatic restart management

z/OS recovery function

Automatic restart management (ARM) is an z/OS recovery function that can improve the availability of specific batch jobs or started tasks. When a job or task fails, or the system on which it is running fails, ARM can restart the job or task without operator intervention.

The goals of sysplex failure management (SFM) and ARM are complementary. While SFM keeps the sysplex running, ARM keeps specific work in the sysplex running. If a job or task fails, ARM restarts it on the same system it was running at the time of failure. If a system fails, ARM restarts the work on other systems in the sysplex; this is called a cross-system restart.

A program cannot use both ARM and checkpoint/restart. If a program using checkpoint/restart tries to register with ARM services, the request is rejected.

The purpose of ARM is to provide fast, efficient restarts for critical applications when they fail. ARM is an z/OS recovery function that improves the time required to restart an application by automatically restarting the batch job or started task (STC) when it unexpectedly terminates. These unexpected outages may be the result of an abend, system failure, or the removal of a system from the sysplex.

A primary availability requirement for all system environments is to reduce the duration and impact of an outage. The sysplex environment helps to achieve this objective through the following recovery strategies:

- Detection and isolation of failures - detect a failure and route work around it.

- Recovery of resources - release critical resources as soon as possible so that other programs and systems can acquire them. This reduces contention and speeds up the recovery or restart of work affected by the failure.
- Recover lost capacity - restart or resume processing that was affected by the failure. For example, in a database environment, the DB manager should be restarted quickly so that its log recovery can be processed.

Used by transaction and resource managers

The intended exploiters of the ARM function are the jobs and STCs of certain strategic transaction and resource managers. Some of these are:

- CICS/ESA
- CP/SM
- DB2
- IMS/TM
- IMS/DBCTL
- ACF/VTAM

In general, an installation can use automatic restart management in two ways:

- To control the restarts of applications (such as CICS) that already use automatic restart management as part of their recovery
- To write or modify installation applications to use automatic restart management as part of recovery

For more information about automatic restart management, see *z/OS MVS Setting Up a Sysplex,* SA22-7625, *z/OS MVS Sysplex Services Guide*, SA22-7617, and *z/OS Parallel Sysplex Services Reference*, SA22-7618.

6.2 ARM environment

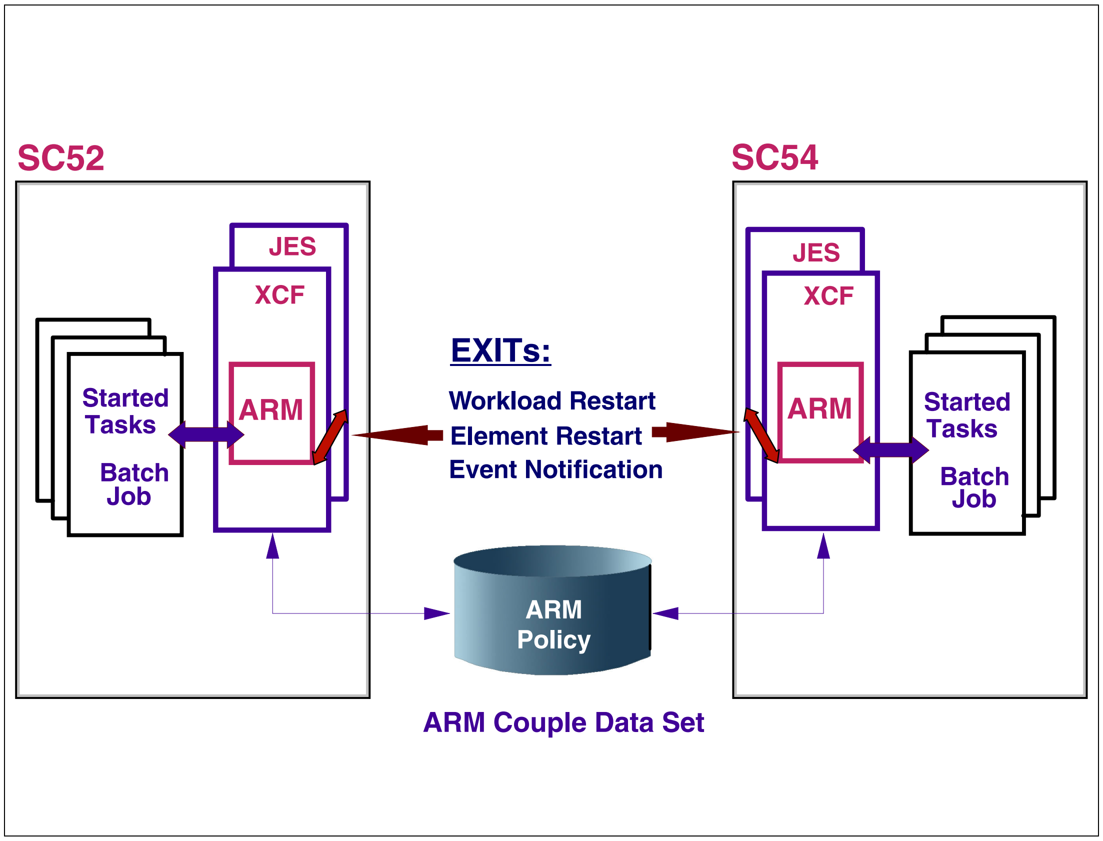


Figure 6-2 ARM environment

ARM environment

In a sysplex environment, a program can enhance its recovery potential by registering as an element of automatic restart management. An automatic restart management element represents a program or an application that:

- Is submitted as a job.
- Is submitted as a started task.
- Is an abstract resource. An abstract resource is a program or a set of programs that is only associated with (or has a bind to) the system on which it is running.

Automatic restart management can reduce the impact of an unexpected error to an element because MVS can restart it automatically, without operator intervention. In general, MVS restarts an element when:

- The element itself fails. In this case, MVS restarts the element on the same system.
- The system on which the element was running unexpectedly fails or leaves the sysplex. In this case, MVS restarts the element on another system in the sysplex; this is called a cross-system restart.

ARM policy

In addition to the policy, three exit points are provided where subsystem and installation-written exit routines can be invoked to cancel a restart or influence how it is done.

This gives you extended control and can simplify or enhance the policy that you build and activate. The exits are:

- The workload restart exit (IXC_WORK_RESTART)
- The element restart exit (IXC_ELEM_RESTART)
- The event exit

A system must be connected to an ARM couple data set with an active automatic restart management policy.

ARM couple data set

Automatic restart management allows different levels or functional versions of ARM couple data sets to be defined. Automatic restart management is a function that runs in the cross-system coupling facility (XCF) address space and maintains its own data spaces. ARM requires a primary couple data set to contain policy information as well as status information for registered elements. It supports both JES2 and JES3 environments.

The following couple data set format levels or functional versions can be created using the IXCL1DSU utility, as follows:

- Base format level. This is the initial or base ARM couple data set format level and is created when the ARM couple data set is formatted using a version of IXCL1DSU.
- This format level is created when the ARM couple data set is formatted using a version of IXCL1DSU.

Note: Automatic restarts do not need to be enabled at all times. For example, you might not want automatic restart management enabled for restarts unless certain jobs are running, or during off-shift hours. However, even while automatic restarts are disabled, elements can register with automatic restart management as long as the system is connected to an ARM couple data set.

6.3 Create an ARM couple data set and policy

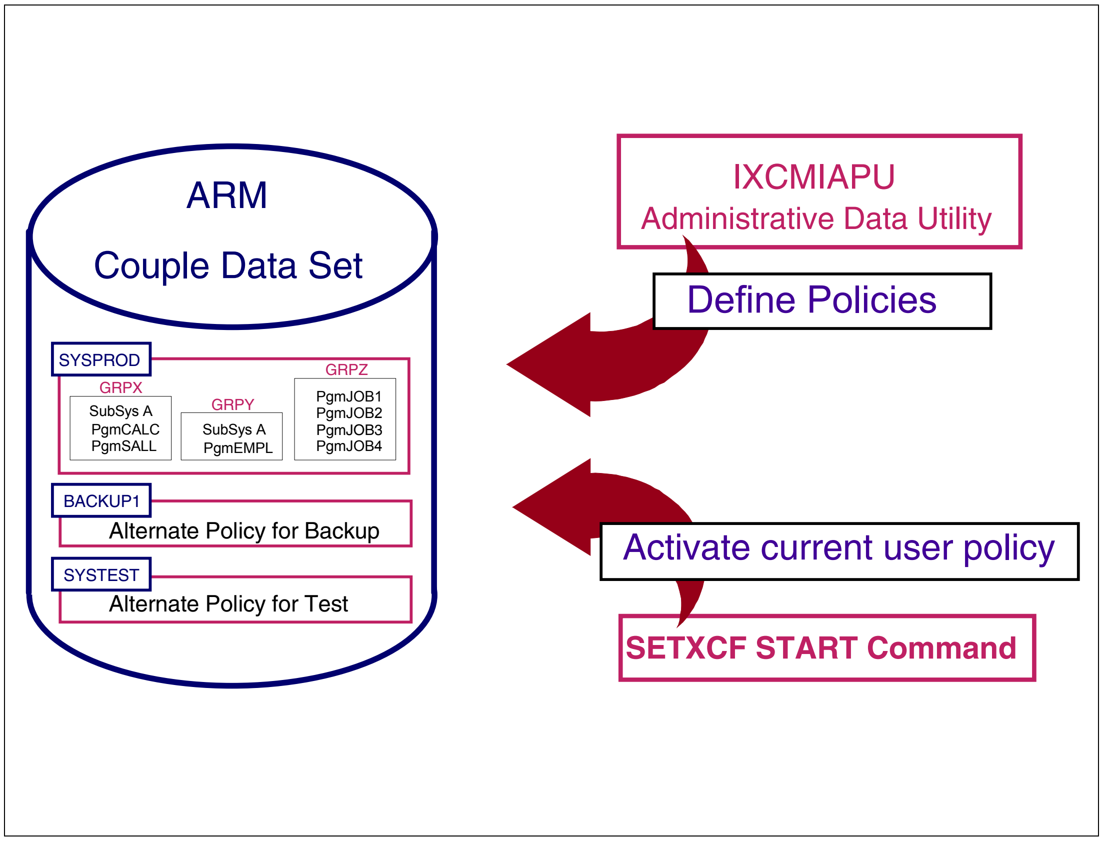


Figure 6-3 Create ARM couple data set and policy

Create ARM couple data set and policy

Figure 6-3 shows the key activities necessary to create an ARM couple data set and an ARM policy:

- Create a primary and alternate couple data set:
  Create and submit a ARM couple data set formatting job (IXCL1DSU).
- Add the primary couple data set to the sysplex:
  Use the command **`SETXCF COUPLE,TYPE=ARM,PCOUPLE=(SYS1.XCF.ARM10)`**
- Add the alternate couple data set to the sysplex:
  Use the command **`SETXCF COUPLE,TYPE=ARM,ACOUPLE=(SYS1.XCF.ARM20)`**
- Don’t forget to update the COUPLExx parmlib member with the ARM CDS names:

```
COUPLE SYSPLEX(PLEX1)
       CFRMPOL(POL5)
       PCOUPLE(SYS1.XCF.CDS04,VS4PL1)
       ACOUPLE(SYS1.XCF.CDS05,VS4PL2)
........
DATA   TYPE(ARM)
       PCOUPLE(SYS1.XCF.ARM10,VS4PL1)
       ACOUPLE(SYS1.XCF.ARM20,VS4PL3)
......
```

Figure 6-4 COUPLExx parmlib member example

JCL for ARM couple data sets

There is a sample job in SYS1.SAMPLIB(IXCARMF) to format a primary and alternate couple data set for automatic restart manager. In this sample JCL, two couple data sets, SYS1.MIGLIB.ARMCPL01 and SYS1.MIGLIB.ARMCPL02, will be allocated. If the size of these couple data sets is different, the larger of the two should always be used as an alternate couple data set.

Define an ARM policy

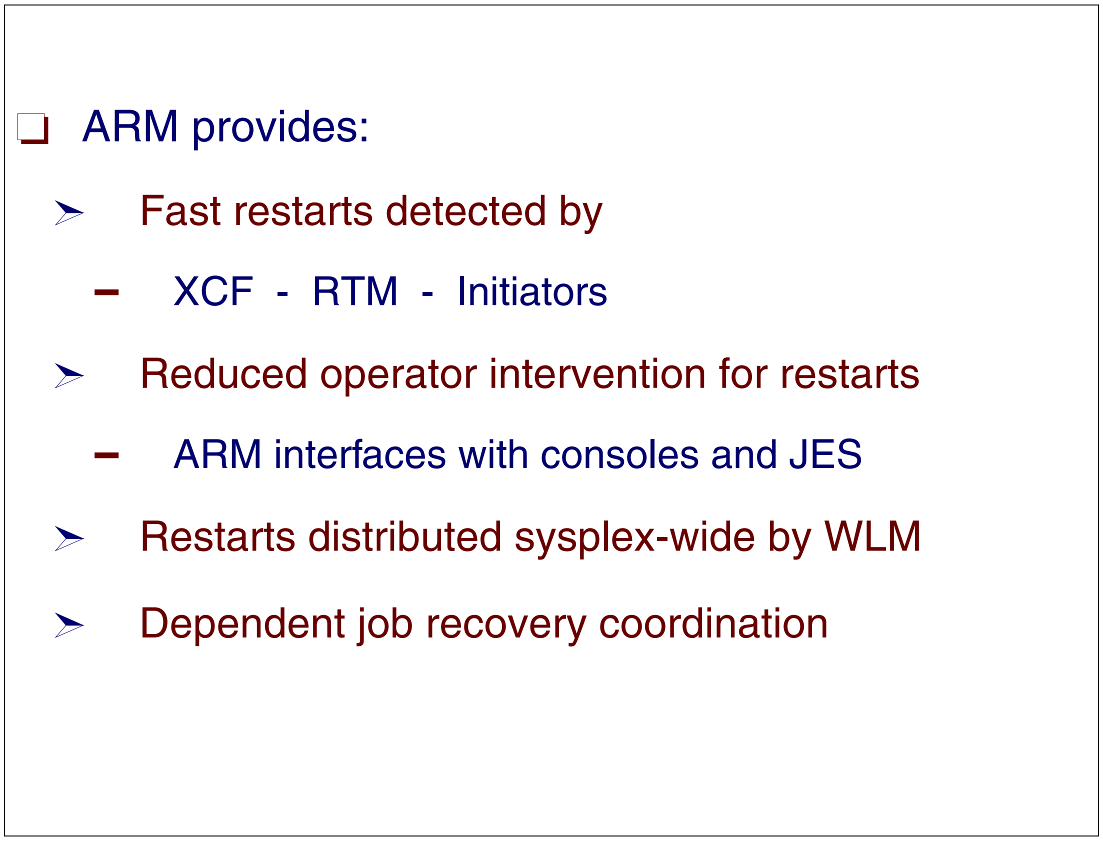
The ARM policy is defined by using the IXCARM macro, and is used as follows:

- If started tasks or jobs need a special restarting policy, then an ARM policy should be defined.
- Start the defined policy with this command:

  ```
  SETXCF START,POLICY,TYPE=ARM,POLNAME=name
  ```

The policy can be user-written or the IBM-supplied default policy.

6.4 ARM restarts



- ARM provides:
  - Fast restarts detected by
    - XCF - RTM - Initiators
  - Reduced operator intervention for restarts
    - ARM interfaces with consoles and JES
  - Restarts distributed sysplex-wide by WLM
  - Dependent job recovery coordination

Figure 6-5 ARM restarts

ARM restarts

The purpose of automatic restart management is to provide fast, efficient restarts for critical applications when they fail. ARM is a z/OS recovery function that improves the time required to restart an application by automatically restarting the batch job or started task (STC) when it unexpectedly terminates. These unexpected outages may be the result of an abend, system failure, or the removal of a system from the sysplex.

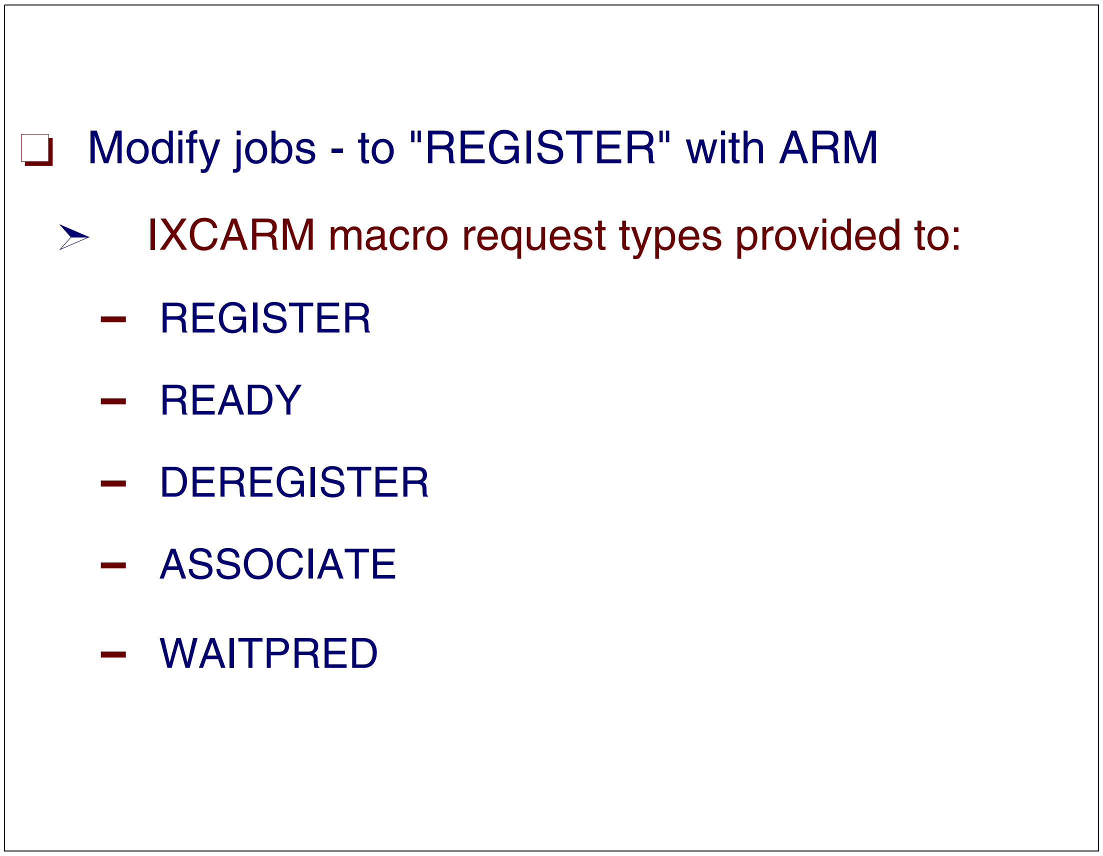
The need for restarts is detected by either XCF, recovery termination management (RTM), or the initiator.

ARM interfaces with MCS console support and JES to provide its recovery mechanisms.

Workload Manager (WLM) provides statistics on the remaining processing capacity in the sysplex.

ARM minimizes failure impact by doing fast restarts without operator intervention in the process. Related groups of programs can be restarted and if programs are start-order dependent, it assures that the order dependency is honored. The chance of work being restarted on systems lacking the necessary capacity is reduced as the best system is chosen, based on the statistics returned to ARM by the Workload Manager.

6.5 Modifying batch and STC jobs



- Modify jobs - to "REGISTER" with ARM
  - IXCARM macro request types provided to:
    - REGISTER
    - READY
    - DEREGISTER
    - ASSOCIATE
    - WAITPRED

Figure 6-6 Modifying batch and STC jobs

Modifying batch and STC jobs

To use ARM, jobs must be modified to register with ARM using the IXCARM macro services. In a sysplex environment, a program can enhance its recovery potential by registering as an *element* of automatic restart management. Automatic restart management can reduce the impact of an unexpected error to an element because z/OS can restart it automatically, without operator intervention. The automatic restart management service routine is given control from the IXCARM macro and is used to:

REGISTER	Register a user of automatic restart management services.
READY	Mark a user of automatic restart management services as ready to accept work.
DEREGISTER	Deregister a user of automatic restart management services.
ASSOCIATE	Associate a user of automatic restart management services with another user for takeover or restart purposes. This allows a program to identify itself as the backup program for another element of the automatic restart management. This identification tells z/OS that the other element should not be restarted unless this backup program is deregistered.
WAITPRED	Wait until predecessor elements have been restarted if applicable. This indicates that z/OS should delay the restart for this program until z/OS completes the restart of a related program, which is called a predecessor element.

6.6 Registering with ARM

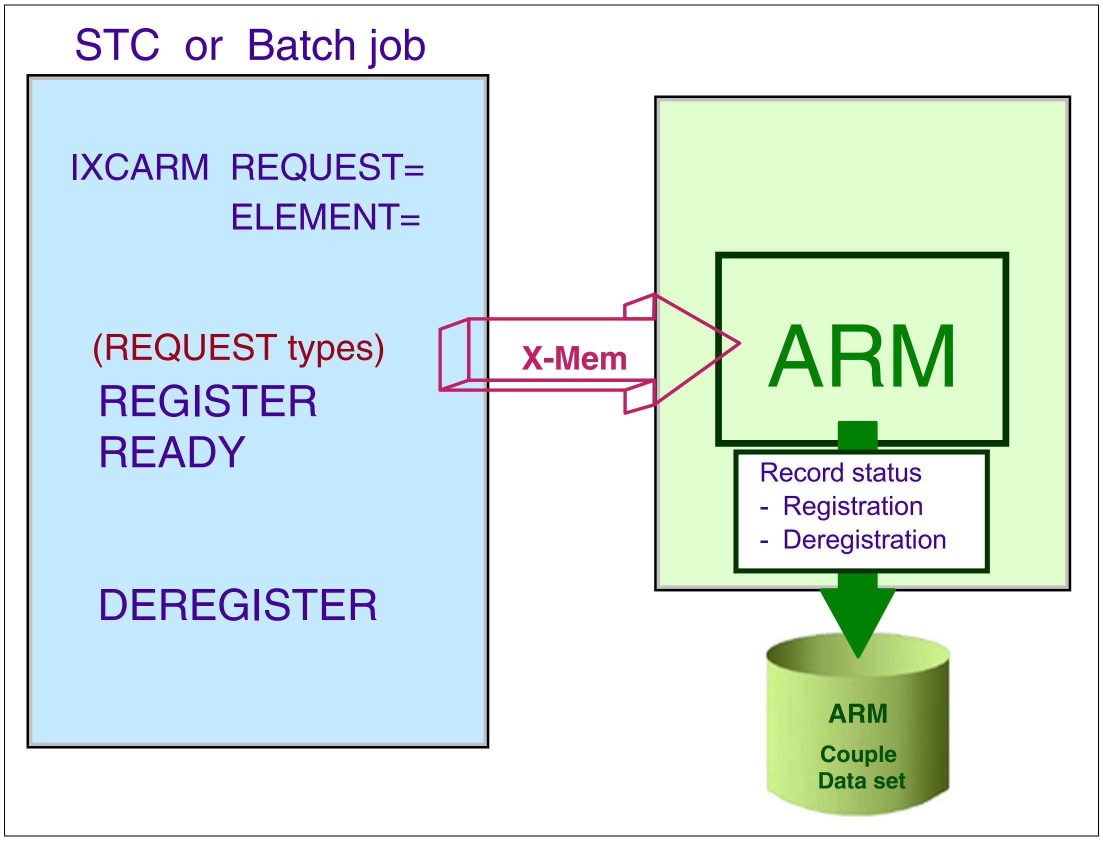


Figure 6-7 Registering with ARM

Registering with ARM

To make batch jobs or started tasks ARM restartable, they must register with ARM using an authorized macro service, IXCARM. The registration requires the specification of an *element* name. This element name must be unique in the sysplex and is up to sixteen characters long. The three basic ARM services that a job must use are as follows:

REGISTER Early in the initialization process for a job, a program that wants to use ARM for restart must issue the IXCARM REQUEST=REGISTER macro. Part of the macro parameters is the element name.
Optionally, you can specify restart parameters and an event exit.
After the job or task has issued the REGISTER, z/OS can automatically restart the program when an unexpected failure occurs. You can specify restart parameters on the REGISTER request; however, restart parameters in an user written ARM policy override IXCARM macro definitions.

READY When a program that has issued the REGISTER request has completed initialization and is ready to run, an IXCARM REQUEST=READY must be issued.

DEREGISTER Before a job completes its normal termination processing, the program must issue an IXCARM REQUEST=DEREGISTER to remove the element and all ARM restart possibilities.
You can deregister from automatic restart management when the job or task no longer needs to be restarted. If a program fails after it deregisters, z/OS will not restart the program.

6.7 ARM element states

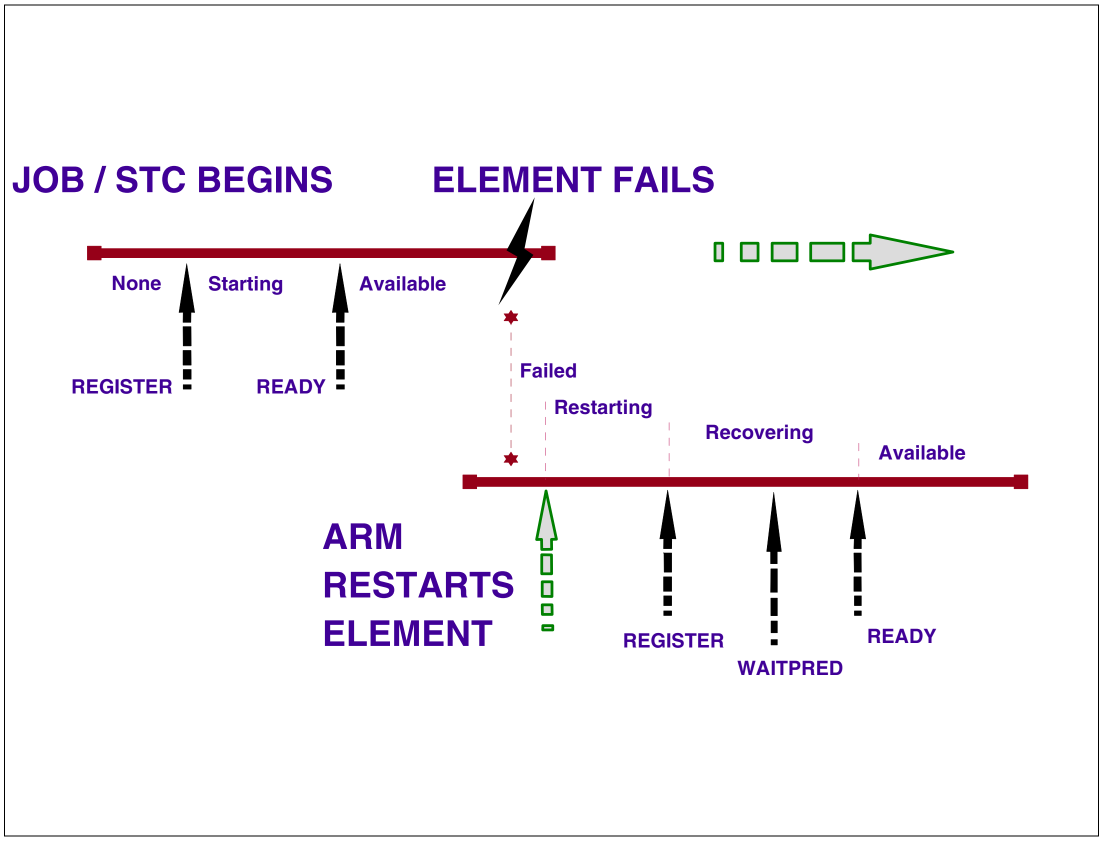


Figure 6-8 ARM element states

ARM element states

Figure 6-8 shows an element's use of ARM services and how they affect the state of that element.

ARM puts each element into one of following states to identify the last interaction:

Starting	From the initial registration (IXCARM-Register) of a program as an ARM element until ready (IXCARM-Ready).
Available	From the time the element becomes ready (IXCARM-Ready) until the element either deregisters from ARM or terminates.
Failed	From ARM detected termination of an element or termination of the system, until ARM initiates a restart of the element.
Restarting	From ARM's initiation of a restart until the subsequent reregistration of the element.
Recovering	From the element's registration after an ARM restart to its subsequent issuing of IXCARM-Ready.
WAITPRED	By issuing IXCARM with the WAITPRED parameter, an element indicates that a predecessor element must become ready before this element can initialize successfully. During restarts, not initial starts, MVS will *wait for* the predecessor to issue IXCARM REQUEST=READY before allowing this element to complete ready processing. Issuing WAITPRED is most useful when an element and its predecessor are in the same restart group, by

specific assignment or by default. Elements should issue WAITPRED after the register request, but before the ready request.

Element processing

Initially, an element is unknown to ARM. The first interaction with ARM occurs when the program makes itself known to ARM by invoking the register service and specifying the element name under which it is to be registered. ARM sets the state of the element to starting.

When the element is ready to accept new work, it invokes the ready service, and ARM sets the state of the element to *available*. Once the element's state is available, there are no more interactions with ARM until the element terminates. Before terminating normally, the element invokes the deregister service, which again makes the element unknown to ARM.

Element termination

When ARM detects that the element has unexpectedly terminated, that is, the program has terminated without invoking the deregister service, or that the system on which the element had been running has left the sysplex, ARM sets the element's state to *failed* as part of its processing to restart the element. An element will be in the failed state for a very brief interval.

Element restart

When ARM restarts the element, it sets the state to *restarting*. When a restarting element subsequently invokes the register service, ARM sets the state to *recovering*. Once the element is prepared to accept new work, it notifies ARM by invoking the ready service, which causes ARM to set the element's state to *available*.

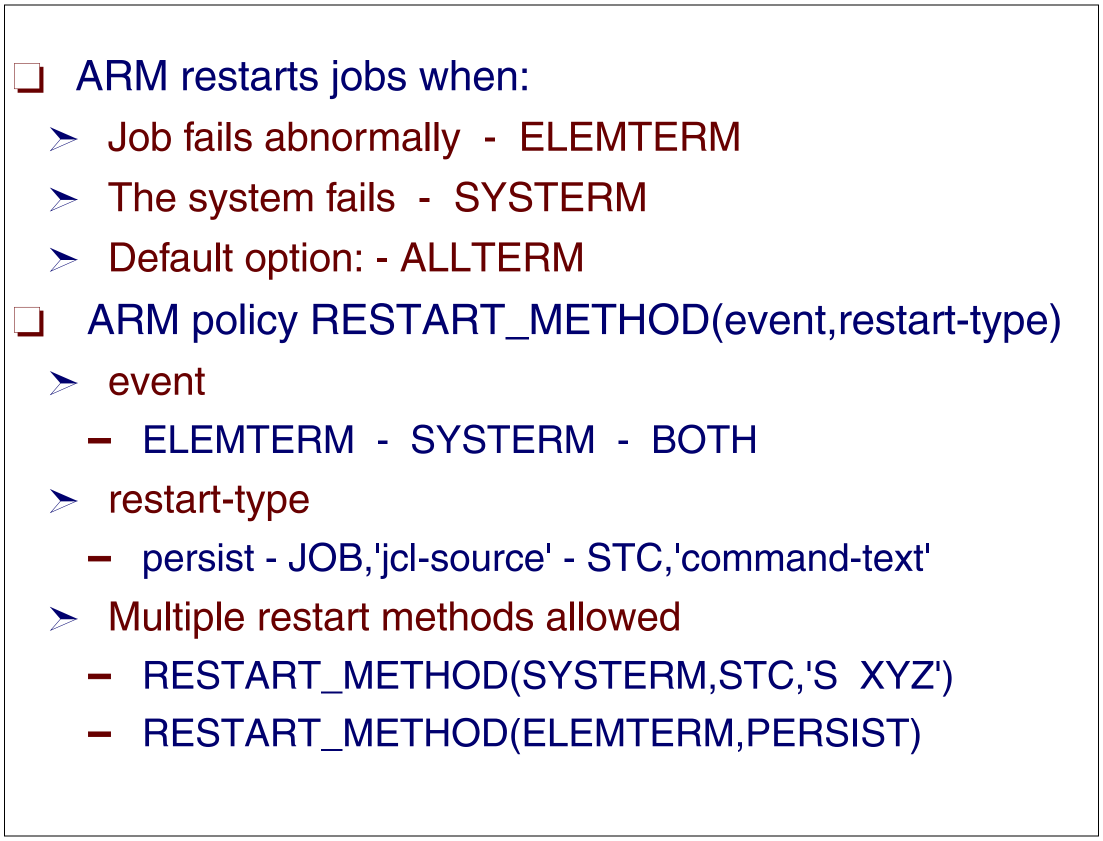
Command for ARM status

The state of a given element named DB2$DB2H will be part of the information provided when ARM status is requested with the **DISPLAY XCF,ARMSTATUS** command for one or more elements.

```
D XCF,ARMS,EL=DB2$DB2H
IXC392I  17.03.26  DISPLAY XCF 602
ARM RESTARTS ARE NOT ENABLED
-------------- ELEMENT STATE SUMMARY --------------   -TOTAL-  -MAX-
STARTING  AVAILABLE  FAILED  RESTARTING  RECOVERING
       0          1       0           0           0        1    200
RESTART GROUP:DEFAULT          PACING  :    0    FREECSA:    0       0
 ELEMENT NAME     STATE        CURR SYS INIT SYS JOBNAME  ASID   LEVEL
 DB2$DB2H         AVAILABLE    SC63     SC63     DB2HMSTR 006B       2
```

Figure 6-9 Example of DISPLAY XCF,ARMSTATUS output

6.8 ARM restart methods



- ❑ ARM restarts jobs when:
  - ➢ Job fails abnormally - ELEMTERM
  - ➢ The system fails - SYSTERM
  - ➢ Default option: - ALLTERM
- ❑ ARM policy RESTART_METHOD(event,restart-type)
  - ➢ event
    - – ELEMTERM - SYSTERM - BOTH
  - ➢ restart-type
    - – persist - JOB,'jcl-source' - STC,'command-text'
  - ➢ Multiple restart methods allowed
    - – RESTART_METHOD(SYSTERM,STC,'S XYZ')
    - – RESTART_METHOD(ELEMTERM,PERSIST)

Figure 6-10 ARM restart methods

ARM restart methods

ARM specifies under which conditions MVS should restart this element. ARM has three restart types, ALLTERM, ELEMTERM, and SYSTERM, as follows:

ALLTERM Indicates that the element should be restarted for all unexpected failures as appropriate. The value specified on an IXCARM REQUEST=REGISTER request for the ELEMBIND keyword determines what types of failures are appropriate.

ELEMTERM Element termination
When an element unexpectedly fails, ARM is given control during end-of-job and end-of-memory termination after all other recovery has taken place. If the job or task terminating is an element of the automatic restart management and should be restarted, then z/OS:

a. Gives control to the element-restart exit.
b. Gives control to the event exit.
c. Restarts the element.
d. Issues an ENF signal when the element re-registers with the automatic restart management.

SYSTERM System termination
When a system unexpectedly fails, ARM determines whether any elements were running on this system. If those elements can be restarted on another

system, and cross-system restarts are allowed, z/OS does the following for each system on which the elements will be restarted:

a. Gives control to the workload restart exit.
b. For each element that will be restarted:
   i. Gives control to the element restart exit.
   ii. Gives control to the event exit.
   iii. Restarts the element.
   iv. Issues an ENF signal when the element reregisters with the automatic restart management.

RESTART_METHOD in ARM policy

The ARM policy contains entries for elements that define what type of restart will be performed for the element. In the ARM policy, the RESTART_METHOD(event,restart-type) is optional. The default is RESTART_METHOD(BOTH,PERSIST). For started tasks only, the IXCARM macro parameter that overrides the default specification is STARTTXT.

The RESTART_METHOD has three *event* options:

ELEMTERM	Indicates that the persistent JCL or command text is to be overridden by the JCL data set or the command text specified in restart-type only when the element itself terminates.
SYSTERM	Indicates that the persistent JCL or command text is to be overridden by the JCL data set or the command text specified in restart-type only when the system the element was running on terminates.
BOTH	Indicates that the persistent JCL or command text is to be overridden by the JCL data set or the command text specified in restart-type when either the system the element was running on terminates, or when the element itself terminates.

Three restart-type options:

STC,'command-text'	Tells ARM to restart this element using the command provided with 'command-text'.
PERSIST	Indicates that z/OS is to use the JCL or command text that previously started the element.
JOB,'jcl-source'	Indicates that z/OS is to restart the element as a batch job. 'jcl-source' is the name of the data set that contains the JCL for restarting the element. This data set name must be enclosed within single quotes. The data set must have the same data set characteristics (for instance, LRECL) as standard procedure libraries.

Restarting an element

When ARM restarts an element that terminated with an ELEMTERM, that element is restarted on the same system where it was previously active. Even if the initiators of the system all are busy or stopped, JES tries to restart the job on that system. This is done by setting the system affinity of the restarting job to the same system.

A job that is being ARM-restarted when it fails with an ELEMTERM termination can never execute on a different system. This is because the job would try to register with ARM using the same element name on a different system than the restart system. This is not allowed, and the following return code and reason code are issued by ARM: (RC0008, RSN0150 of IXCARM REQUEST=REGISTER).

6.9 Restart on the same system

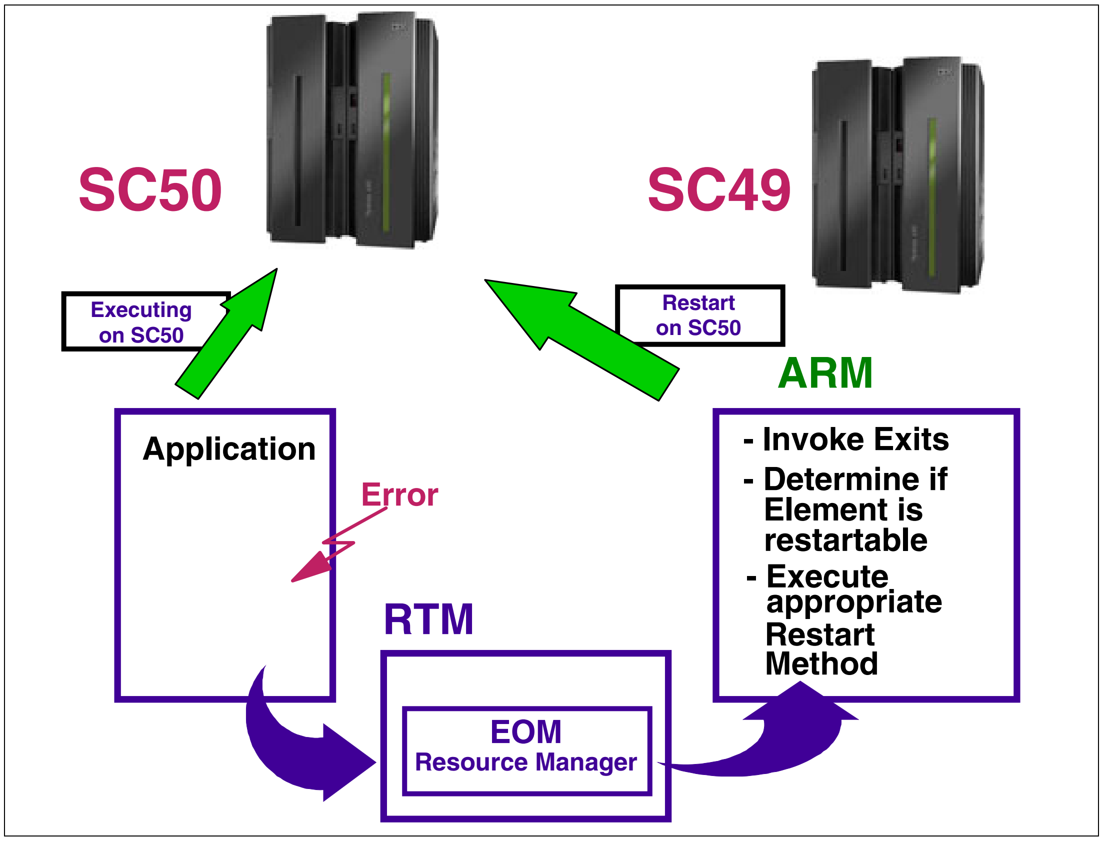


Figure 6-11 Restart on the same system

Restart on the same system

When an element ABENDS or otherwise terminates while registered with ARM, ARM restarts that element on the same system on which it had been running. Such ARM restarts are referred to as *restarts in place*.

You should be aware that ARM restart processing does not include any kind of cleanup or recovery. That is considered internal to the element being restarted. Element-related processing, such as the releasing of *local* resources or the analysis of the impact of the termination to on-going work, is the responsibility of recovery routines of the element. Figure 6-11 shows the various factors involved in restarting work on the same system.

ARM EOM processing

ARM's end-of-memory (EOM) resource manager determines if a terminating address space is a registered element. If so, it schedules the restarting of that element. The elements restarted from ARM's EOM resource manager are started tasks. Restarts for batch jobs are performed in essentially the same way from the ARM end-of-job (EOJ) resource manager.

ARM restart

If the element to be restarted specified an event exit routine when it registered as an ARM element, then ARM calls the specified exit routine before restarting that element. This processing could instruct ARM *not* to restart the element. If there are any element restart exits defined, ARM also invokes them. An element restart exit routine can cancel the restart of a given element or change the method by which ARM restarts it.

The method by which ARM restarts an element depends on whether the element is a started task or a batch job. You can specify the restart method through either the ARM policy or an exit routine. A registering program can also specify a command (through the REGISTER macro) to be used for ARM restarts.

ARM exit routine for restarts

If an installation policy or an exit routine provides an override JCL data set or command text to restart an element, ARM determines if the data set name or command text contains any symbolic substitution parameters. If it does, these parameters are resolved before the command is issued or the JCL submitted. This resolution of symbolic substitution parameters is done using the values from the system on which the element initially registered. For restarts in place, ARM's replacement of these values provides the same result as would occur if the normal z/OS resolution had been done.

After initiating the restart, ARM loses contact with the element and cannot track how the restart is processing. ARM establishes a time interval to wait for the subsequent registration of the element from its initialization process. If the time interval expires before registration is received, then a *restart time-out* of the element is recognized. An error message is produced, an ENF is issued indicating that ARM is no longer handling restarts for the element, and all resources related to the element are cleaned up. Lastly, ARM deregisters the element. If the element was just delayed and registers after the restart time-out, then ARM will treat this as a new registration.

Once registered, the time-out interval is set for the element to become ready. If this expires, then a warning message is produced, but no other actions occur. The element would remain in the RECOVERING state. Finally, when the element requests the READY service, the state is set to AVAILABLE, and restart processing is considered complete.

6.10 Restart on different systems

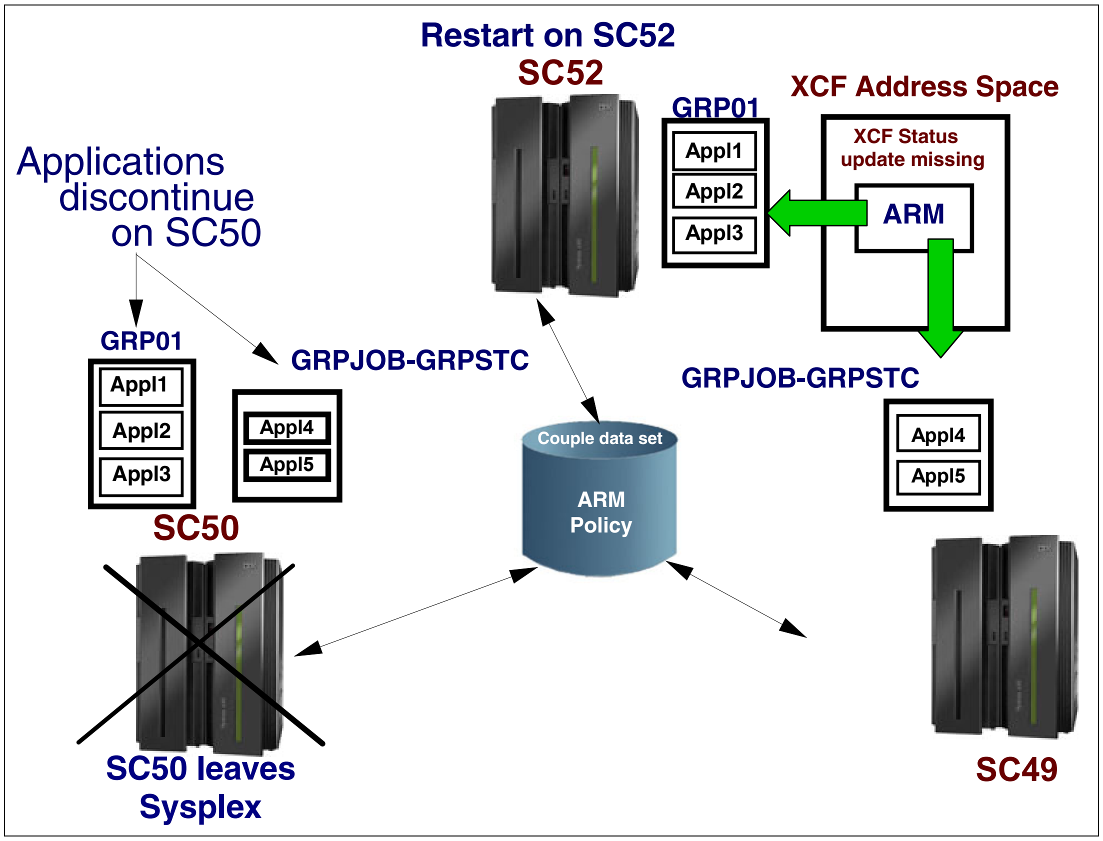


Figure 6-12 Restart on different systems

Restart on different systems

Figure 6-12 shows the factors involved in restarting work on the same system. When a system fails or otherwise leaves the sysplex, ARM has the overall responsibility of restarting registered elements that had been running on the failed system, on the remaining systems in the sysplex. ARM depends on several other components to achieve this goal.

Operator commands for ARM restart

In order to perform restarts, ARM must issue operator commands from the XCF address space (XCFAS). You must be certain that the XCFAS has enough authority to issue any commands required to restart a failed element.

Note: The XCF address space must have adequate authorization through the z/OS Security Server (RACF) or an equivalent security product.

ARM restart actions

The first action taken is the detection and the initial termination of the failed system. Following is a summary of the actions performed by other components:

1. The detecting z/OS XCF issues a *status update missing event* because the XCF failure detection interval has elapsed.
2. The XCF indeterminate status interval expires. Available options are:
   – Operator prompt

   - Reset after isolation
   - LPAR system reset or deactivation via ARF (PR/SM only)
3. The failed system is removed from the sysplex (sysplex partitioning) by:
   - Logical partition deactivation and acquiring storage resources via ARF (PR/SM only)
   - Multisystem XCF exploiters recovering resources owned by the failed system:
     - Console services - release console resources owned by the failed system
     - GRS - release global ENQs owned by the failed system

Processing on remaining systems

All remaining systems are notified of the system's termination through the XCF group exit. XCF, on each z/OS image in the sysplex, directly notifies ARM of the system's termination. The ARM processing that initiates restart processing runs on the z/OS image that first detected the system termination. The ARM processing on this system has several responsibilities:

- Determine if ARM's threshold for system terminations prohibits its restarting of the element that had been running on the failed system
- Determine if any registered elements were executing on the failed system
- Determine which elements can be restarted on another system
- Determine which elements must be restarted on the same system (same restart group)
- Determine the system on which to restart each restart group

Sysplex analysis

When a system leaves the sysplex, ARM determines if there have been two or more other SYSGONE events within the last ten minutes. If so, ARM does not restart any of the elements from the system that experienced this latest SYSGONE event. ARM issues a message for this event. This is done in case there is some problem in ARM or in its elements that causes the system failure. If left unchecked, such a problem could cause ARM restarts to cascade the failure to many or all systems in the sysplex.

ARM restart of a different system

ARM attempts to restart the elements from a system that left the sysplex on the remaining systems that have the most available capacity. The steps that ARM follows to identify and prioritize the candidate systems are:

- Determine the relative available capacity and available CSA/ECSA. ARM builds a list of these systems, sorted according to the available capacity.
- For each restart group that is to be restarted, ARM will then:
  - Identify the systems that are in the same JES XCF group as the system that has terminated. Systems not in the same JES XCF group are deleted from the list of systems just created.
  - Determine if the installation's policy specifies either TARGET_SYSTEM or FREE_CSA for the restart group, and if so, eliminate from the list any system that does not meet the criteria defined by these parameters.

  The top system on the list is then assigned to restart this restart group. If there are no candidate systems left after these checks, ARM aborts the restart of that restart group.

Workload restart exit

On each system where restarts are to occur, ARM calls your workload restart exit if it is installed. This allows the installation to do things such as cancel work that is presently running on that system to prepare for the elements that ARM is about to restart. ARM then initiates restart processing for the elements within the restart group. They are essentially started at the same time to allow for maximum parallelism during restart processing.

6.11 Group restart overview

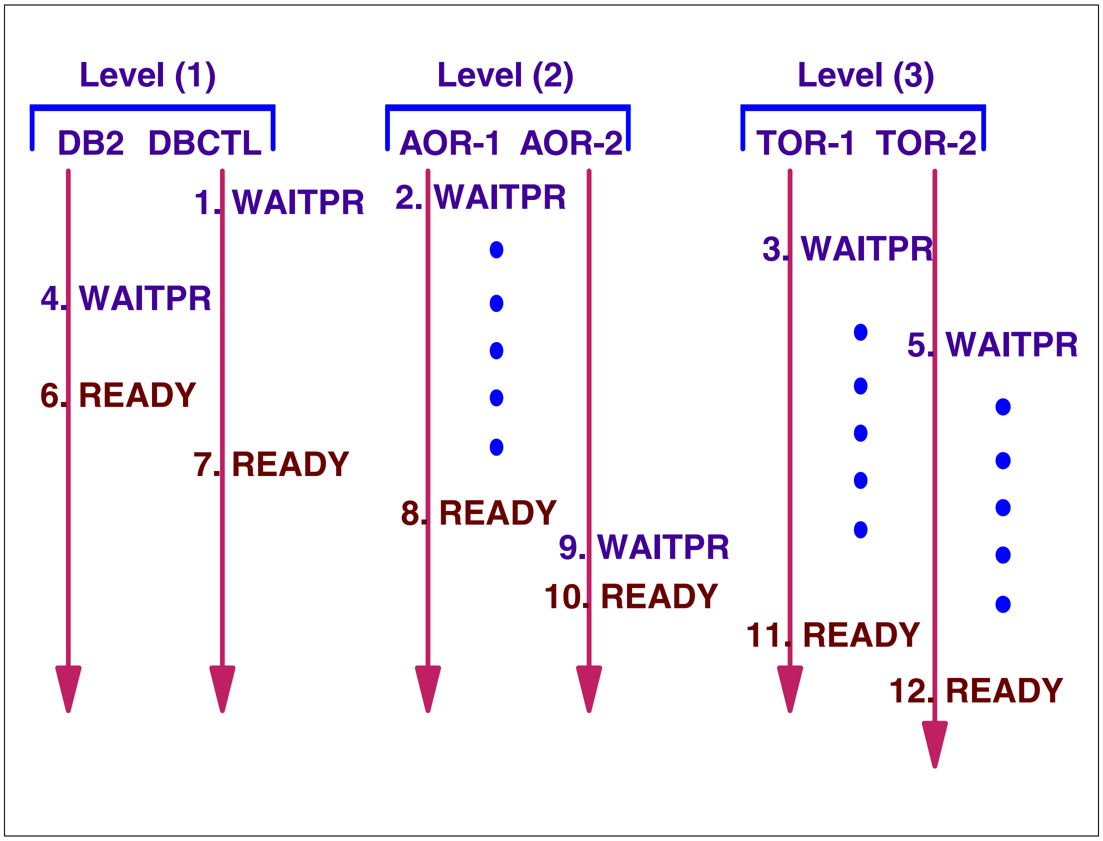


Figure 6-13 Group restart overview

Group restart overview

This example of a group restart shows how DB2 and CICS coordinate their restarts. Both DB2 and CICS can be started at the same time, including AORs and TORs. Since they are at different levels, as defined in the ARM policy, they must wait for higher levels to complete their initialization before they can continue, so they issue a WAITPRED to wait for a higher level to issue a READY.

When the ready status of a higher level is given, the next level can continue its initialization.

RESTART_ORDER processing

Specifies the order in which elements in the same restart group are to become ready after they are restarted, as shown in Figure 6-13:

RESTART_ORDER The RESTART_ORDER parameter applies to all restart groups. RESTART_ORDER is optional. The default values for RESTART_ORDER are:

- LEVEL(0) - ELEMENT_TYPEs: DB2, DBCTL
- LEVEL(1) - ELEMENT_TYPEs: AOR-1, AOR-2
- LEVEL(2) - ELEMENT_TYPEs: TOR-1, TOR-2

LEVEL(level) Specifies the level associated with elements that must be restarted in a particular order. The elements are restarted from the lowest level to the highest level. The LEVEL keyword can be specified multiple times.

The set of elements associated with a particular level is identified by the ELEMENT_NAME or ELEMENT_TYPE parameters. ELEMENT_NAME or ELEMENT_TYPE must be specified with each LEVEL specification. (You can specify both ELEMENT_NAME and ELEMENT_TYPE.)

Level must be a decimal number from 0 through 65535.

Group restart considerations

Determine which batch jobs and started tasks will be using automatic restart management for recovery purposes. For the IBM products that use automatic restart management, read their documentation for any policy considerations. Here are the parameters that can be used for the restart processing:

RESTART_GROUP Determine whether any of these elements is interdependent, that is, needs to run on the same system.

> **Note:** Any elements that are not explicitly assigned to a restart group become part of the restart group named DEFAULT. Thus, if these elements are restarted, they are restarted on the same system.

RESTART_ORDER Determine whether there is an order in which MVS should restart these elements. That is, are any elements in the restart group dependent upon other elements being restarted and ready first?

RESTART_PACING Determine whether the elements in a restart group need to be restarted at specific intervals.

TERMTYPE Determine whether the element should be restarted when only the element fails, or when either the element or the system fails.

RESTART_METHOD Determine whether specific JCL or command text is required to restart an element.

FREE_CSA Determine whether a minimum amount of CSA/ECSA is needed on the system where the elements in a restart group are to be restarted.

TARGET_SYSTEM Determine whether you want the elements in the restart group to be restarted on a specific system.

6.12 ARM exit facilities

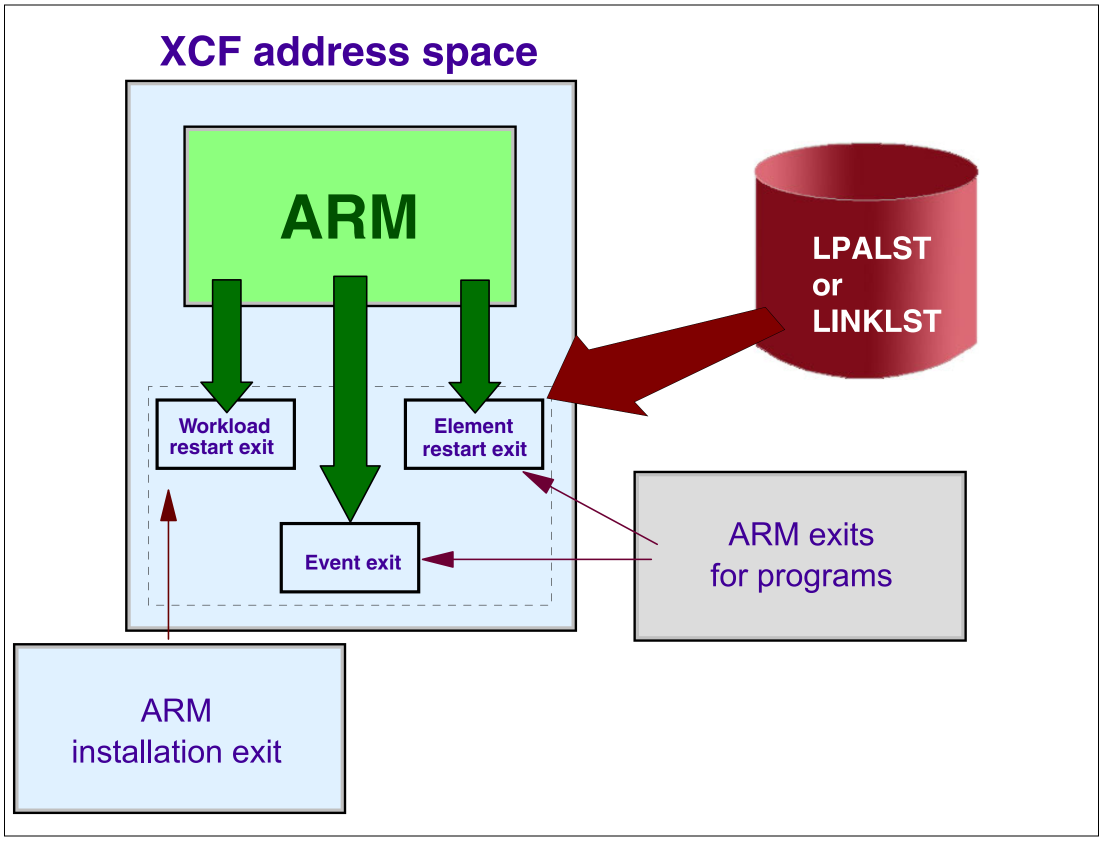


Figure 6-14 ARM exit facilities

ARM exit facilities

The exits shown in Figure 6-14 are described in this topic.

The workload restart exit (IXC_WORK_RESTART)

Through the IXC_WORK_RESTART exit, your installation can prepare a system to receive additional workload from a failing system in the sysplex. MVS invokes IXC_WORK_RESTART one time on each system that is selected to restart work from a failing system. MVS selects the system most capable of handling the additional work. Because of the system’s resources or unusual workload, your installation might want to improve this system’s capability. Your installation can do so by coding the workload restart exit to perform tasks such as cancelling lower priority work.

This exit cannot cancel or change the restart of an element. This exit is executed once on each system where ARM is about to restart elements that had been running on a system that terminated. You might provide a routine for this exit to cancel low-priority work in preparation for the additional workload from the failed system that ARM is about to restart.

The element restart exit (IXC_ELEM_RESTART)

Through the IXC_ELEM_RESTART exit, your installation can modify or cancel the automatic restart management initiated restart of an element. Your installation can use this exit to coordinate the restart of an element with other automation routines, and to make decisions about how, or if, it will be restarted. ARM invokes this exit once for each element that is to be restarted, on the system where it will be restarted.

This exit is executed before ARM restarts an element. It is used to coordinate restarts with the automation packages you may have installed.

The event exit

This exit is usually provided by a program that is registering to become an ARM element. ARM invokes this exit for certain events that affect the element. For example, you could use the ENFREQ macro to listen for the deregister ENF signal for an element, then restart the element. ARM issues an asynchronous ENF signal, event-code 38, to communicate the element type to listeners of the automatic restart management ENF Code 38 for events that pertain to this element.

6.13 ARM SMF records

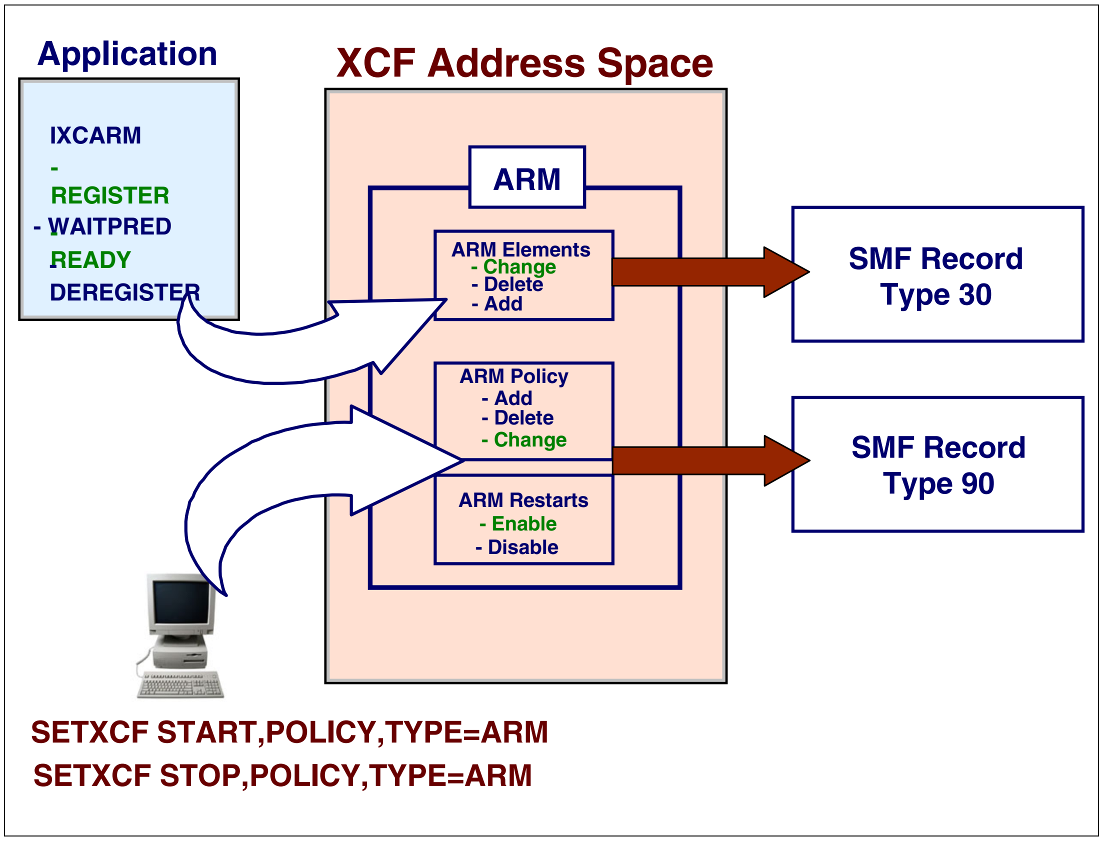


Figure 6-15 ARM SMF records

ARM SMF records

The Type 30 (common address space work) and Type 90 (system status) SMF records have been enhanced to provide information relating to ARM activity, elements, and policy management.

The SMF Type 30 record has been altered in these areas:

- Header section
- Product section
- ARM section

SMF record type 30

The new ARM section contains element identification information and time stamps for when various events complete. If the time stamp for an event is zero, it usually indicates that the event failed. This new section is created each time an element:

- Is started or restarted - The element issued an IXCARM REGISTER request.
- Is explicitly waiting for other elements to become functional - The element issued an IXCARM WAITPRED request.
- Is functional - The element issued an IXCARM READY request.
- Is no longer functional - The element issued an IXCARM DEREGISTER request during normal termination.

As each record is generated, the ARM time stamps that have been collected to that point will be included in the record. If a started task or job running in an address space has not requested an IXCARM REGISTER, then there will be no ARM section in the record. However, once the IXCARM REGISTER is requested, the ARM section will appear in every future Type 30 record that is generated for that started task or job.

SMF record type 90

The SMF Type 90 record helps to assist in tracking the use of ARM policies. ARM generates a Type 90, subtype 27, 28, 29, or 30 record when one of the following events occur:

- An ARM policy is defined (new or changed) on the couple data set
- An ARM policy is deleted from the couple data set
- ARM is enabled for restarts via the SETXCF START (and a policy may be activated or changed)
- ARM is disabled for restarts via the SETXCF STOP (and a policy may be deactivated)

SMF record type 90 contents

The contents of the SMF Type 90 records include:

- A function code identifying the event related to this record
- The name of the ARM policy that was affected:
  - The policy, if any, that was activated via a **SETXCF START** command
  - The policy, if any, that was deactivated via a **SETXCF STOP** command
  - The policy that was either defined (new or changed) or deleted from the ARM couple data set
- The name of the user who performed this function

ARM creates an SMF Record Type 90 (with new subtypes) to provide an audit trail when ARM restarts are enabled or disabled or a new policy is activated. This enables you to determine what ARM policy was active across a given interval if a question arises about an ARM action during that interval.

7

Geographically Dispersed Parallel Sysplex

How would a shutdown of your z/OSs affect your business? Do you put off system maintenance and upgrades to avoid system downtime? Is your business-critical processing and data protected from a site disaster? For how many hours (or minutes) can your enterprise survive without running critical business transactions? What is the cost associated with the outage?

The costs associated with an outage are expected to be much higher now because all enterprises have become much more dependent on Information Technology (I/T). For example, a retail brokerage business could experience a massive hourly impact because on-line trading volumes have dramatically increased; cell phone and ATM usage has also increased significantly.

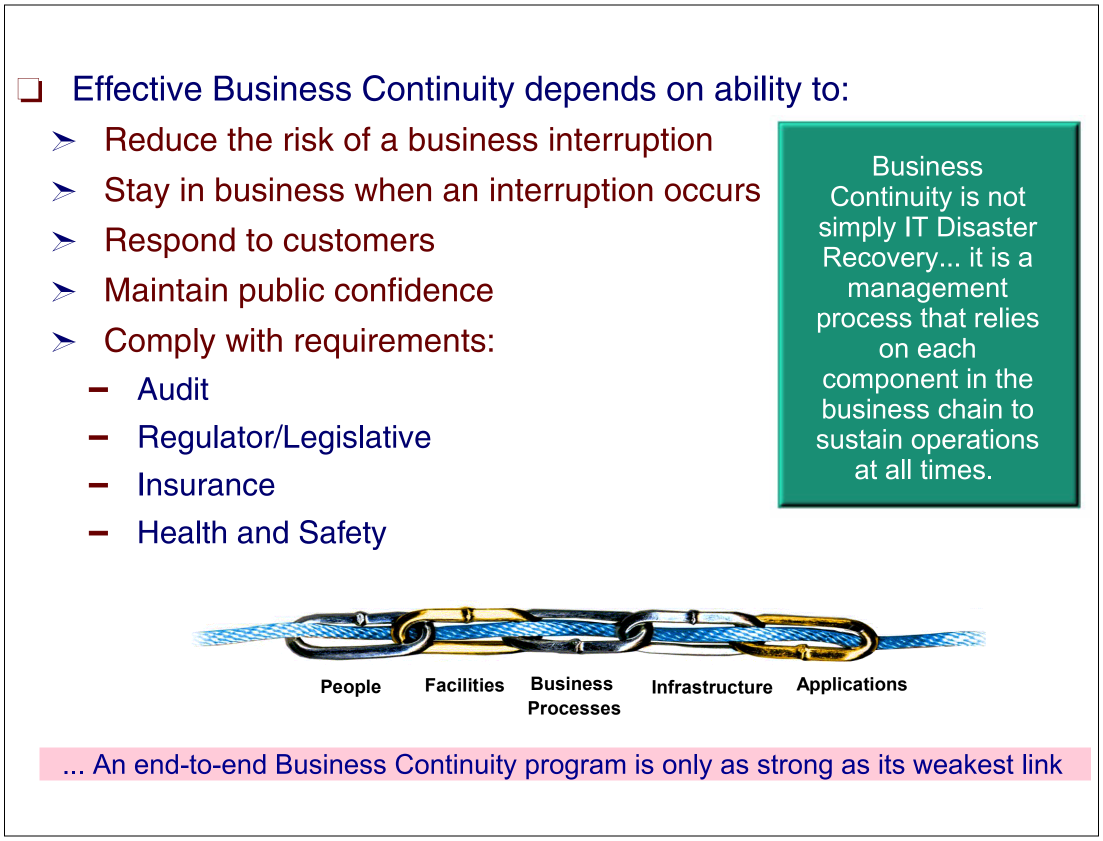
An important part of preparing for a disaster is understanding the type of risks an organization faces in relation to its own continuance.

This chapter covers the business continuance (also called business continuity) plan and one of its major components – the IT disaster recovery plan. A disaster recovery plan focuses on continuity of IT operations. Geographically Dispersed Parallel Sysplex (GDPS®) is an IBM product that can assist the IT department in reducing the outage time for critical business transactions and enforcing data integrity during a disaster. GDPS is constituted by service, code, and automation procedures.

From an IT-centric perspective, outages are classified as planned or unplanned disruptions to operations. In modern business, we expect that planned outages will be a very rare event (as in a site shutdown). When some component in the IT solution is stopped for maintenance, it should not cause a planned outage for key business transactions. To avoid such an outage, lots of redundancies has been introduced into modern systems. An unplanned IT outage can equate to a disaster, depending on the scope and severity of the problem.

A disaster, accord to its unfolding time, can be a “rolling disaster” or an “instantaneous” one. It may not be obvious, but an instantaneous disaster is actually preferable to a rolling one because of data integrity issues.

7.1 Business continuity



- Effective Business Continuity depends on ability to:
  - Reduce the risk of a business interruption
  - Stay in business when an interruption occurs
  - Respond to customers
  - Maintain public confidence
  - Comply with requirements:
    - Audit
    - Regulator/Legislative
    - Insurance
    - Health and Safety

Business Continuity is not simply IT Disaster Recovery... it is a management process that relies on each component in the business chain to sustain operations at all times.

People
Facilities
Business Processes
Infrastructure
Applications

... An end-to-end Business Continuity program is only as strong as its weakest link

Figure 7-1 Business continuity

Business continuity

Business Continuity Planning (BCP) is an enterprise-wide planning process that creates detailed procedures to be used in the case of a large unplanned outage or disaster. Maintaining continuity of business processes is the overall objective.

When investigating IT resilience options, two things that must be at the forefront of your thinking are:

- **Recovery Time Objective (RTO) -** This is how long your business can afford to wait for IT services to be resumed *following a disaster*.

  If this number is not clearly stated now, think back to the last time you had a significant service outage. How long was that outage, and how much pain did your company suffer as a result? This will help you get a feel for whether your RTO should be measured in days, hours, or minutes.

- **Recovery Point Objective (RPO) -** This is how much data your company is willing to recreate *following a disaster*. In other words, what is the acceptable time difference between the data in your production system and the data at the recovery site.

  As an example, if your disaster recovery solution depends on once daily full volume tape dumps, your RPO is 24 to 48 hours depending on when the tapes are taken off site. If your business requires an RPO of less than 24 hours you will almost certainly be forced to do some form of offsite real-time mirroring instead of relying on these tapes alone.

Business processes for recovery

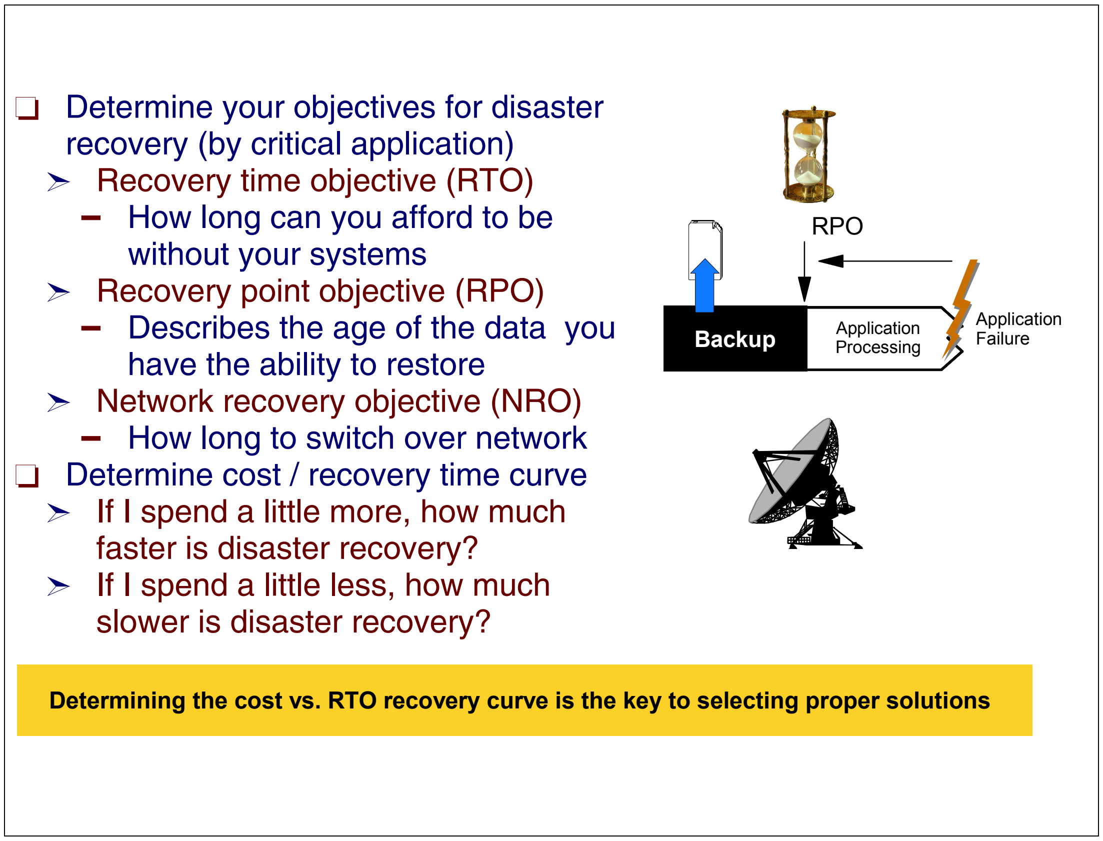
Generally, some form of real-time software or hardware replication will be required to achieve an RPO of minutes or less, but the only technologies that can provide an RPO of zero are synchronous replication technologies coupled with automation to ensure no data is written to one location and not the other.

The recovery time is largely dependent on the availability of hardware to support the recovery as well as control over that hardware. You might have real-time software- or hardware-based replication in place, but without server capacity at the recovery site you will have hours to days before you can recover this once very current data. Furthermore, even with all the spare capacity and current data, you might find that you are relying on people to perform the recovery actions. In this case, you will undoubtedly find that these same people are not necessarily available in the case of a true disaster or, even more likely, they find that processes and procedures for the recovery are neither practiced nor accurate. This is where automation comes in to mitigate the point of failure introduced by the human element and to ensure you actually meet the RTO required of the business.

Planning for recovery involves the following plans and decisions:

1. The *disaster recovery plan* should alleviate the risk of a total IT system failure, or the risk that a disaster will affect your production systems.
2. A *business recovery plan* addresses the people side of your business processes—how to handle the staff during recovery and who is responsible for the different recovery activities.
3. A *business resumption plan* is the work-around plan to be followed until the business processes are back online.
4. A *contingency plan* preplans for external events that could affect your company, such as loss of a critical supplier or key members of the executive team.
5. A *crisis management plan* directs overall management of the event, for example, supporting personnel and families and handling statements to the public and major customers."

7.2 Disaster recovery objectives



- ❑ Determine your objectives for disaster recovery (by critical application)
  - ➢ Recovery time objective (RTO)
    - – How long can you afford to be without your systems
  - ➢ Recovery point objective (RPO)
    - – Describes the age of the data you have the ability to restore
  - ➢ Network recovery objective (NRO)
    - – How long to switch over network
- ❑ Determine cost / recovery time curve
  - ➢ If I spend a little more, how much faster is disaster recovery?
  - ➢ If I spend a little less, how much slower is disaster recovery?

Determining the cost vs. RTO recovery curve is the key to selecting proper solutions

Figure 7-2 Disaster recovery objectives

Disaster recovery objectives

In today’s highly competitive e-business world, outages can have a devastating impact on a business; they could mean its demise. Many companies have inadequate business continuance plans developed on the premise that back office and manual processes will keep the business running until computer systems are available. As mentioned earlier, the practice of preparing for *Disaster Recovery (DR)* is something that has been a focus of IT planning for many years. In turn, there is a wide range of offerings and approaches available to accomplish DR. Some options rely on off-site or even outsourced locations that are contracted to provide data protection or even servers in the event of a true IT disaster. Other options rely on in-house IT infrastructures and technologies that can be managed by your own teams. There is no right answer for which approach is better for every business, but the first step in deciding what makes the most sense for you is to have a good view of your IT resiliency objectives; specifically, your RPO and RTO.

Common DR option for RPO and RTO

Although Table 7-1 on page 297 is not comprehensive of all possible DR offerings and approaches, it does provide a view of what RPO and RTO might typically be achieved with some common options.

Generally, some form of real-time software or hardware replication will be required to achieve an RPO of minutes or less, but the only technologies that can provide an RPO of zero are synchronous replication technologies coupled with automation to ensure no data is written to one location and not the other.

Table 7-1 Typical achievable RPO and RTO for some common DR options

Description	Typically achievable Recovery Point Objective (RPO)	Typically achievable Recovery Time Objective (RTO)
No disaster recovery plan	N/A - all data lost	N/A
Tape vaulting	Measured in days since last stored backup	Days
Electronic vaulting	Hours	Hours (hot remote location) to days
Active replication to remote site (w/o recovery automation)	Seconds to minutes	Hours to days (dependent on availability of recovery hardware)
Active storage replication to remote "in-house" site	Zero to minutes (dependent on replication technology and automation policy)	1 or more hours (dependent on automation)

Recovery time

The recovery time is largely dependent on the availability of hardware to support the recovery as well as control over that hardware. You might have real-time software- or hardware-based replication in place, but without server capacity at the recovery site you will have hours to days before you can recover this once very current data. Furthermore, even with all the spare capacity and current data, you might find that you are relying on people to perform the recovery actions. In this case, you will undoubtedly find that these same people are not necessarily available in the case of a true disaster or, even more likely, they find that processes and procedures for the recovery are neither practiced nor accurate. This is where automation comes in to mitigate the point of failure introduced by the human element and to ensure you actually meet the RTO required by the business.

DR options

Finally, you might decide that one DR option is not appropriate for all aspects of the business. Some applications may tolerate a much greater loss of data and may not have an RPO as low as others. At the same time, some applications may not require recovery within hours whereas others most certainly do. While there is obvious flexibility in choosing different DR solutions for each application, the approach supported by GDPS is to provide a single optimized solution for the enterprise. This generally leads to a simpler solution and, because less infrastructure and software might need to be duplicated, often a more cost-effective solution too.

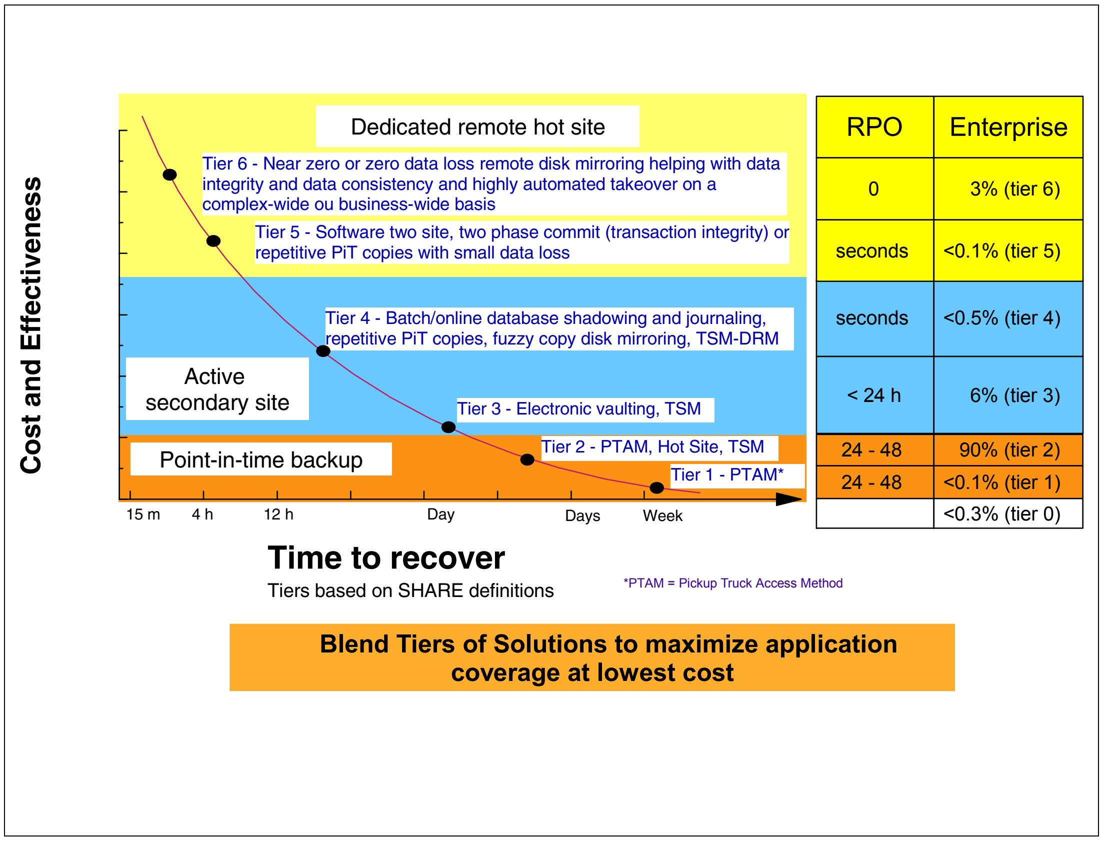
Disaster recovery definitions

Just to review, following are a few considerations regarding disaster recovery:

Recovery time objective (RTO):	What is the business cost-justified elapsed time to recovery?
Recovery point objective (RPO):	When the Recovery Time Objective is met, what amount of data must be recreated?
Network recovery objective (NRO):	How long does it take to switch the entire network over to the backup data center?

These first two objectives (RTO and RPO) can often be balanced against each other to optimize the cost/benefit ratio. When the third objective (NRO) comes into play, networking issues come into consideration. For example: there is no need to "purchase" a 30-minute RTO solution if the network provider requires two hours to switch the network.

7.3 SHARE disaster/recovery tiers



RPO	Enterprise
0	3% (tier 6)
seconds	<0.1% (tier 5)
seconds	<0.5% (tier 4)
< 24 h	6% (tier 3)
24 - 48	90% (tier 2)
24 - 48	<0.1% (tier 1)
	<0.3% (tier 0)

Figure 7-3 SHARE disaster/recovery tiers

SHARE disaster/recovery tiers

In 1992, the SHARE user group in the United States, in combination with IBM, defined a set of disaster recovery tier levels. This was done to address the need to properly describe and quantify various methodologies for successful mission-critical computer systems disaster recovery implementations. Accordingly, within the IT Business Continuance industry, the tier concept continues to be used, and is very useful for describing today’s disaster recovery capabilities. They need only to be updated for today’s specific disaster recovery technologies and associated RTO/RPO.

Disaster recovery tiers

The “Seven Tiers of Disaster Recovery Solutions” offer a a simple methodology of how to define your current service level, the current risk, and the target service level and target environment.

7.4 Tier 0 - No off-site data

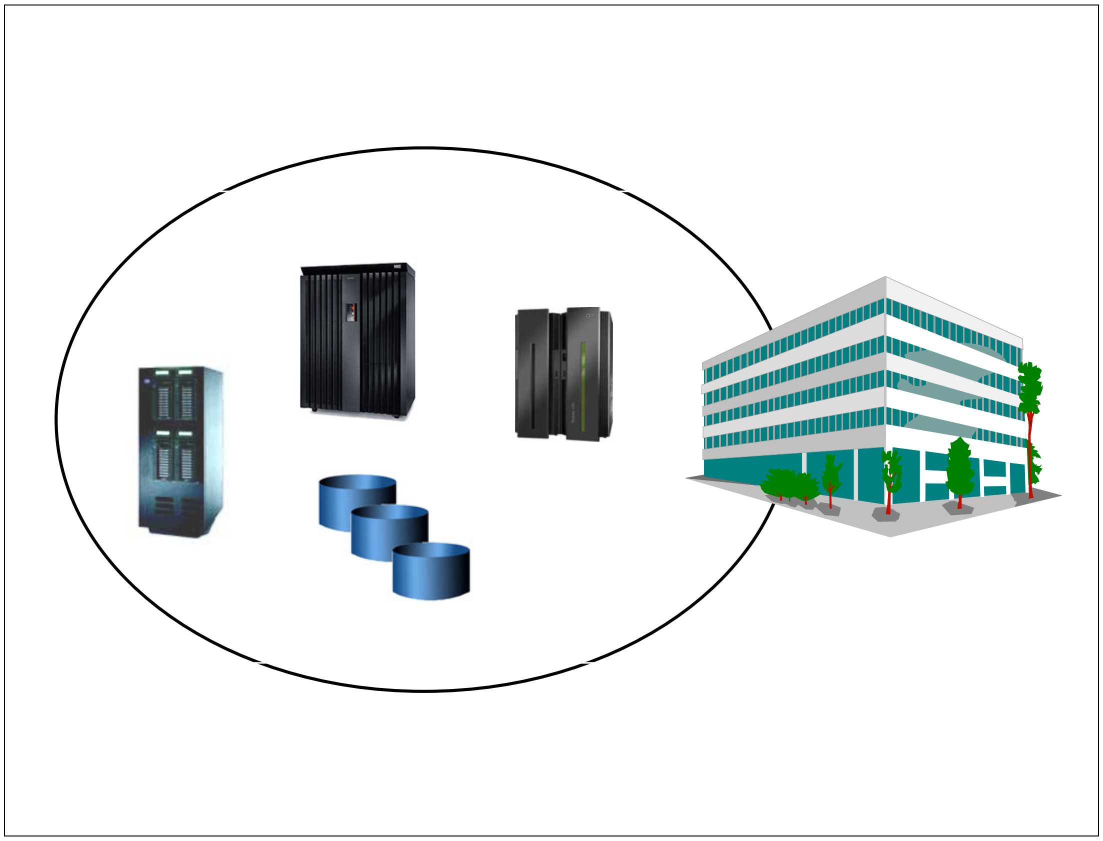


Figure 7-4 Tier 0 - No off-site data

Tier 0 - No off-site data

Following is some information on Tier 0:

- Tier 0 is defined has having no requirements to back up data or implement a Disaster Recovery Plan. There is therefore no disaster recovery capability at all.
- There is no saved information, no documentation, no backup hardware, and no contingency plan.
- Typical recovery time: The length of recovery time in this instance is unpredictable. In fact, you may not be able to recover at all.

Figure 7-4 shows an example of a Tier 0 installation. This demonstrates that no backups are being taken, nor are they being stored off-site.

7.5 Tier 1- Pickup truck access method

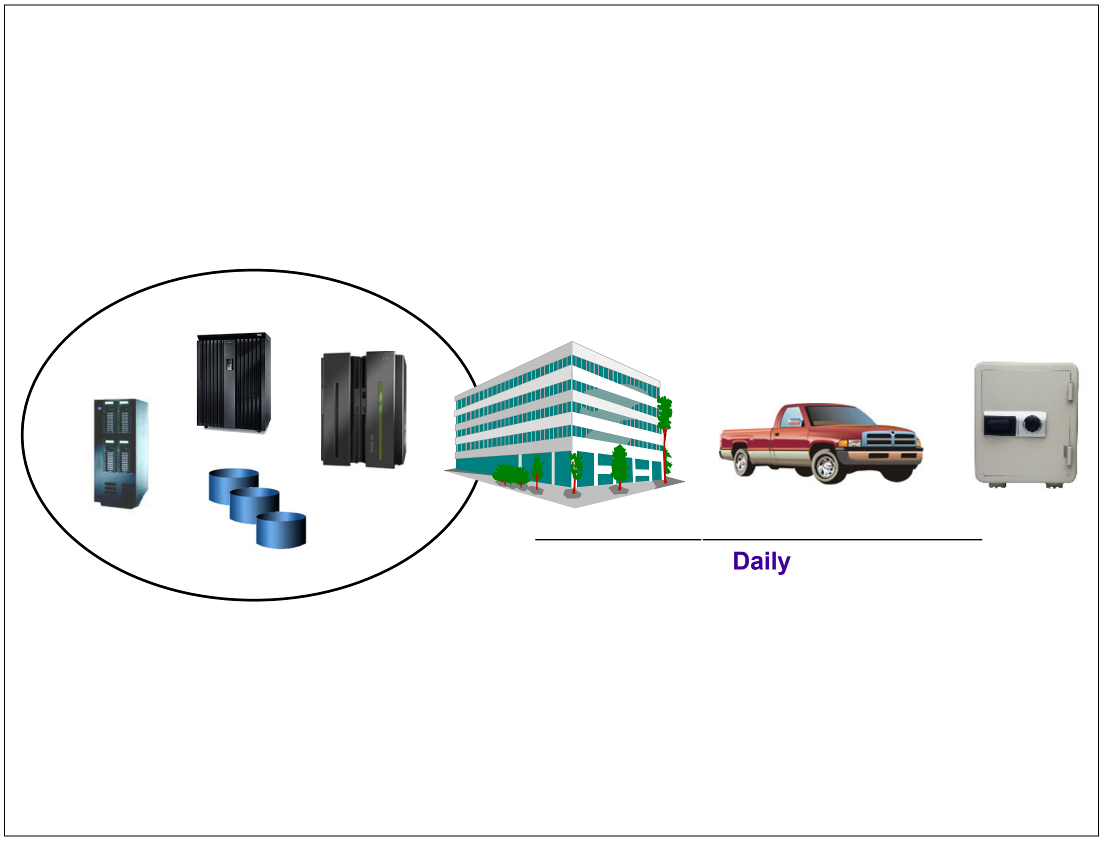


Figure 7-5 Tier 1 - Pickup truck access method

Tier 1 - Pickup truck access method

Following is some information on Tier 1:

- A Tier 1 installation is defined as having a Disaster Recovery Plan. It backs up and stores its data at an off-site storage facility and has determined some recovery requirements. It may also have established a backup platform, although it does not have a site at which to restore its data, nor the necessary hardware on which to restore the data.
- Pickup truck access method (PTAM) is a method used by many sites, as this is a relatively inexpensive option. It can, however, be difficult to manage; that is, difficult to know where the data is.
- There is selectively saved data. Certain requirements have been determined and documented in a contingency plan and there is optional backup hardware and a backup facility available.
- Recovery is dependent on when hardware can be supplied, or possibly when a building for the new infrastructure can be located and prepared
- Typical recovery time: The length of time for recovery is normally more than a week.

Figure 7-5 shows an example of a Tier 1 installation. This illustrates that backups are being taken and stored at an off-site storage facility.

7.6 Tier 2 - PTAM and hot site

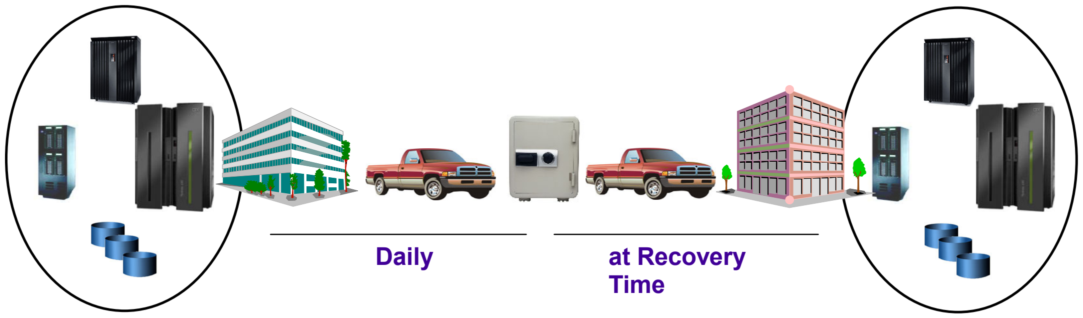


Figure 7-6 Tier 2 - PTAM and hot site

Tier 2 - PTAM and hot site

Following is some information on Tier 2:

- Tier 2 encompasses all requirements of Tier 1 plus a hot site. The hot site has sufficient hardware and a network infrastructure able to support the installation's critical processing requirements.
- Processing is considered critical if it must be supported on hardware existing at the time of the disaster.
- Tier 2 installations rely on PTAM to get the data to a storage facility. In the event of a disaster, the data from the storage facility is moved to the hot site and restored onto the backup hardware provided.
- Moving to a hot site increases the cost but reduces the recovery time significantly.
- Typical recovery time: The length of time for recovery is normally more than a day.

Figure 7-6 shows an example of a Tier 2 installation. This demonstrates that backups are being taken and are being stored at an off-site storage facility. There is also a hot site available and the backups can be transported there from the off-site storage facility.

7.7 Tier 3 - Electronic vaulting

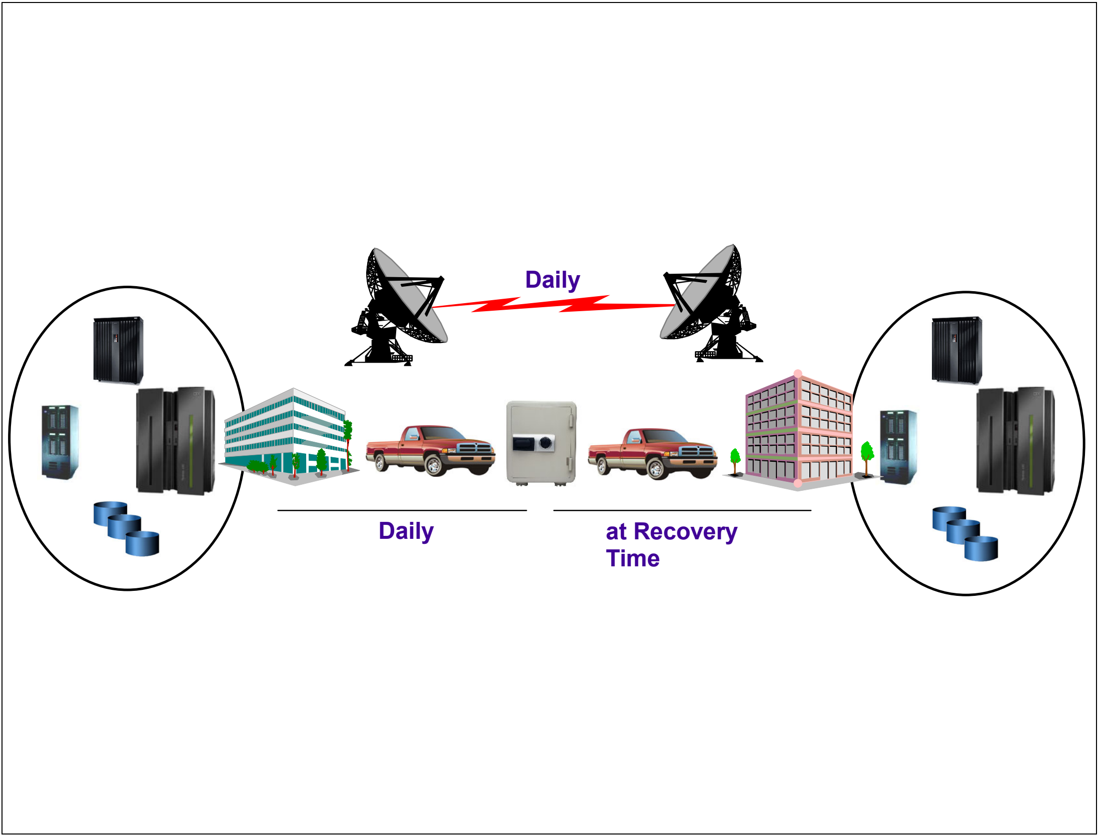


Figure 7-7 Tier 3 - Electronic vaulting

Tier 3 - Electronic faulting

Following is some information on Tier 3:

- Tier 3 encompasses all the components of Tier 2 and, in addition, supports electronic vaulting of some subset of the critical data.
- The receiving hardware must be physically separated from the primary site and the data stored for recovery should there be a disaster at the primary site.
- The hot site is kept running permanently, thereby increasing the cost. As the critical data is already being stored at the hot site, the recovery time is once again significantly reduced.
- Typical recovery time: The length of time for recovery is usually about one day.

Figure 7-7 shows an example of a Tier 3 installation. This demonstrates that backups are being taken and are being stored at an off-site storage facility. There is also a hot site available and the backups can be transported there from the off-site storage facility. There is also electronic vaulting of critical data occurring between the primary site and the hot site.

7.8 Tier 4 - Active secondary site

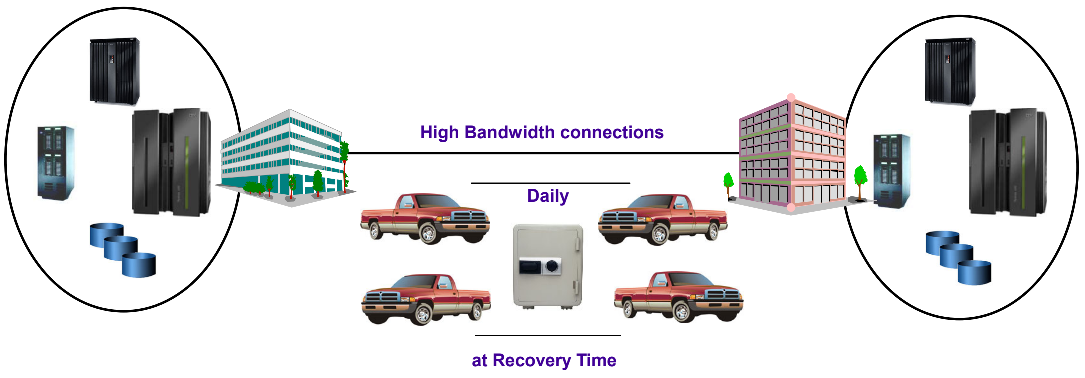


Figure 7-8 Tier 4 - Active secondary site

Tier 4 - Active secondary site

Following is some information on Tier 4:

- Tier 4 introduces the requirements of active management of the data being stored at the recovery site. This is managed by a processor at the recovery site and caters to bi-directional recovery.
- The receiving hardware must be physically separated from the primary platform.
- In this scenario, the workload may be shared between the two sites. There is a continuous transmission of data between the two sites with copies of critical data available at both sites.In other words the remote copy operation is asynchronous.
- Any other non-critical data still needs to be recovered in the event of a disaster.
- Typical recovery time: The length of time for recovery is usually up to one day.

Figure 7-8 shows an example of a Tier 4 installation. This demonstrates that backups are being taken and they are being stored at an off-site storage facility. There is also a hot site available and the backups can be transported there from the off-site storage facility. There is also continuous transmission of data between the primary site and the hot site.

7.9 Tier 5 - Two-site two-phase commit

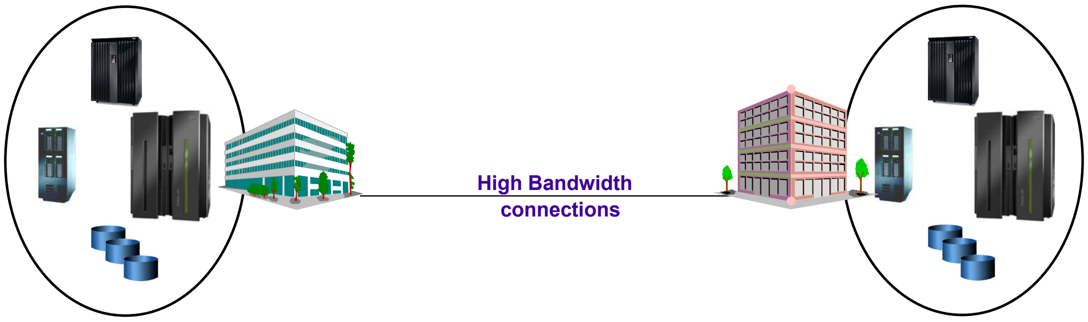


Figure 7-9 Tier 5 - Two-site two-phase commit

Two-site two-phase commit

Following is some information on Tier 5:

- Tier 5 encompasses all the requirements of Tier 4 and, in addition, will maintain selected data in image status (updates will be applied to both the local and the remote copies of the database within a single-commit scope).
- Tier 5 requires that both the primary and secondary platform data be updated before the update request is considered successful.
- Tier 5 also requires partially or fully dedicated hardware on the secondary platform with the ability to automatically transfer the workload over to the secondary platform. There is also a requirement for a high bandwidth connection between the primary and secondary sites.
- We now have a scenario where the data between the two sites is synchronized by remote two-phase commit. The critical data and applications are therefore present at both sites and only the in-flight data is lost during a disaster. With a minimum amount of data to recover and reconnection of the network to implement, recovery time is reduced significantly.
- Typical recovery time: The length of time for recovery is usually less than 12 hours.

Figure 7-9 shows an example of a Tier 5 installation. This demonstrates that the two sites are synchronized utilizing a high-bandwidth connection between the primary site and the hot site.

7.10 Tier 6 - Zero data loss

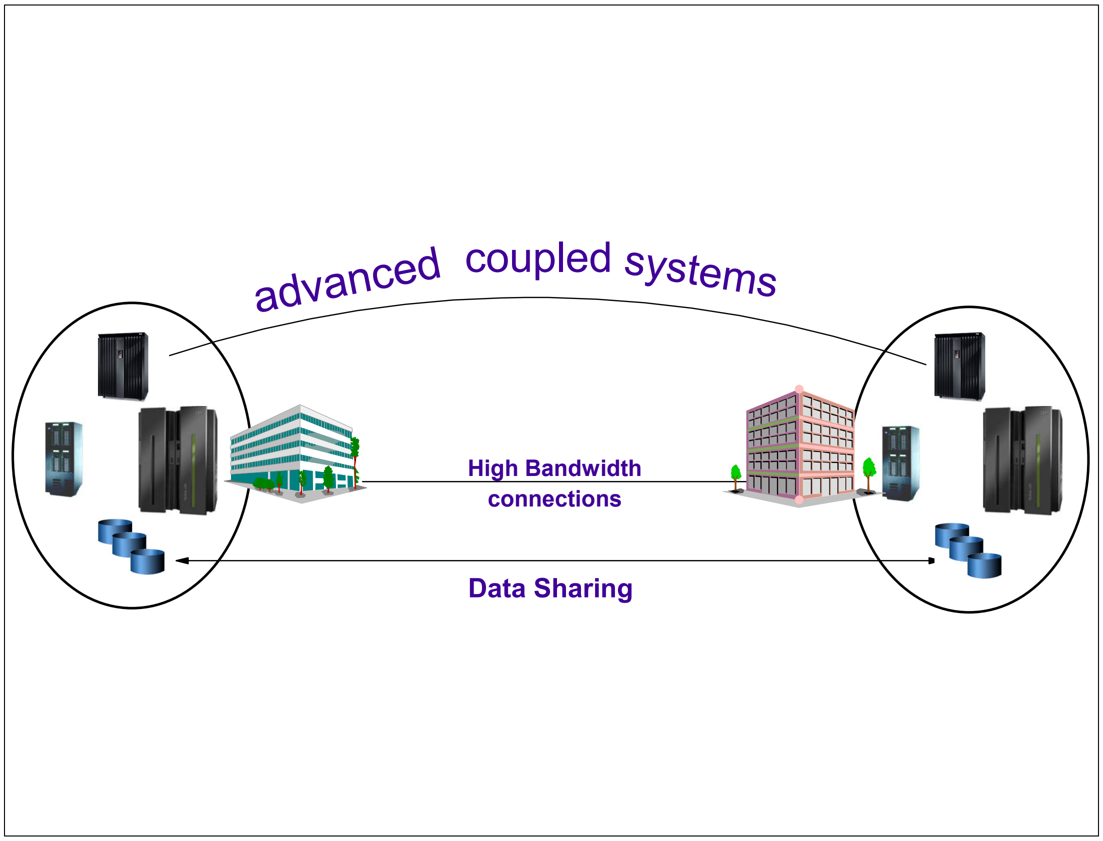


Figure 7-10 Tier 6 - Zero data loss

Tier 6 - Zero data loss

Following is some information on Tier 6:

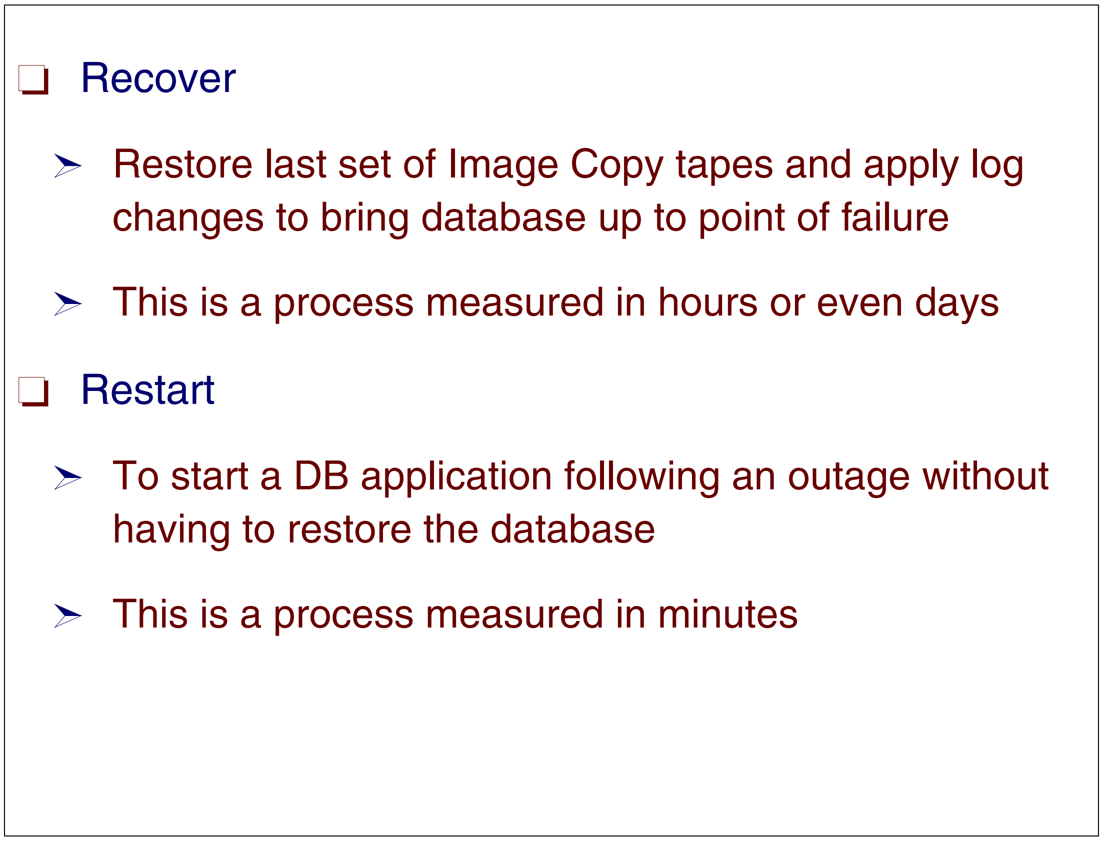
- Tier 6 encompasses zero loss of data and immediate and automatic transfer to the secondary platform.
- Data is considered lost if Enter has been accepted (at the terminal) but the request has not been satisfied.
- Tier 6 is the ultimate level of disaster recovery. Local and remote copies of all data are updated and dual online storage is utilized with a full network switching capability.
- This is the most expensive disaster recovery solution but also offers the speediest recovery by far.
- Typical recovery time: The length of time for recovery is typically a few minutes.

Figure 7-10 shows an example of a Tier 6 installation. This demonstrates that the two sites are fully synchronized utilizing a high-bandwidth connection between the primary site and the hot site. The two systems are advanced coupled, allowing an automated switchover from one site to the other when required.

The zero data loss does not imply that the RTO is zero.

One negative aspect of this tier is: “if there is a problem with the link in charge of mirroring the data then the software writing data in primary site gets back an unrecoverable I/O error that may cause transaction abends.”

7.11 Database restart versus recovery



- Recover
  - Restore last set of Image Copy tapes and apply log changes to bring database up to point of failure
  - This is a process measured in hours or even days
- Restart
  - To start a DB application following an outage without having to restore the database
  - This is a process measured in minutes

Figure 7-11 Database restart versus recovery

Database restart versus recovery

The ability to do a database restart, rather than a database recovery, is essential to meet the recovery time objective (RTO) of many businesses – typically less than an hour. Database restart allows starting a database application (as you would do following a database manager, or system abend) without having to restore it from backups.

Recovery

Database recovery is normally a process measured in many hours (especially if you have hundreds or thousands of databases to recover), and involves restoring the last set of image copies and applying log changes to bring the database up to the point of failure. But there is more to consider than simply the data for one data manager. What if you have an application that updates data in IMS, DB2, and VSAM? If you need to do a recover for these, will your recovery tools not only allow you to recover them to the same point in time, but also to the level of granularity that ensures that either all or none of the updates made by one transaction are recovered? Being able to do a restart rather than a recover avoids these issues.

Restart

Data consistency across all primary and secondary volumes spread across any number of storage (both tape and disk) subsystems is essential to provide not only data integrity, but also the ability to do a normal database restart in the event of a disaster. Restart is just the restart of the DB application after the outage using the mirrored database on the secondary subsystem. This usually takes minutes, and does not require taking further time-consuming recovery steps.

7.12 GDPS introduction

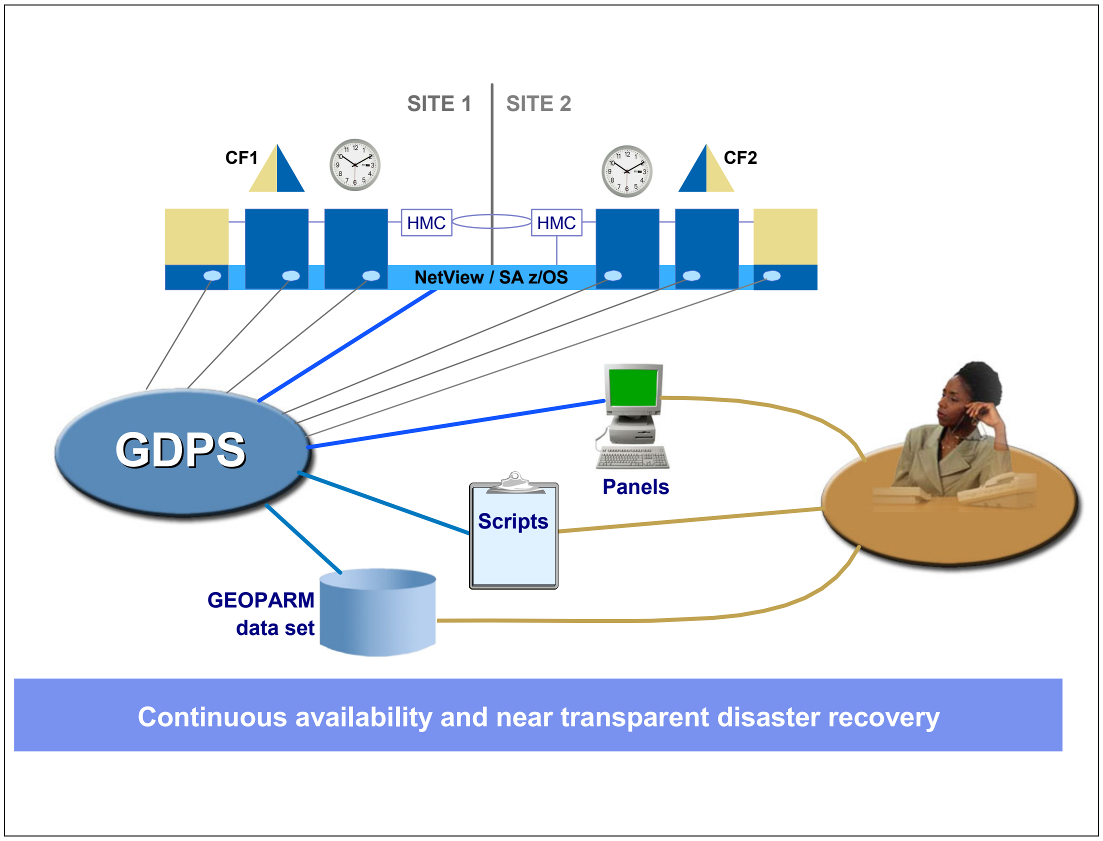


Figure 7-12 GDPS introduction

GDPS introduction

Geographically Dispersed Parallel Sysplex (GDPS) is a multi-site application availability solution that provides the capability to manage remote copy configuration and storage subsystems, automates Parallel Sysplex operational tasks, and performs failure recovery from a single point of control, thereby improving application availability and decreasing RTO figures.

The considerations for a multi-site sysplex depend on whether you plan to run production systems in both sites at the same time, or if all the production systems will be in a single site at any one time. Configurations where production systems can run in both sites at the same time are referred to as multi-site workload configurations. Configurations where the production systems run together in one site or the other (but not split between multiple sites) are referred to as single-site workload configurations or sometimes as “active/standby” configurations. Other variations on this, where production systems are predominantly running at one site, but where partially active systems or systems only enabled for queries are running at the other site, are still multi-site workloads.

Minimize impact of failures

GDPS is designed to minimize the impact of any failure, including disasters or a planned site outage. It provides the ability to perform a controlled site switch for both planned and unplanned site outages, with almost no data loss, maintaining full data integrity across multiple volumes and storage subsystems, and the ability to perform a normal Data Base Management System (DBMS) restart (not DBMS recovery) at the other site.

The majority of outages are indeed planned, and even among unplanned outages, the majority are not disasters. However, in the current business world of 24x7 Internet presence, and web-based services shared across and also between enterprises, even planned outages can be a serious disruption to your business. Unplanned outages are unexpected events. Examples of unplanned outages are software or hardware failures. Some of these outages may be quickly recovered from, but others may be considered a disaster.

You will undoubtedly have both planned and unplanned outages while running your organization, and your business resiliency processes must cater to both types. You will likely find, however, that coordinated efforts to reduce the numbers of and impacts of unplanned outages often are complementary to doing the same for planned outages.

GDPS and applications

GDPS is application independent, thereby covering your total application environment. GDPS is enabled by means of the following key IBM technologies:

- Parallel Sysplex, but could be base Sysplex
- Tivoli NetView for z/OS
- System Automation for OS/390
- IBM TotalStorage® Enterprise Storage Server® (ESS) or DS8000®
- Peer-to-Peer Virtual Tape Server (PtP VTS)
- Optical Dense Wavelength Division Multiplexer (DWDM)
- Peer-to-Peer Remote Copy (PPRC) architecture - also named Metro Mirror in DS8000 terms
- Extended Remote Copy (XRC) architecture - also named Global Mirror in DS8000 terms
- Virtual Tape Server Remote Copy architecture

GDPS support for PPRC and XRC

GDPS supports both the synchronous (PPRC) as well as the asynchronous (XRC) forms of remote copy. It also supports the PtP VTS form of remote copying tape data.

PPRC (Metro Mirror) is a hardware solution that synchronously mirrors data residing on a set of disk volumes, called the primary volumes in Site 1, to secondary disk volumes on a second system in Site 2. Only when the primary storage subsystem receives "write complete" from the secondary storage subsystem is the application I/O write signaled completed.

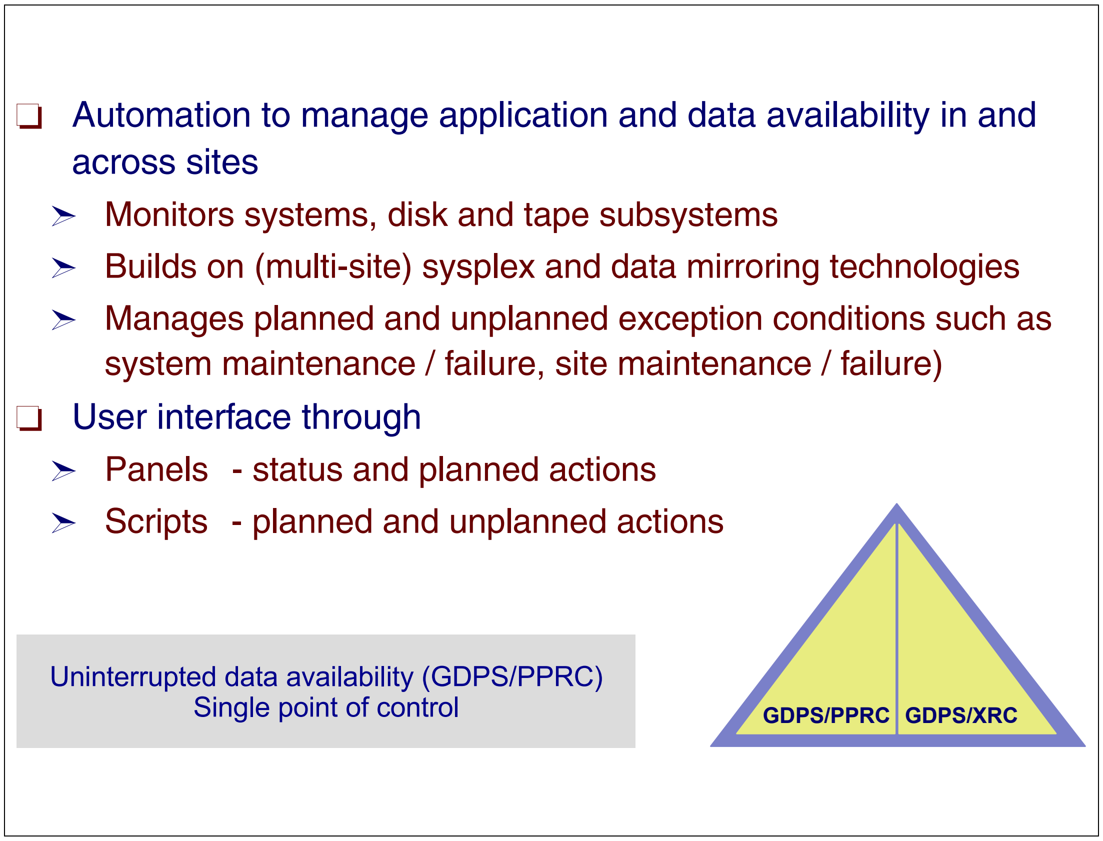
XRC (Global Mirror), on the other hand, is a combined hardware and software asynchronous remote copy solution. The application I/O write is signaled completed when the data update to the primary storage is completed. Subsequently, a DFSMSdfp component called System Data Mover (SDM), asynchronously off loads data from the primary storage subsystem's cache and updates the secondary disk volumes.

For more information, refer to the GDPS home page at:

http://www-1.ibm.com/servers/eserver/zseries/gdps/

Terminology note: In this book we continue to use the term Peer-to-Peer Remote Copy (PPRC) when referring to the synchronous disk replication architecture. The rebranded name of the IBM implementation of this architecture is IBM Metro Mirror, which will be used when specifically referring to the IBM implementation on the IBM Enterprise Storage Server (ESS) and the IBM DS8000 family of products. Similarly, we continue to use the term eXtended Remote Copy (XRC) when referring to the asynchronous disk copy technology that leverages the z/OS System Data Mover (SDM). The rebranded name of the IBM disk storage implementation is z/OS Global Mirror, which will be used specifically when referring to the IBM implementation on the IBM Enterprise Storage Server (ESS) and the IBM DS8000 family of products.

7.13 GDPS overview



- ❑ Automation to manage application and data availability in and across sites
  - ➢ Monitors systems, disk and tape subsystems
  - ➢ Builds on (multi-site) sysplex and data mirroring technologies
  - ➢ Manages planned and unplanned exception conditions such as system maintenance / failure, site maintenance / failure)
- ❑ User interface through
  - ➢ Panels - status and planned actions
  - ➢ Scripts - planned and unplanned actions

Uninterrupted data availability (GDPS/PPRC)
Single point of control

GDPS/PPRC GDPS/XRC

Figure 7-13 GDPS overview

GDPS overview

The main purpose of GDPS is to provide a disaster recovery solution mainly based on remote copy technology. There are several standard SW products to enable GDPS acting as a controlling system. One part of GDPS is the automation that ensures consistency across any number of disk subsystems.

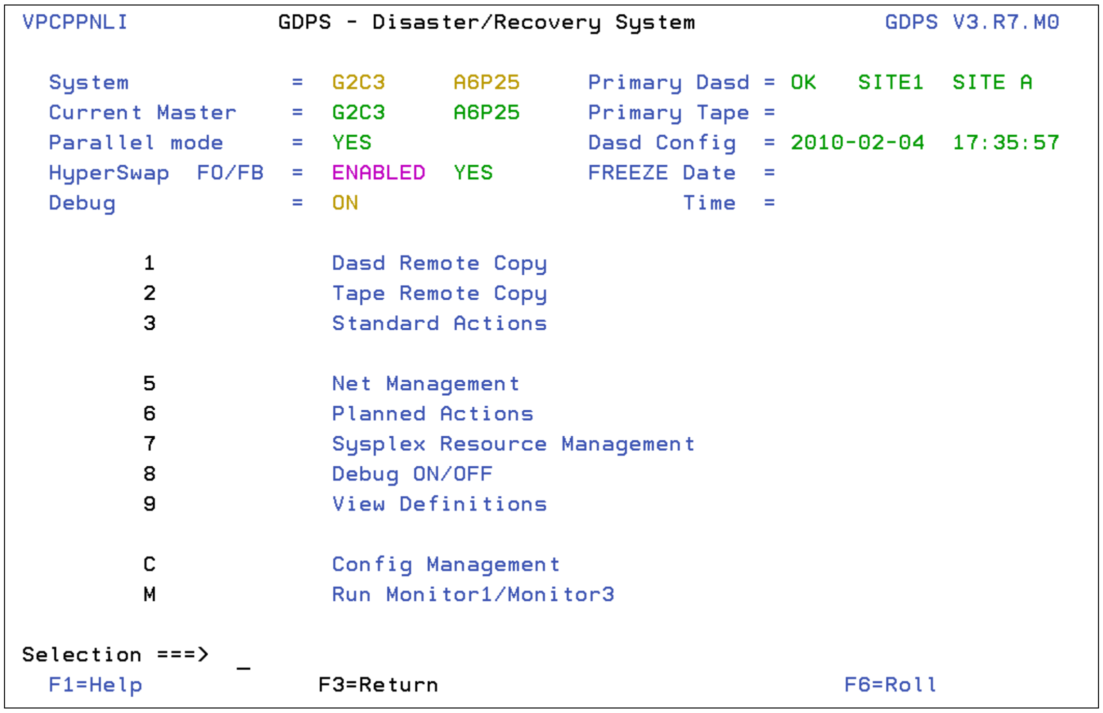
Automation

Automation is responsible for monitoring systems, DASD, and tape subsystems across recovery sites, and taking action according to defined script rules if any exception condition is detected. GDPS is also able to manage planned conditions like system and site maintenance by providing a user interface.

User interface

There are two primary user interface options available for GDPS/PPRC, the NetView 3270 panels and a browser-based graphical user interface.

An example of the main GDPS/PPRC 3270-based panel is shown in Figure 7-14 on page 310. This panel is relatively simple to use, with a summary of configuration status at the top of the panel and a menu of choices that a user can select from below. As an example, a user would simply type a `1` at the `Selection ===>` prompt and press Enter to view the disk mirroring (“DASD Remote Copy”) panels.



```
VPCPPNLI           GDPS - Disaster/Recovery System                GDPS V3.R7.M0

  System              =  G2C3     A6P25     Primary Dasd = OK    SITE1   SITE A
  Current Master      =  G2C3     A6P25     Primary Tape =
  Parallel mode       =  YES                Dasd Config  = 2010-02-04  17:35:57
  HyperSwap  FO/FB    =  ENABLED  YES       FREEZE Date  =
  Debug               =  ON                        Time  =

         1               Dasd Remote Copy
         2               Tape Remote Copy
         3               Standard Actions

         5               Net Management
         6               Planned Actions
         7               Sysplex Resource Management
         8               Debug ON/OFF
         9               View Definitions

         C               Config Management
         M               Run Monitor1/Monitor3

Selection ===> _
  F1=Help             F3=Return                                      F6=Roll
```

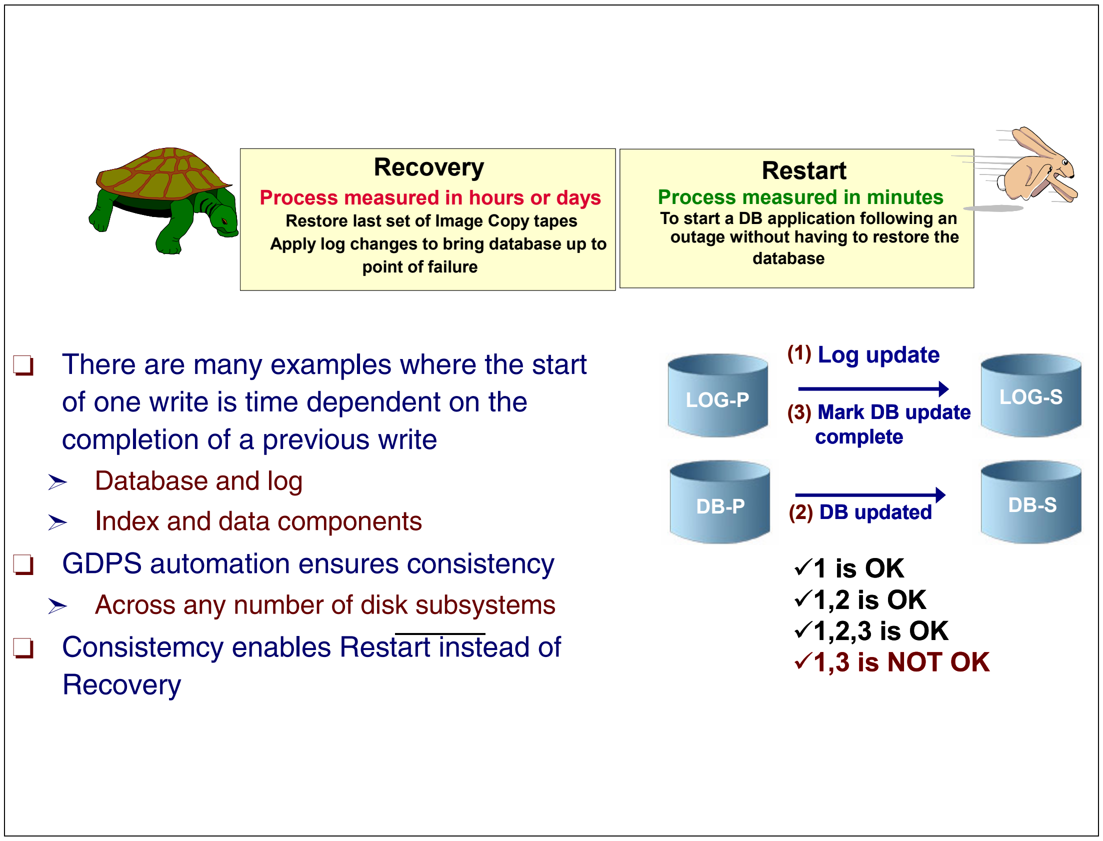
Figure 7-14 Main GDPS/PPRC 3270-based panel

GDPS operational interfaces

GDPS provides operation interfaces such as the following:

- DASD remote copy panels
  - Provide information about DASD remote copy status
  - Provide a manual operation interface for the remote copy DASD configuration
- Tape remote copy panels
  - Provide manual control of the GDPS peer-to-peer virtual tape server (PtPVTS)
- GDPS action panels with main operator interface for starting, stopping, and restarting systems.
- GDPS sysplex resource management panels to control CF structures and couple data sets.

7.14 Need for data consistency



Recovery
Process measured in hours or days
Restore last set of Image Copy tapes
Apply log changes to bring database up to point of failure

Restart
Process measured in minutes
To start a DB application following an outage without having to restore the database

- ❑ There are many examples where the start of one write is time dependent on the completion of a previous write
  - ➢ Database and log
  - ➢ Index and data components
- ❑ GDPS automation ensures consistency
  - ➢ Across any number of disk subsystems
- ❑ Consistemcy enables Restart instead of Recovery

Figure 7-15 Need for data consistency

Need for data consistency

Data consistency across all primary and secondary volumes spread across any number of storage subsystems is essential for providing the ability to keep data integrity and allowing a normal database restart in the event of a disaster.

Dependent write logic

Database applications commonly ensure the consistency of their data by using dependent write logic irrespective of whether remote data mirroring is being used. Dependent write logic states that if I/O B must logically follow I/O A, then B will not be started until A completes successfully. This logic would normally be included in all software to manage data consistency. There are numerous instances within the software subsystem itself, such as databases, catalog/VTOC, and VSAM file updates, where dependent writes are issued.

It is unlikely that all the components in a data center would fail at the same instant – even in the rare case of a full data center outage. The networks may fail first, or possibly one disk subsystem, or any other components in unpredictable combinations. No matter what happens, the remote image of the data must be managed such that cross-volume and subsystem data consistency is preserved during intermittent and staged failures that might occur over many seconds, even minutes. Such a staged failure is generally referred to as a “rolling disaster.”

Data consistency

Data consistency during a rolling disaster is difficult to achieve for synchronous forms of remote copy because synchronous remote copy is entirely implemented within disk subsystem pairs. Data consistency means that, from an application's perspective, the secondary disks contain all updates until a specific point in time, and no updates beyond that specific point in time; in other words, the time sequence of the writes was respected when remotely copied. In a disaster recovery it is possible to loose some data updates in the secondary copy but it is not acceptable to not have data consistency.

For example, in Figure 7-15 on page 311 the synchronously mirrored data sets are spread across multiple disk subsystems for optimal performance. The volume containing the DBMS log on the LOG-P disk subsystem in Site1 is mirrored to the secondary volume in the LOG-S disk subsystem in Site2, and the volume containing the data segments in the DB-P disk subsystem in Site1 is mirrored to the secondary volume in the DB-S disk subsystem in Site2. Assume that a disaster is in progress in Site1, causing the link between DB-P and DB-S to be lost before the link between LOG-P and LOG-S is lost. With the link between DB-P and DB-S lost, a write sequence of (1), (2), and (3) might be completed on the primary devices (depending on how the remote copy pair was defined) and the LOG writes (1) and (3) would be mirrored to the LOG-S device, but the DB write (2) would not have been mirrored to DB-S. A subsequent DBMS restart using the secondary copy of data in Site2 would clean up in-flight transactions and resolve in-doubt transactions, but the missing DB write (2) would not be detected. In this example of the missing DB write the DBMS integrity was compromised.

The fact that the secondary copy of the data is data consistent means that applications can be restarted in the secondary location without having to go through a lengthy and time-consuming data recovery process, that implies backups being restored.

GDPS automation

The main focus of GDPS automation is to make sure that, whatever happens in Site 1, the secondary copy of the data in Site 2 is *data consistent* (the primary copy of data in Site 1 will be data consistent for any Site 2 failure).

GDPS/PPRC

GDPS/PPRC uses a combination of storage subsystem and Parallel Sysplex technology triggers to capture a data consistent secondary site copy of the data, using the PPRC freeze function at the first indication of a potential disaster in the primary site.

The PPRC Critical attribute has two values: Yes and No. Yes means that if there is an error along PPRC mirroring (in the link or in the secondary controller) the application executing the write receives back a status of unrecoverable I/O error in the write operation. For NO, the primary controller generates an extended long busy interrupt to be processed just by IOS. See "An example of write dependency" on page 313 to understand better this action.

The GDPS freeze function, initiated by automated procedures, is designed to freeze the image of the secondary data at the very first sign of a disaster in the primary. This prevents the logical contamination of the secondary copy of data that would occur if any storage subsystem mirroring were to continue after a failure that prevents some, but not all, secondary volumes from being updated.

Data consistency in a GDPS/XRC environment is provided by the Consistency Group (CG) processing performed by the System Data Mover (SDM). The CG contains records that have their order of update preserved across multiple Logical Control Units within a storage subsystem and across multiple storage subsystems.

Providing data consistency optimizes the secondary copy of data to perform normal restarts (instead of performing database manager recovery actions).

7.15 An example of write dependency

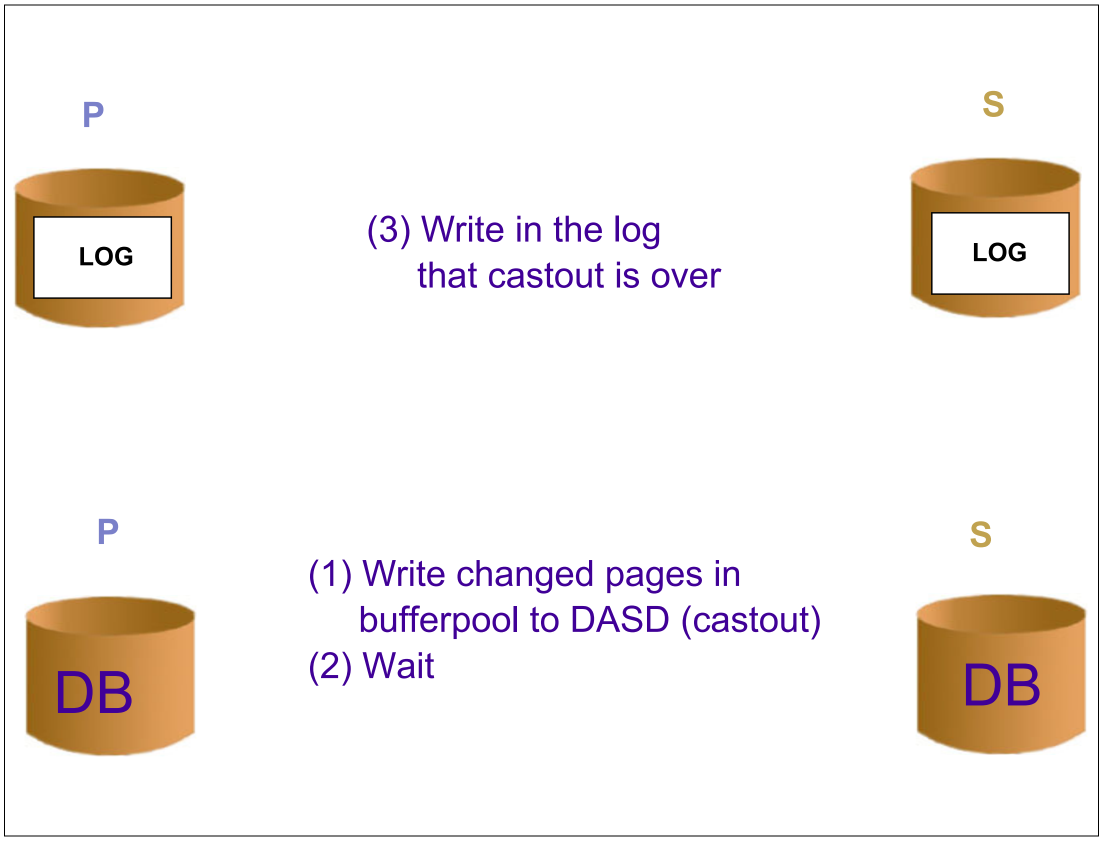

P
S
LOG
LOG
(3) Write in the log that castout is over
P
S
DB
DB
(1) Write changed pages in bufferpool to DASD (castout)
(2) Wait

Figure 7-16 An example of write dependency

An example of write dependency

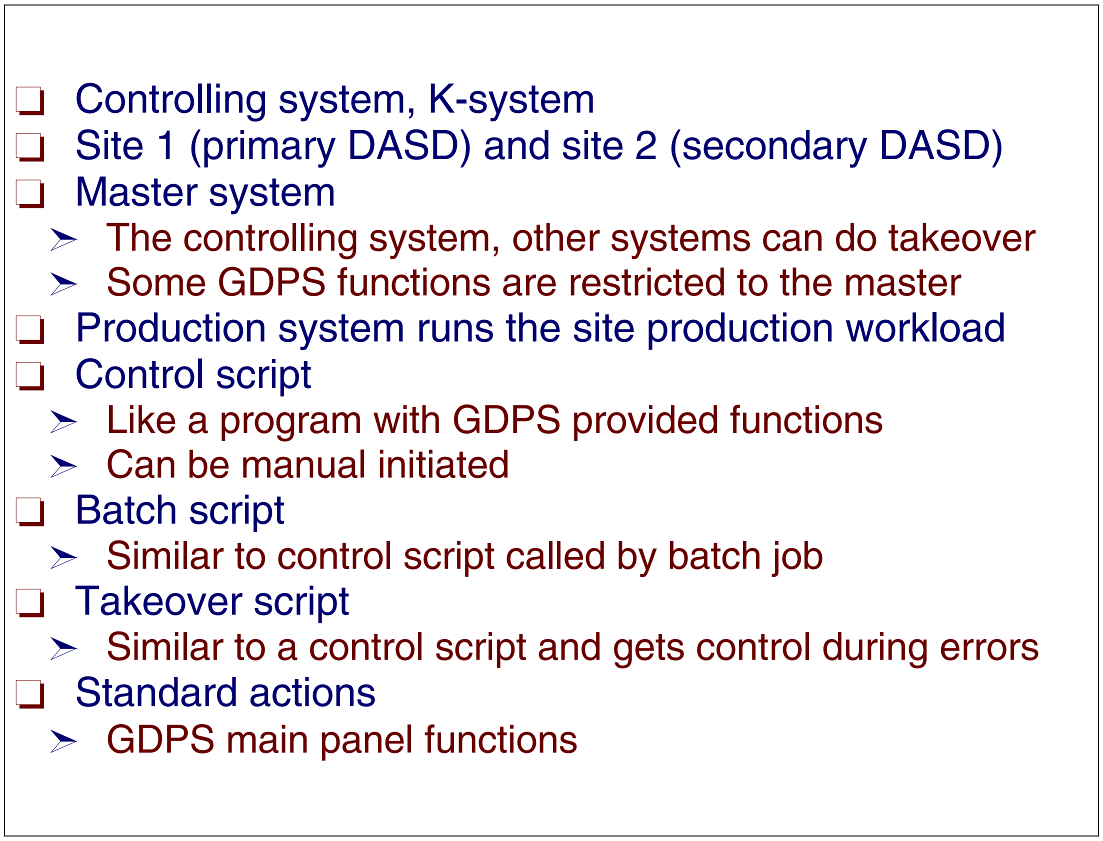
Let us imagine a DB2 scenario where the log and the table spaces are in 3390s located in different physical controllers, as usually recommended. It is time to write the changed pages in the local buffer pool (cast out) to DASD. These 3390s are PPRC remote copied. Another key point is that DB2 is not aware about such copying.

First, DB2 writes pages to the table space in DASD; then enters in wait (to guarantee that the last write was executed); next it writes to the log informing that the cast out was complete. With this procedure (including the wait) DB2 forces the time sequence of writes in primary 3390s.

The last write to the DB and the write to log are time dependent writes and they must be executed exactly in their time sequence. We loose data consistency and consequently integrity if these writes are written out of the time sequence, because we will have an interval where the log is telling that the cast out was complete and it was not.

The challenge with remote copy is to guarantee data consistency in the *secondary* copy; the DB2 Wait function guarantees it in the *primary*. This consistency is called secondary consistency. It implies that the time write sequence is respected in the secondary copies. It implies that either both writes are copied, or just the final table space write is copied, or none are copied. Never will we have a case where (even for a short amount of time) just the write to log was copied. There are two ways to guarantee secondary consistency with PPRC: PPRC Critical= No or the GDPS freezing function.

7.16 GDPS terminology



- ❑ Controlling system, K-system
- ❑ Site 1 (primary DASD) and site 2 (secondary DASD)
- ❑ Master system
  - ➢ The controlling system, other systems can do takeover
  - ➢ Some GDPS functions are restricted to the master
- ❑ Production system runs the site production workload
- ❑ Control script
  - ➢ Like a program with GDPS provided functions
  - ➢ Can be manual initiated
- ❑ Batch script
  - ➢ Similar to control script called by batch job
- ❑ Takeover script
  - ➢ Similar to a control script and gets control during errors
- ❑ Standard actions
  - ➢ GDPS main panel functions

Figure 7-17 GDPS terminology

Controlling system

The controlling system, also known as the K-system, is a system that is in the same sysplex as the production systems, thus, it can see all the messages from those systems that are being managed by GDPS. However, its primary role is to be able to control the recovery following an outage. For this reason, the controlling system resides and has all its DASD in the recovery site, Site 2, and it shares only the couple data sets (CDSs) with other systems in the sysplex.This ensures that the controlling system will not be affected by any events that might impact the availability of the managed systems.

Site 1 and Site 2

In a GDPS/PPRC environment, the site that contains the primary DASD during normal operations is known as Site 1. The site that contains the secondary DASD during normal operations is known as Site 2. The terms Site 1 and Site 2 are used when defining the PPRC DASD to GDPS. If there is a disaster, or a planned site switch, the DASD in Site 2 would become the primary DASD.

Master system

Normally the controlling system is the master system. However, if the controlling system is unavailable for some reason, one of the other systems takes over as the master system. Some GDPS functions (configuration processing, takeover processing, and control script execution) can only be carried out by the master system.

Production system

A production system is a system that normally runs the site's production workload and updates the primary DASD.

Control script

A control script is like a program, consisting of GDPS-provided functions, that can be manually initiated. Control scripts can do things like IPLing a system, stopping the use of a coupling facility, and so forth.

Batch script

A batch script is similar in concept to a control script. However, while a control script is invoked from the GDPS panels, a batch script is initiated by calling a GDPS-provided routine, either from a batch job, or as a result of messages being issued.

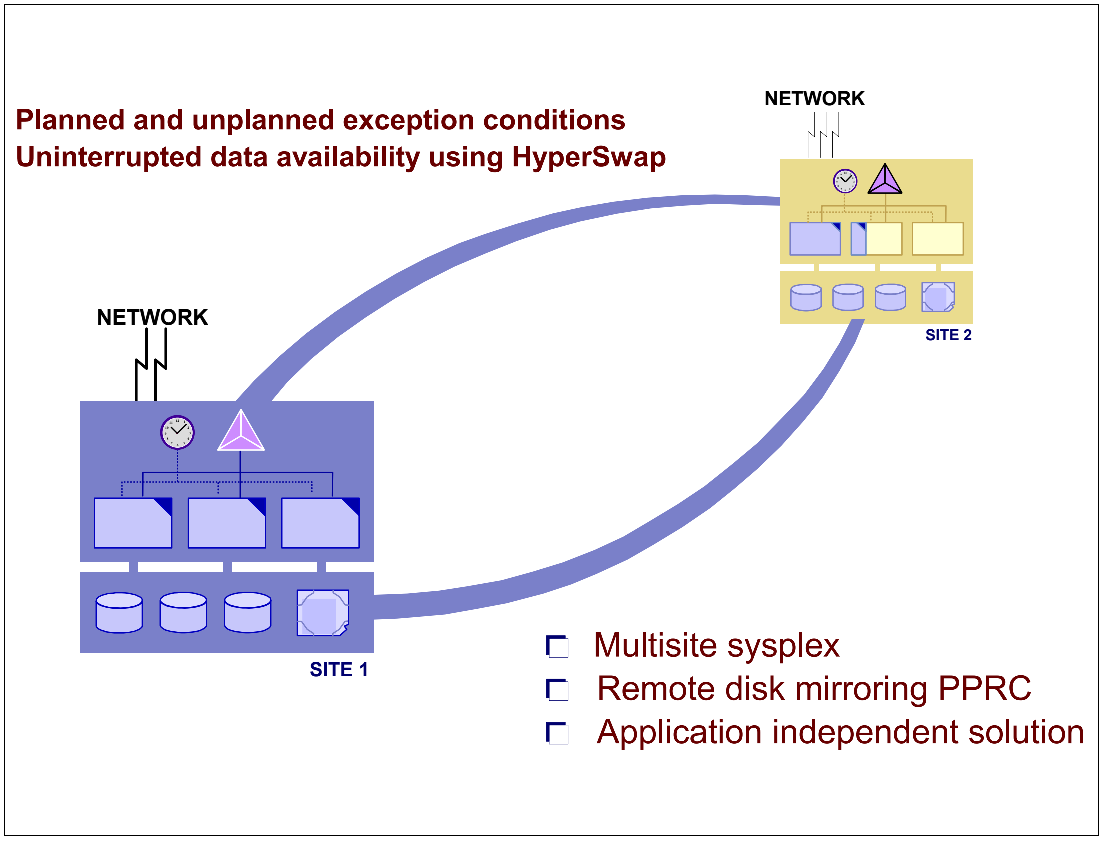
Takeover script

A takeover script is also like a program, again consisting of GDPS-provided functions. Unlike a control script, however, a takeover script can only be initiated following an error that results in a possible takeover situation (for example, a DASD failure or a system failure). Takeover scripts are initiated as a result of messages being issued.

Standard actions

Standard actions are a set of functions that can be performed for z/OS images and LPs using GDPS standard actions, on the GDPS main panel. The standard actions panels should be the main interface for the operator for routine functions such as starting, stopping, and restarting systems when you have GDPS in your production environment.

7.17 GDPS/PPRC



Planned and unplanned exception conditions
Uninterrupted data availability using HyperSwap

- Multisite sysplex
- Remote disk mirroring PPRC
- Application independent solution

Figure 7-18 GDPS/PPRC

GDPS/PPRC

GDPS/PPRC is one of the GDPS offerings that manages and protects IT services by handling planned and unplanned exception conditions, and maintaining full data integrity across multiple volumes and storage subsystems. By managing both planned and unplanned exception conditions, GDPS/PPRC maximizes application availability and provides business continuity.

Topology

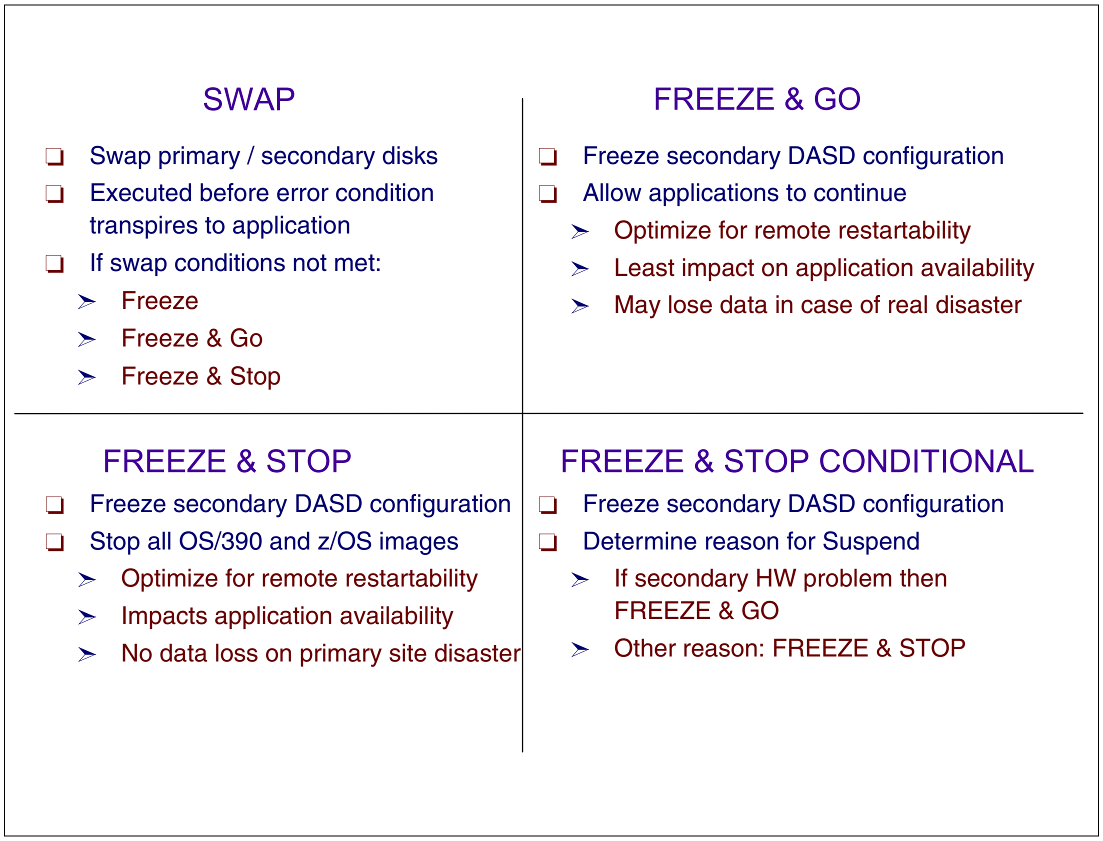
The physical topology of a GDPS/PPRC is shown in Figure 7-18.

GDPS/PPRC consists of a base or Parallel Sysplex cluster spread across two sites (known as Site 1 and Site 2), with one or more z/OS systems at each site.

The multisided Parallel Sysplex cluster must be configured with redundant hardware (for example, a Coupling Facility and a Sysplex Timer, or an STP server, in each site), and the cross-site connections must be redundant.

All critical data resides on storage subsystems in Site 1 (the primary copy of data) and is mirrored to the storage subsystems in Site 2 (the secondary copy of data) via PPRC synchronous remote copy.

7.18 Freeze policy options



SWAP

- ❑ Swap primary / secondary disks
- ❑ Executed before error condition transpires to application
- ❑ If swap conditions not met:
  - ➢ Freeze
  - ➢ Freeze & Go
  - ➢ Freeze & Stop

FREEZE & GO

- ❑ Freeze secondary DASD configuration
- ❑ Allow applications to continue
  - ➢ Optimize for remote restartability
  - ➢ Least impact on application availability
  - ➢ May lose data in case of real disaster

FREEZE & STOP

- ❑ Freeze secondary DASD configuration
- ❑ Stop all OS/390 and z/OS images
  - ➢ Optimize for remote restartability
  - ➢ Impacts application availability
  - ➢ No data loss on primary site disaster

FREEZE & STOP CONDITIONAL

- ❑ Freeze secondary DASD configuration
- ❑ Determine reason for Suspend
  - ➢ If secondary HW problem then FREEZE & GO
  - ➢ Other reason: FREEZE & STOP

Figure 7-19 Freeze policy options

Freeze policy options

All the following explanations apply when the ESS/DS8000 PPRC option Critical is equal to NO. When Critical=YES, the application issuing the write (this could be DB2) receives back an unrecoverable I/O error status when there is any problem with the PPRC mirroring. With such information the application does not issue the next dependent write operation and then the secondary consistency is guaranteed. However, all the transactions reaching this primary data will abend.

With Critical=NO, the ESS/DS8000 controller generates the extended long busy (ELB) interrupt when the remote copy is not possible. (Maybe the problem in the link is the start of a rolling disaster.) IOS in z/OS catches the interrupt and issues a message in console, that is caught by the GDPS automation process.

When the remote copy in a primary volume is suspended (with Critical=NO) because of an error condition in the remote copy, the GDPS automation routines immediately freeze the image of all the required secondary volumes, applying the CGROUP FREEZE command to all application-related LSSs that are known to GDPS. This preserves the consistency of the recovery data at that specified point in time. Imagine that the DB2 table space write remote copy failed; then the log remote copy must be frozen to guarantee the secondary consistency when DB2 writes to the log. In this case, the primary DB and log are updated and the secondary DB and log are not.

After freezing, GDPS does an analysis step. The analysis is driven by a predetermined installation policy, which can be either FREEZE and GO, or FREEZE and STOP, or FREEZE and STOP CONDITIONAL. They are described in the following paragraphs.

FREEZE and GO

With this option the automation routines allow applications to continue after the secondary volumes have been frozen. The implication is that updates to primary volumes after the freeze will be lost if a disaster occurs, and thus applications will have to be restarted in the recovery site. The other implication is that the availability of the application is favored, as it is not stopped by this error circumstance.

FREEZE and STOP

With this option the automation routines stop all applications right where they are at the time of freezing the secondary volumes. Application availability will be impacted, but the currency of the recovery site data is preserved.

FREEZE and STOP CONDITIONAL

With this option the automation routines freeze the image of the secondary volumes, and then analyze the reason for the suspend condition. If it turns out to be nonthreatening, for instance, because a secondary piece of equipment has failed, then the applications will be allowed to continue. However, if there is any indication of a serious problem, the automation routines will then stop all the required applications.

There is a difference between using CRIT=YES or Freeze automation, as described here.

If you have CRIT=YES option

The sequence of events is:

- The controller returns a unit check (unrecoverable I/O error) to the DB2.
- DB2 enters on the log the fact that this table space on the first 3390 requires recovery.
- Since the primary 3390 containing the log does not appear to have a mirroring problem, this log information gets mirrored to the secondary.
- Now, the primary site fails, so GDPS recovers the secondary 3390s and restarts your systems using these disks
- DB2 will force table space recovery on this table (and probably many others) because of the information in the log.

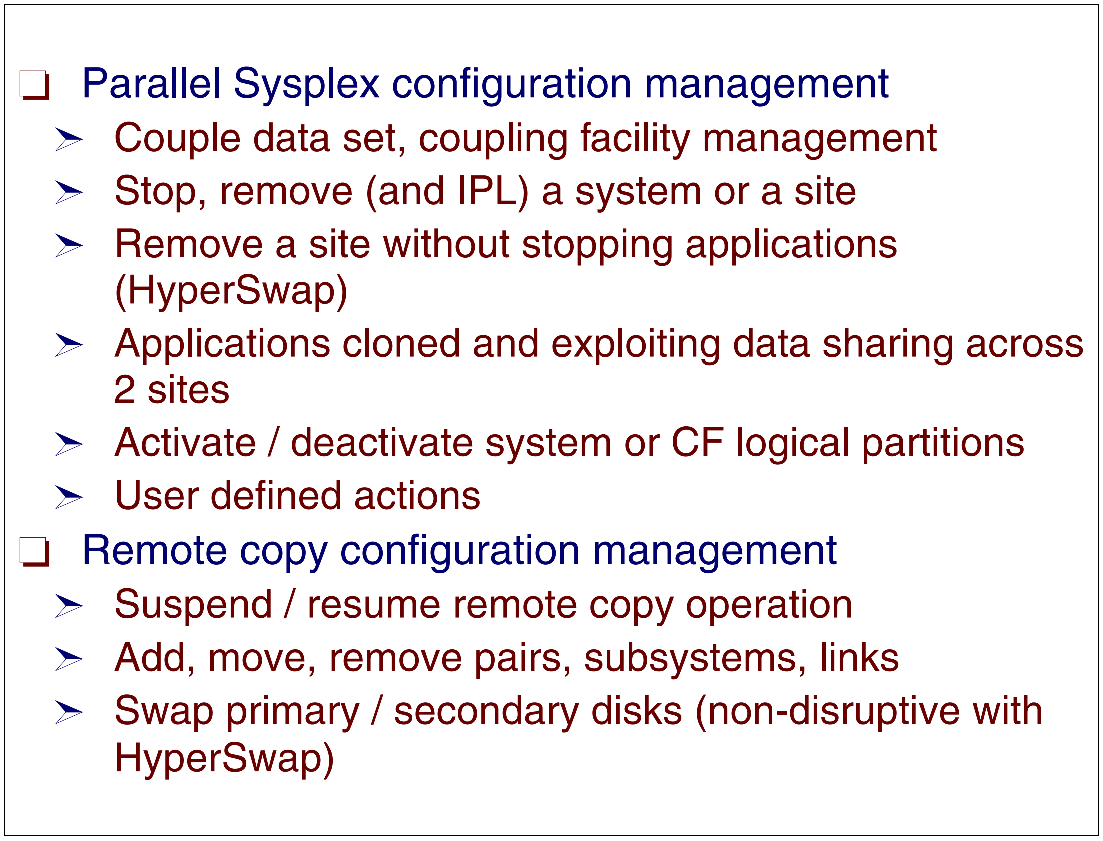
If you have Freeze automation

The sequence of events is:

- The controller returns an extended long busy (ELB) status to z/OS and issues a console message.
- GDPS automation captures this message and issues the `CGROUP FREEZE` command against all PPRC SSIDs; CGROUP FREEZE severs all PPRC links.
- The primary site comes tumbling down so GDPS recovers the secondary 3390s and restarts your systems using these disks.
- There is no log information about I/O errors and the restart is very simple.

So as you can see, with CGROUP FREEZE we do not have to run any lengthy forward recovery, whereas with CRIT=Y we do.

7.19 Planned reconfiguration support



- ❑ Parallel Sysplex configuration management
  - ➢ Couple data set, coupling facility management
  - ➢ Stop, remove (and IPL) a system or a site
  - ➢ Remove a site without stopping applications (HyperSwap)
  - ➢ Applications cloned and exploiting data sharing across 2 sites
  - ➢ Activate / deactivate system or CF logical partitions
  - ➢ User defined actions
- ❑ Remote copy configuration management
  - ➢ Suspend / resume remote copy operation
  - ➢ Add, move, remove pairs, subsystems, links
  - ➢ Swap primary / secondary disks (non-disruptive with HyperSwap)

Figure 7-20 Planned reconfiguration support

Planned reconfiguration support

GDPS/PPRC planned reconfiguration support automates procedures performed by an operations center. In addition to the disaster recovery and planned reconfiguration capabilities, GDPS/PPRC also provides a much more user-friendly interface for monitoring and managing the remote copy configuration. These include standard actions to do the tasks described in this section.

Parallel Sysplex configuration management

Coupling links are required in a Parallel Sysplex configuration to provide connectivity from the z/OS images to the Coupling Facility. Coupling links are also used to transmit timekeeping messages when Server Time Protocol (STP) is enabled. If you have a multi-site Parallel Sysplex, you will need to provide coupling link connectivity between sites.

These are the recovery actions related to Parallel Sysplex:

- ► Recreate the couple data sets or change the policies.
- ► Quiesce a system’s workload and remove the system from the Parallel Sysplex cluster (for example, stop the system prior to a change window).
- ► Remove a site without stopping applications by using the HyperSwap® functionality. The applications must be cloned and exploiting data sharing across two sites, and the network must be switchable if the applications use it.
- ► IPL a system (for example, start the system after a change window).

- Quiesce a system's workload, remove the system from the Parallel Sysplex cluster, and re-IPL the system (for example, recycle a system to pick up software maintenance).

Remote copy configuration management

These are the recovery actions related to remote copy, such as Initialize and monitor a remote copy volume pair based upon policy, and perform routine operations on installed storage subsystems, both disk and tape.

Standard actions can be initiated against a single system or a group of systems.

Additionally, GDPS/PPRC provides customized scripting capability for user-defined actions (for example, planned site switch in which the workload is switched from processors in Site 1 to processors in Site 2).

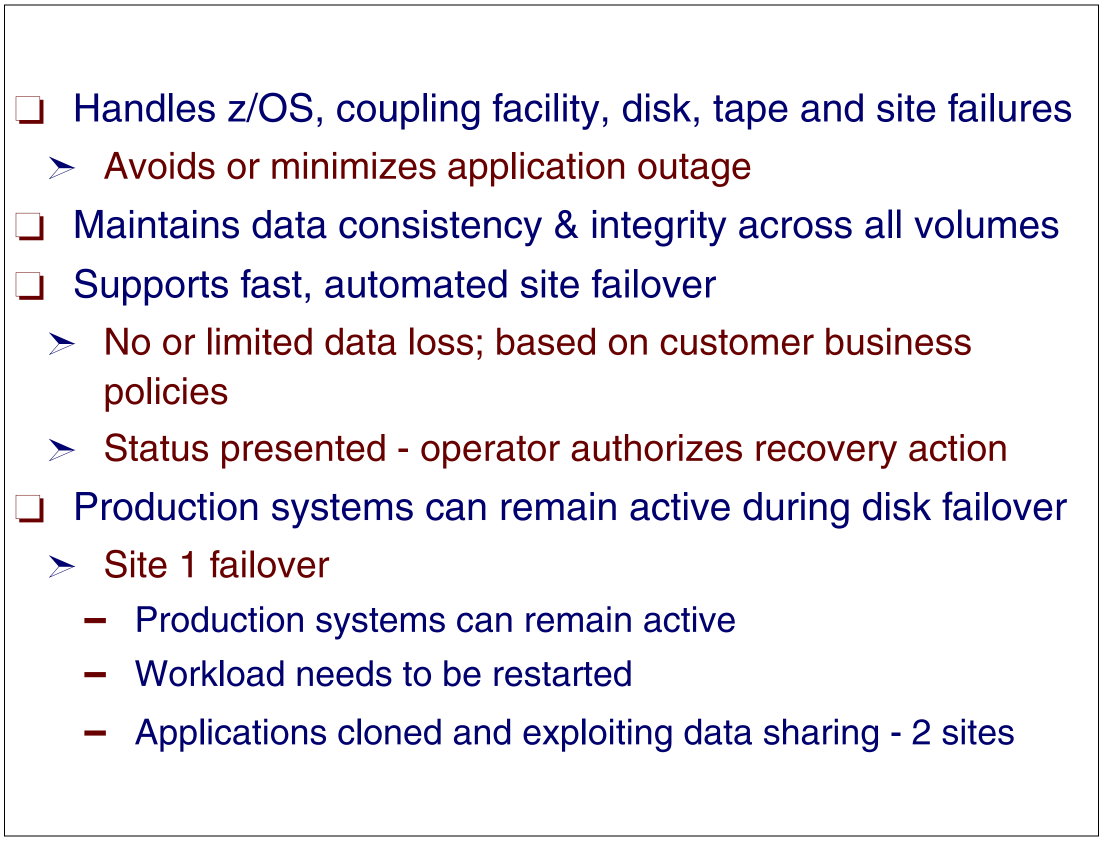
In order for GDPS to manage the remote copy environment, you must first define the configuration (primary and secondary LSSs, primary and secondary devices, and PPRC links) to GDPS in a file called the GEOPARM file. After the configuration is known to GDPS, you can use the panels to check that the current configuration matches the desired one. You can start, stop, suspend, and resynchronize mirroring at the volume or LSS level. You can initiate a FlashCopy® and you can reverse the direction of mirroring. These actions can be carried out at the device or LSS level, or both, as appropriate.

HyperSwap function

GDPS/PPRC delivers a powerful function known as "HyperSwap." HyperSwap provides the ability to non-disruptively switch from using the primary volume of a mirrored pair to using what had been the secondary volume. Prior to the availability of HyperSwap, an IPL was required on every system if you wanted to switch and run from the secondary volumes, meaning that it was not possible to maintain application availability across a switch from primary to secondary volumes.

With HyperSwap, such a move can be accomplished without IPL and with just a brief hold on application I/O. The HyperSwap function is designed to be completely controlled by automation, thus allowing all aspects of the site switch to be controlled via GDPS.

7.20 Unplanned reconfiguration support



- ❑ Handles z/OS, coupling facility, disk, tape and site failures
  - ➢ Avoids or minimizes application outage
- ❑ Maintains data consistency & integrity across all volumes
- ❑ Supports fast, automated site failover
  - ➢ No or limited data loss; based on customer business policies
  - ➢ Status presented - operator authorizes recovery action
- ❑ Production systems can remain active during disk failover
  - ➢ Site 1 failover
    - – Production systems can remain active
    - – Workload needs to be restarted
    - – Applications cloned and exploiting data sharing - 2 sites

Figure 7-21 Unplanned reconfiguration support

Unplanned reconfiguration support

GDPS/PPRC unplanned reconfiguration support not only automates procedures to handle site failures, but can also minimize the impact and potentially mask a z/OS system, processor, Coupling Facility, disk or tape failure, based upon GDPS/PPRC policy.

If z/OS fails, the failed system and workload can be automatically restarted.

Server failures

If a server fails, the failed systems and their workload can be restarted on other servers. If the primary disk storage subsystem fails, a disk reconfiguration will allow access of the secondary PPRC volumes, which contain mirrored data consistent with the primary data. The following tasks can be executed:

- Present actual status to operator
- Request failover authorization
- Remove systems from Parallel Sysplex
- Perform disk reconfiguration
- Perform tape reconfiguration
- Perform CBU activation
- Perform CF reconfiguration
- Perform CDS reconfiguration
- Acquire processing resources and IPL systems into Parallel Sysplex
- Initiate application startup

Disk failover

When a primary disk failure occurs and the disks are switched to the secondary devices, PPRC Failover/Failback (FO/FB) support eliminates the need to do a full copy when reestablishing replication in the opposite direction. Because the primary and secondary volumes are often in the same state when the freeze occurred, the only differences between the volumes are the updates that occur to the secondary devices after the switch. Failover processing sets the secondary devices to primary suspended status and starts change recording for any subsequent changes made. When the mirror is re-established with failback processing, the original primary devices become secondary devices and a resynchronization of changed tracks takes place.

Note: All disk subsystems in your GDPS configuration, in both Site1 and Site2, must support PPRC Failover/Failback for GDPS to exploit this capability.

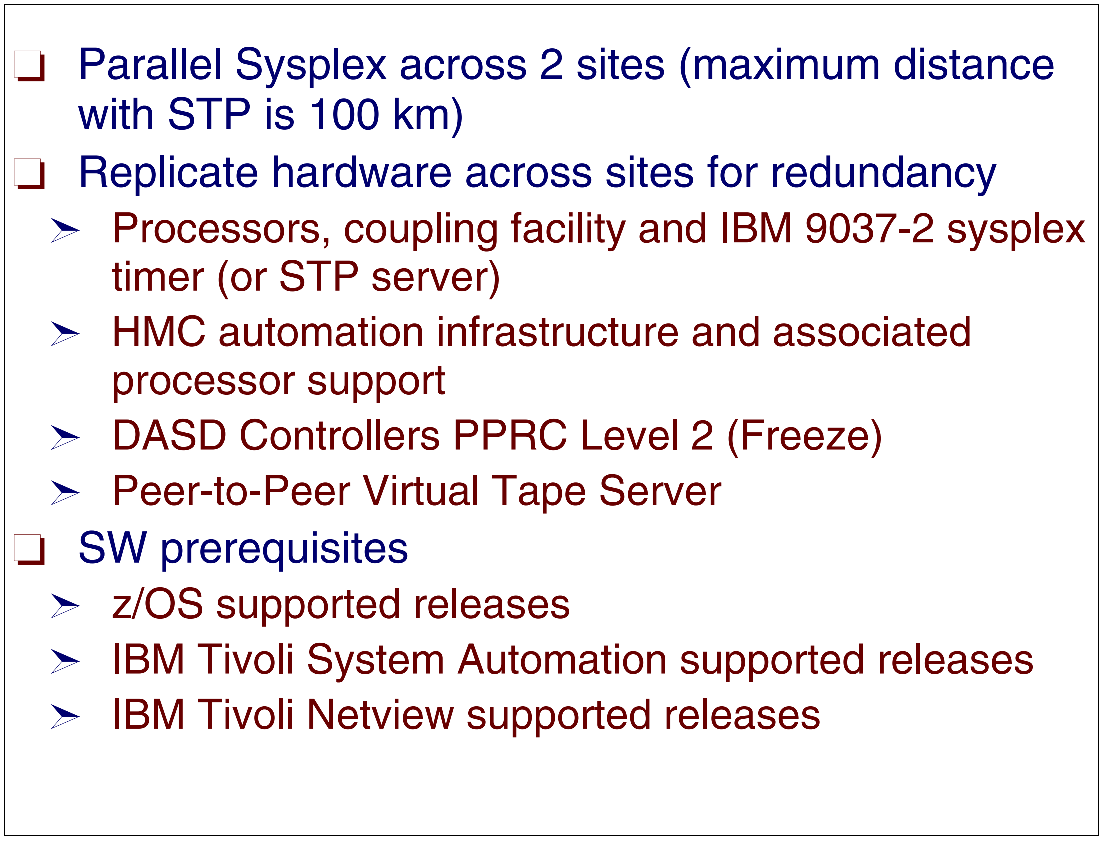
Site 1 failover

An existing GDPS script that previously performed failover and restart of System z resources can be extended to also include statements to automate the failover of SA AppMan clusters to Site2. When the additional script statements are executed as part of the site takeover script, GDPS automation performs the following actions to move System z resources from Site1 to Site2:

- It resets production systems in Site1.
- It reconfigures the secondary disks.
- It activates CBU for servers in Site2.
- It switches couple data sets to those in Site2.
- It activates partitions in Site2.
- It reIPLs P1, P2, and P3 in Site2.

Note: GDPS/GM V3.6 has been enhanced to provide DCM support for VCS clusters. Distributed Cluster Management (DCM) is a GDPS capability introduced in GDPS V3.5 which allows the management and coordination of planned and unplanned outages across distributed servers which might be clustered using clustering solutions, and the System z workloads that GDPS is responsible for. DCM support was initially introduced in GDPS/PPRC V3.5 and GDPS/XRC V3.5. The support in GDPS V3.5 was for distributed clusters managed by Veritas Cluster Server (VCS) and IBM Tivoli System Automation Application Manager (SA AppMan).

7.21 GDPS/PPRC prerequisites



- ❑ Parallel Sysplex across 2 sites (maximum distance with STP is 100 km)
- ❑ Replicate hardware across sites for redundancy
  - ➢ Processors, coupling facility and IBM 9037-2 sysplex timer (or STP server)
  - ➢ HMC automation infrastructure and associated processor support
  - ➢ DASD Controllers PPRC Level 2 (Freeze)
  - ➢ Peer-to-Peer Virtual Tape Server
- ❑ SW prerequisites
  - ➢ z/OS supported releases
  - ➢ IBM Tivoli System Automation supported releases
  - ➢ IBM Tivoli Netview supported releases

Figure 7-22 GDPS/PPRC prerequisites

GDPS/PPRC prerequisites

The prerequisites for GDPS/PPRC are described in this section.

Parallel Sysplex across two sites

PPRC is the IBM synchronous remote copy technology. When an application issues a write to a PPRC device, the I/O request does not complete until the write request has been successfully sent to the remote (secondary) control unit. PPRC is a HW-based solution that consists of a primary control unit connected to a secondary in different sites. Site 1 and 2 (this is GDPS site terminology) must have z/OSs in the same Parallel Sysplex (or Base Sysplex). This requirement limits the distance for PPRC to about 100 km. However, at such distance the PPRC performance will adversely affect applications.

Replicate hardware across sites for redundancy

As a minimum GDPS requirement, you must define a cross-site Parallel Sysplex, which means the HW must be replicated on Site 1 and Site 2. The following HW should be available or is optional depending on the environment:

- ► CPCs, coupling facilities
- ► Sysplex Timer or STP servers
- ► HMC automation infrastructure
- ► Disk subsystems supporting PPRC level 2 freeze function

- Peer-to-peer virtual tape server
- Dense wavelength division multiplexors (DWDMs)

Software prerequisites

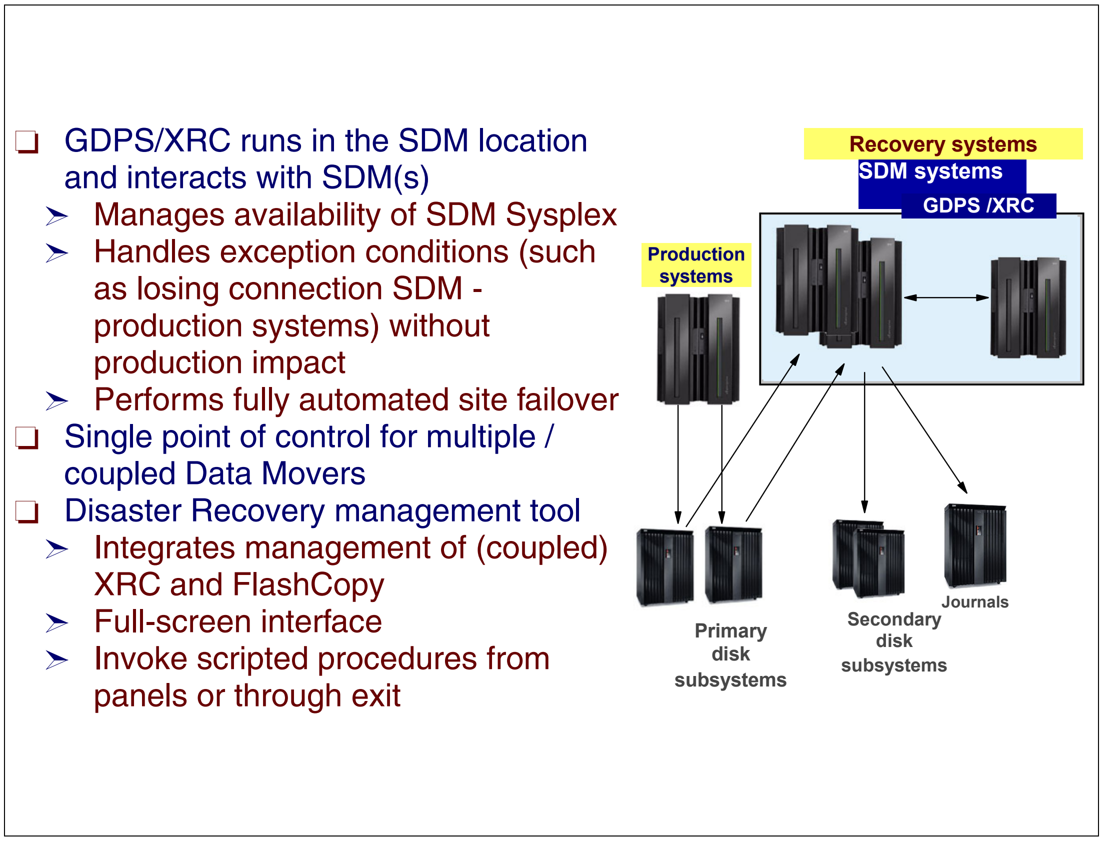
Following are the software prerequisites:

- Installation Services Prerequisites for GDPS/PPRC
  - Generally supported release of z/OS or z/OS.e
  - Generally supported release of IBM Tivoli Netview
  - Generally supported release of IBM Tivoli System Automation for z/OS
  - Storage Subsystems with PPRC CGROUP Freeze/Run support
  - Multisite Base or Parallel Sysplex (1)

Note: Certain GDPS configurations may not require a cross-site sysplex.

- Optional - Prerequisites for GDPS / PPRC Multiplatform Resiliency for System z support:
  - Generally supported release of z/VM
  - Linux on System z
  - Generally supported release of IBM Tivoli System Automation for Multiplatforms
  - IBM Tivoli System Automation for Multiplatforms XDR for Linux on System z
- Optional - Prerequisites for GDPS FlashCopy support:
  - Storage subsystems with FlashCopy support
- Optional - Prerequisites for GDPS Open LUN support:
  - IBM Tivoli Netview for z/OS V5.1 (or later)
  - Storage subsystems with support for:
    - Open PPRC Management via CKD device addresses
    - Open PPRC SNMP alerts
  - CKD utility device (that is, CKD LSS) on the disk subsystems that house the distributed LUNs.
- Optional - Prerequisites for GDPS HyperSwap:
  - Storage subsystems with Extended CQUERY support
  - PPRC Failover/Failback support required for additional HyperSwap performance for planned swaps and to avoid initial copy following an unplanned swap
- Optional - Prerequisites for GDPS DCM for Veritas Cluster Server:
  - Veritas Cluster Server HA/DR 5.0 package or later
  - Tivoli NetView for z/OS V5.2 or later
  - Distributed server hardware, the operating system release and service levels supported by Veritas Cluster Server
- Currently supported releases of Tivoli products can be found in:
  - Tivoli product lifecycle dates
- Currently supported releases of z/OS and z/OS.e can be found in:
  - z/OS support

7.22 GDPS/XRC



- ❑ GDPS/XRC runs in the SDM location and interacts with SDM(s)
  - ➢ Manages availability of SDM Sysplex
  - ➢ Handles exception conditions (such as losing connection SDM - production systems) without production impact
  - ➢ Performs fully automated site failover
- ❑ Single point of control for multiple / coupled Data Movers
- ❑ Disaster Recovery management tool
  - ➢ Integrates management of (coupled) XRC and FlashCopy
  - ➢ Full-screen interface
  - ➢ Invoke scripted procedures from panels or through exit

Figure 7-23 GDPS/XRC

GDPS/XRC

Extended Remote Copy (XRC), rebranded to IBM System Storage® z/OS Global Mirror, is a combined hardware and software asynchronous remote copy solution. Consistency of the data is maintained via the Consistency Group function within the z/OS System Data Mover (SDM).

Because of the asynchronous nature of XRC, it is possible to have the secondary disk at greater distances than would be acceptable for PPRC. Channel extender technology can be used to place the secondary disk up to thousands of kilometers away. Because XRC is asynchronous, the impact it has on response times is minimal, and is independent of the distance between the primary and secondary volumes.

GDPS/XRC combines the benefits of GDPS with the extended distance capabilities of XRC. It includes automation to manage remote copy pairs and automates the process of recovering the production environment with limited manual intervention, including invocation of CBU1, thus providing significant value in reducing the duration of the recovery window and requiring less operator interaction.

GDPS offering

GDPS/XRC is a GDPS offering that provides the automation for planned and unplanned reconfiguration support, the primary attribute of most D/R solutions. The physical topology of a GDPS/XRC (see Figure 7-23) consists of production systems in Site 1 (the production systems could be a single system, multiple systems sharing disks, or a base or Parallel

Sysplex cluster). Note that the Parallel Sysplex cluster does not span Sites 1 and 2. Site 2 (the recovery site) can be located at virtually any distance from Site 1 (the production site). During normal operations, the XRC System Data Mover (one or more) will be executing in Site 2 and is in a base sysplex with the GDPS controlling system (refer to "GDPS terminology" on page 314 for a definition of the GDPS controlling system).

All critical data resides on storage subsystems in Site 1 (the primary copy of data) and is mirrored to the storage subsystems in Site 2 (the secondary copy of data) via XRC asynchronous remote copy.

SDM in recovery site

The following planned reconfiguration actions are provided by GPDPS/XRC for the SDM in the recovery site:

1. Quiesce a system's workload and remove the system from the Parallel Sysplex cluster (for example, stop the system prior to a change window).
2. IPL a system (for example, start the system after a change window).
3. Quiesce a system's workload, remove the system from the Parallel Sysplex cluster, and re-IPL the system (for example, recycle a system to pick up software maintenance).

Planned Actions with XRC and FlashCopy

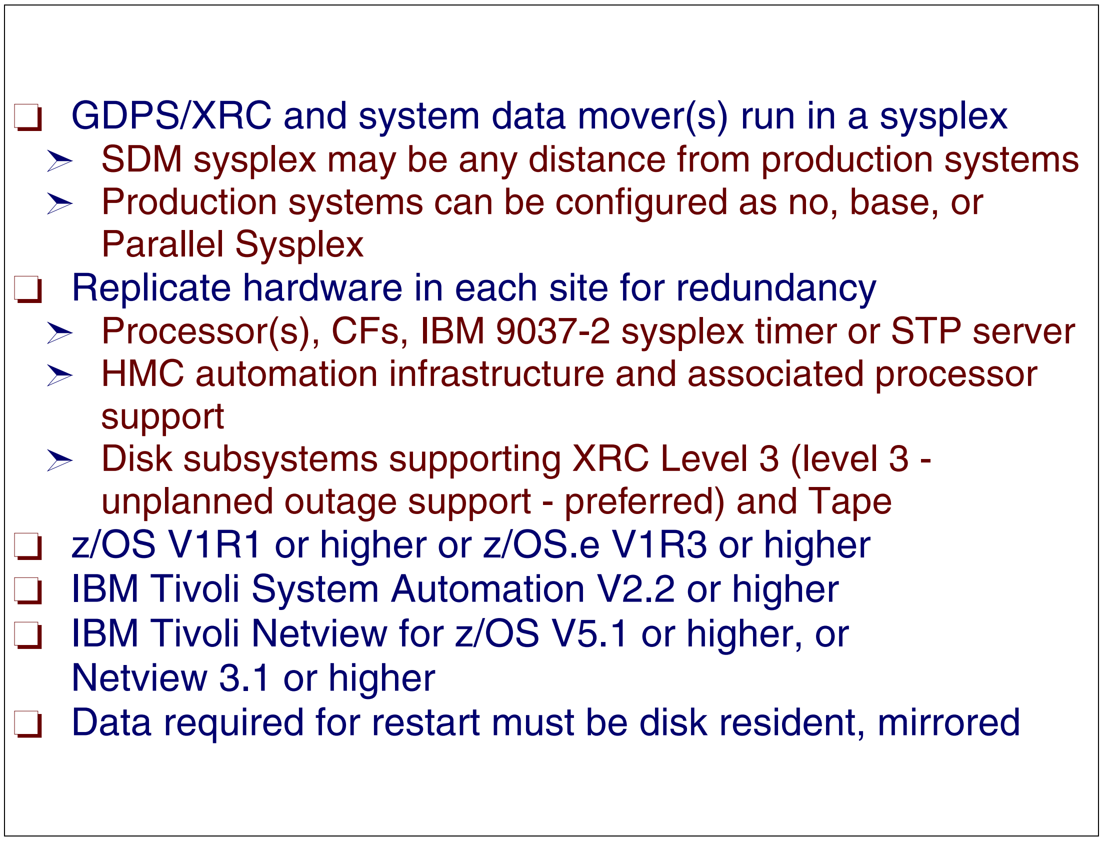
Planned Actions are GDPS scripts that are initiated from the GDPS panels (option 6 on the main GDPS panel, as shown in Figure 5-3 on page 106). A Planned Action script might consist of a number of tasks. For example, you could have a script that would stop an LPAR, change its IPL address to the alternate SYSRES, and then re-IPL it, all from a single script.

A more complex example of a Planned Action is when a single action in GDPS results in a tertiary copy of the secondary disks being taken, followed by IPLing the "production" systems in LPARs in the recovery site. This allows you to test your recovery procedures while XRC sessions between the primary to the secondary volumes are running, and you maintain up-to-date disaster readiness. Specifically, the following actions are carried out by GDPS in this script:

- Zero Suspend FlashCopy is initiated:
  - This prevents the SDMs from writing new consistency groups to the secondary disks.
  - A FlashCopy is taken of all XRC secondary devices, as well as the devices that house the XRC state, control, and journal data sets.
  - Zero Suspend FlashCopy completes and SDM processing resumes writing new consistency groups to the secondary disks.
- An XRC recover on the tertiary devices is run.
- The CBU capacity reserved PUs on CPCD are activated.
- Any test systems whose LPARs would be used for a production system in case of a disaster are deactivated.
- The CF LPARs and the LPARs that will contain the recovered production systems are activated, followed by an IPL of those systems.

So, the result of a single action on your behalf (initiating the Planned Action) is that you have created a copy of your production system that can be used for DR testing, you have brought up recovery versions of your production systems, and you have maintained disaster readiness by still having synchronization between the primary and secondary volumes.

7.23 GDPS/XRC prerequisites



- ❑ GDPS/XRC and system data mover(s) run in a sysplex
  - ➢ SDM sysplex may be any distance from production systems
  - ➢ Production systems can be configured as no, base, or Parallel Sysplex
- ❑ Replicate hardware in each site for redundancy
  - ➢ Processor(s), CFs, IBM 9037-2 sysplex timer or STP server
  - ➢ HMC automation infrastructure and associated processor support
  - ➢ Disk subsystems supporting XRC Level 3 (level 3 - unplanned outage support - preferred) and Tape
- ❑ z/OS V1R1 or higher or z/OS.e V1R3 or higher
- ❑ IBM Tivoli System Automation V2.2 or higher
- ❑ IBM Tivoli Netview for z/OS V5.1 or higher, or Netview 3.1 or higher
- ❑ Data required for restart must be disk resident, mirrored

Figure 7-24 GDPS/XRC prerequisites

GDPS/XRC prerequisites

Figure 7-24 shows the prerequisites for GDPS/XRC. The HW and SW products must be installed and customized prior the start of GDPS implementation. GDPS will run on every system of the recovery side, so it is vital that all systems adhere to these requirements. There is no need for HyperSwap with GDPS/XRC because the secondary volumes are online in the secondary site.

- Installation services prerequisites for GDPS/XRC:
  - Generally supported release of z/OS or z/OS.e
  - Generally supported release of IBM Tivoli Netview (Enterprise or Procedural level)
  - Generally supported release of IBM Tivoli System Automation for z/OS
  - Storage Subsystems (Primary) with XRC support
  - Base or Parallel Sysplex in recovery site between K-sys and SDMs
- Optional prerequisites for GDPS FlashCopy support:
  - Storage subsystems with FlashCopy support
- Optional prerequisites for Linux on z/OS support:
  - SuSE SLE8+ Linux for System z
- Optional prerequisites for GDPS DCM for Veritas Cluster Server:
  - Veritas Cluster Server HA/DR 5.0 package or later

  - Tivoli NetView for z/OS V5.2 or later
  - Distributed server hardware, the operating system release and service levels supported by Veritas Cluster Server

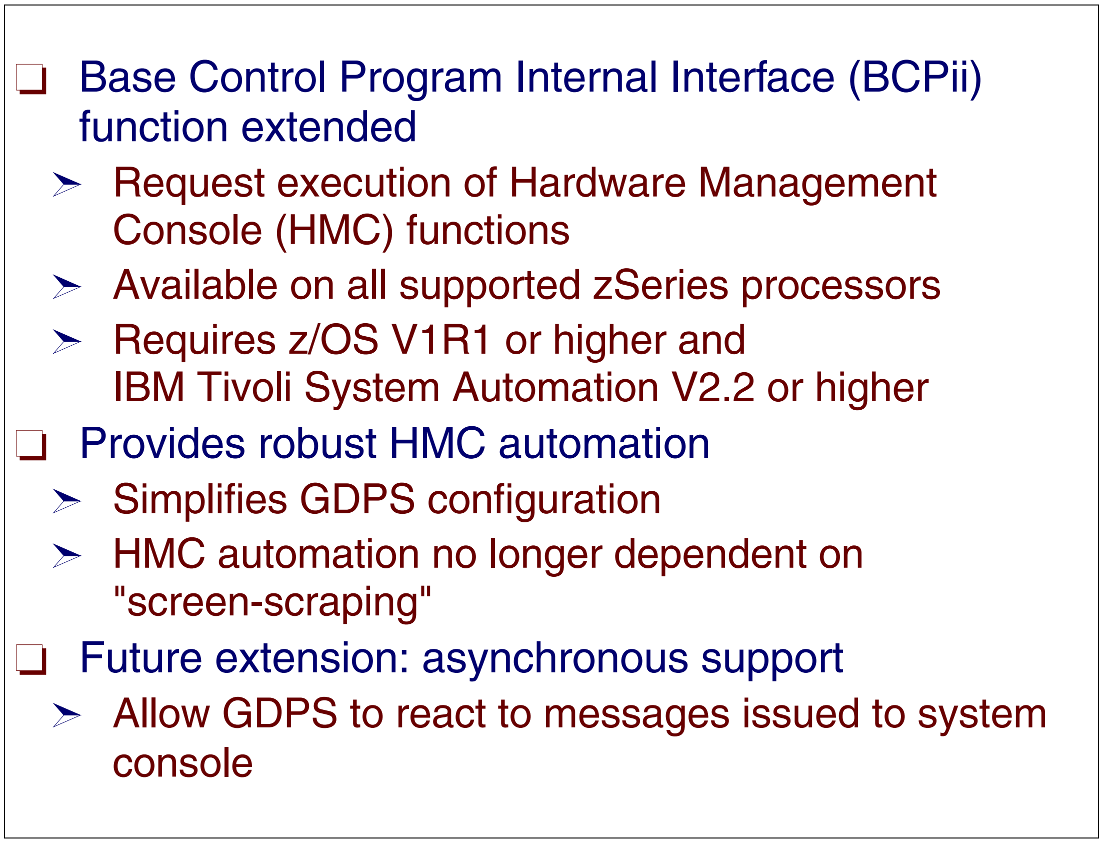
GDPS/XRC summary

GDPS/XRC is a powerful offering that provides an industry-leading, long distance, disaster recovery capability. It is based on the XRC technology, which is highly scalable (there are customers with close to 20,000 volumes being remote copied by XRC). XRC is industry-proven, having been available for well over a decade. XRC also has interoperability advantages: it is possible to have different disk subsystem types, and even different vendors, for the primary and secondary devices.

Building on the base of XRC, GDPS adds the powerful script capability that allows you to perfect the actions to be taken, either for planned or unplanned changes, eliminating the risk of human error. Combining its support of FlashCopy with the scripting capabilities significantly reduces the time and complexity to set up a disaster recovery test. And anyone who has been involved in DR planning will confirm that one of the most important factors in a successful disaster recovery process is frequent and realistic testing that is tied into your change management system. Having the ability to test your DR capability any time a significant change is implemented ensures that all aspects of application management are addressed.

In addition to its disaster recovery capability, GDPS/XRC also provides a much more user-friendly interface for monitoring and managing the remote copy configuration. This includes the initialization and monitoring of the XRC volume pairs based upon policy and performing routine operations on installed storage subsystems.

7.24 GDPS HMC automation interface



- ❑ Base Control Program Internal Interface (BCPii) function extended
  - ➢ Request execution of Hardware Management Console (HMC) functions
  - ➢ Available on all supported zSeries processors
  - ➢ Requires z/OS V1R1 or higher and IBM Tivoli System Automation V2.2 or higher
- ❑ Provides robust HMC automation
  - ➢ Simplifies GDPS configuration
  - ➢ HMC automation no longer dependent on "screen-scraping"
- ❑ Future extension: asynchronous support
  - ➢ Allow GDPS to react to messages issued to system console

Figure 7-25 HMC automation interface

GDPS HMC automation interface

GDPS requires the automation products listed in Figure 7-25 to be installed on all systems in the GDPS complex. GDPS depends on the application startup and the system shutdown sequences to be fully automated through an automation product.

HMC and consoles

To be able to control the processors in the remote center, you need to have access to the LAN containing the SEs and HMCs for the processors in that location. Such connectivity is typically achieved using bridges or routers. If you are running systems at the remote site, you will also want to be able to have consoles for those systems. Two options are 2074 control units and OSA-ICC cards. Alternatively, you could use SNA consoles; however, be aware that they cannot be used until VTAM is started, so they cannot be used for initial system loading.

Tivoli products

The GDPS automation code relies on the runtime capabilities of Tivoli NetView and Tivoli System Automation (SA). While these products provide tremendous first level automation capabilities in and of themselves, there are alternative solutions you may already have from other vendors. GDPS continues to deliver features and functions that take advantage of properties unique to the Tivoli products (like support for alert management through Tivoli IOM), but Tivoli NetView and Tivoli SA also work very well alongside other first level automation solutions. In other words, while there are indeed advantages to a comprehensive solution from IBM, you do not have to replace your current automation investments before moving forward with a GDPS solution.

Screen-scraping

“Screen scraping” does not work well with the 3270 block mode, so the OSA-ICC is expanded to include 3215 support on the z9 EC and BC and all newer processors as a PPRQ.

GDPS messages

Automation of NIP messages through the BCPii is contingent on no NIP console devices being active during IPL of a system. If there is an active NIP console device, the messages will be sent to this device and will not be automated.

To ensure that the NIP automation is successful, you have two options:

- Do not define any NIPCONS devices in your HCD definitions for the systems in your GDPS (neither for production systems nor for controlling systems). This will guarantee that the NIP messages are intercepted by the BCP Internal Interface and are sent to SYSCONS for automation. However, if you select to do this, you must ensure that the operators that perform IPLs have access to the HMC. This will allow the operators to troubleshoot in case an IPL appears to be hung for some reason.
- To use GDPS NIP console monitoring, a user must not have an active console at IPL time, thereby enabling the NIP messages to flow to GDPS via the BCPii SYSCONS interface. If a console previously defined as OFFLINE at IPL was changed to ONLINE at IPL, or if a console was recently installed at an MVS address corresponding to a NIP console (and that console is available at IPL time), then the GDPS NIP message monitoring is not operational, which can cause this GEO280A message since the external console would receive the NIP messages, thereby preventing them from being delivered across the BCPii to GDPS.

7.25 Peer-to-Peer Remote Copy (PPRC)

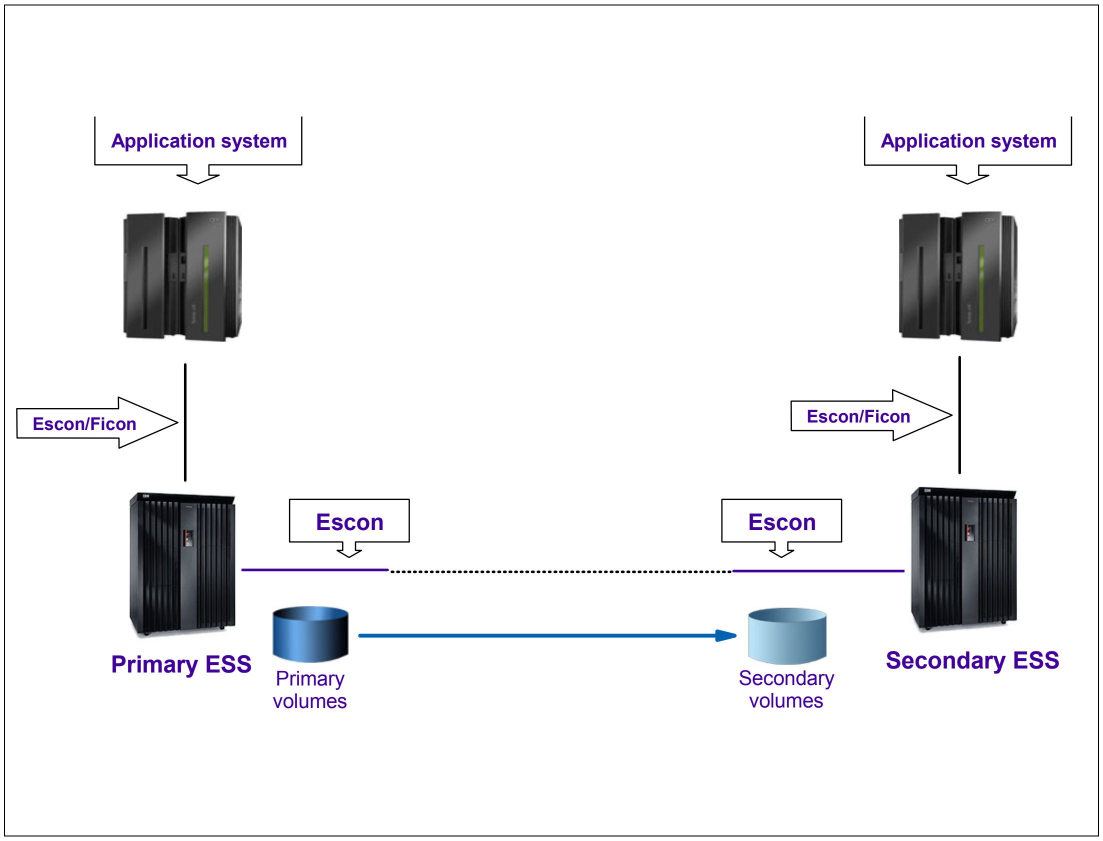


Figure 7-26 Peer-to-Peer Remote Copy

Peer-to-Peer Remote Copy (PPRC)

Here we summarize the Peer-to-Peer Remote Copy. PPRC (also called Metro Mirror in DS8000) is a hardware solution that enables the shadowing (mirroring) of data from one site and its associated DASD volumes (the primary volumes) to a second system at another site (the recovery system) and its DASD volumes (the secondary volumes).

Updates made on primary DASD volumes are synchronously shadowed onto secondary DASD volumes. The application write operation is only completed when data is secure on the secondary site. It is used primarily as part of the business continuance solution for protecting the organization's data against disk storage subsystem loss or complete site failure.

PPRC connectivity options

Connectivity between the primary and secondary disk subsystems can be provided by direct connections between the primary and secondary disk subsystems, by FICON switches, by DWDMs, and by channel extenders. The type of inter-site connection (dark fiber or telecommunications link) available determines the type of connectivity you use:

- Telecommunication links can be used by channel extenders.
- Other types of connectivity require dark fiber.

In a DS8000 the connection between the two controllers is through FCP links with an aggregate data rate of 200 MB/sec. For larger distances you might need:

- Channel Extenders over wide area network (WAN) lines
- Dense Wave Division Multiplexors (DWDM) over dark fibers

PPRC connectivity considerations

We have the following connectivity considerations:

- PPRC synchronous (SYNC) maximum supported distance is 300 km, but much less than this we have performance issues.

  The maximum distance supported for IBM Metro Mirror is 300 km (without an RPQ). Note that typical GDPS/PPRC and GDPS/HyperSwap manager configurations are limited to distances less than this due to Coupling Link or timer configurations. Additionally, you will need to contact other storage vendors (if required) to understand the maximum distances supported by their PPRC-compatible mirroring implementations.

  Note: References to Metro Mirror were replaced by PPRC when discussing IBM's synchronous mirroring architecture. The brand name of IBM Metro Mirror continues to be used for the implementation of the PPRC architecture included on the IBM Enterprise Storage Server (ESS) and DS8000 family of storage products. A similar change was made for XRC and the IBM brand name of z/OS Global Mirror.

- The connectivity infrastructure is transparent to PPRC.
- Evaluation, qualification, approval, and support of PPRC configurations using channel extender products, is the sole responsibility of the channel extender vendor.
- The channel extender vendors and DWDM vendors should be consulted regarding prerequisites when using their products.

7.26 PPRC data flow

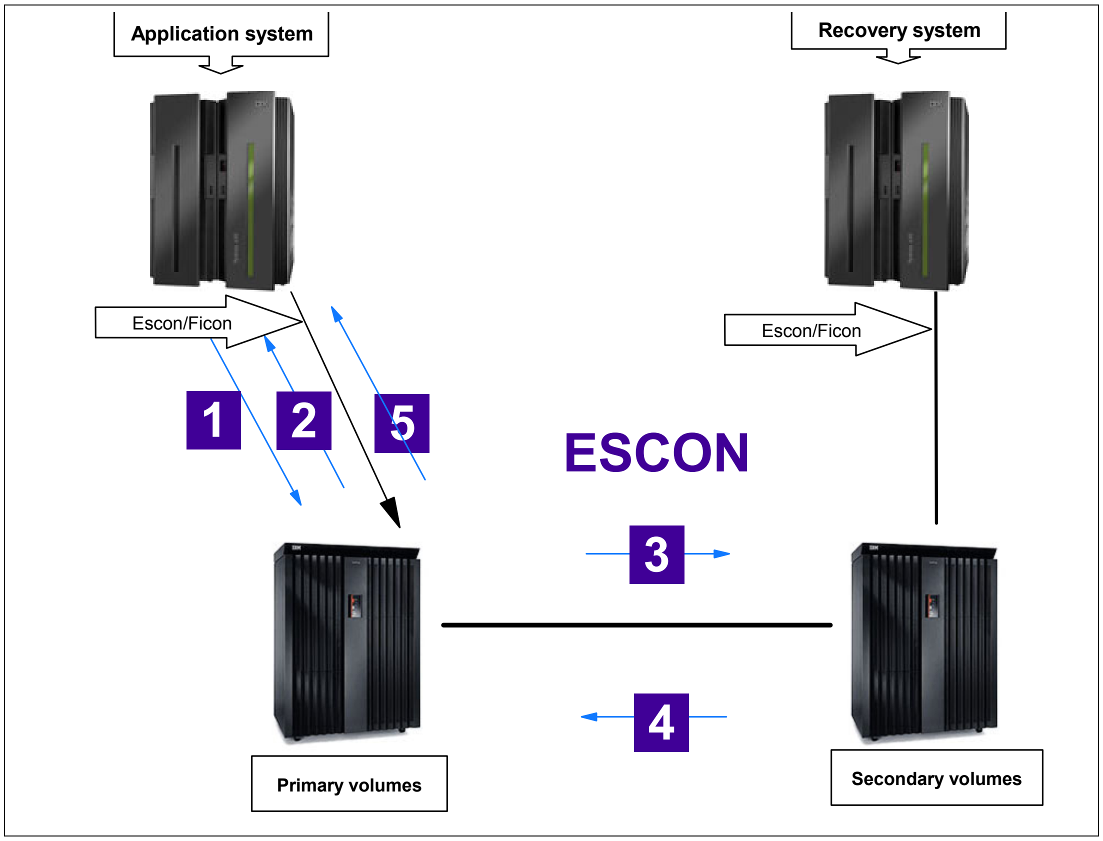


Figure 7-27 PPRC data flow

PPRC data flow

A PPRC data copy to the secondary ESS (or DS8000) is done synchronously as part of the primary volume's update I/O operation. When the application performs a write update operation upon a primary volume, this is what happens:

1. Write to the primary volume (disk subsystem cache and Non-Volatile Store (NVS)). Your production system writes data to a primary volume and a cache hit occurs.
2. Write to the secondary (disk subsystems cache and NVS). The primary disk subsystem's microcode then sends the update to the secondary disk subsystem's cache and NVS.
3. Signal write complete on the secondary. The secondary disk subsystem signals write complete to the primary disk subsystem when the updated data is in its cache and NVS.
4. Post I/O complete. When the primary disk subsystem receives the write complete from the secondary disk subsystem, it returns Device End (DE) status to your application program. At this point, the application program can continue its processing, and move on to any dependent writes that might have been waiting for this one to complete.

PPRC data consistency

PPRC on its own only provides consistency for a single write. Guaranteeing consistency across multiple logical subsystems and even across multiple disk subsystems requires automation on top of the PPRC function itself. This is where GDPS comes in with Freeze automation.

Note: GDPS uses automation, keyed off events or messages, to stop all mirroring when a remote copy failure occurs. In particular, the GDPS automation uses the IBM PPRC CGROUP FREEZE and CGROUP RUN commands, which have been implemented as part of Metro Mirror and also by other enterprise disk vendors. In this way, as long as the disk hardware supports CGROUP FREEZE/RUN commands, GDPS can ensure consistency across all data in the sysplex (consistency group) regardless of disk hardware type. This preferred approach differs from proprietary hardware approaches that only work for one type of disk hardware.

PPRC synchronous technique

This synchronous technique of PPRC ensures that application-dependent writes will be applied in the secondary volumes in the same sequence as in the primary volumes, thus providing application consistency at the recovery site at every moment. If during the process an error occurs that prevents the secondary copy from being updated, PPRC automatically suspends the mirroring function and applies the PPRC CRITICAL attribute behavior. Refer to "An example of write dependency" on page 313 for further discussion. Also, if the logical paths have been defined with the consistency group option enabled, then an extended long busy condition can be instigated that will allow for automation routines to take appropriate actions.

7.27 PPRC volume states - synchronous mode of operation

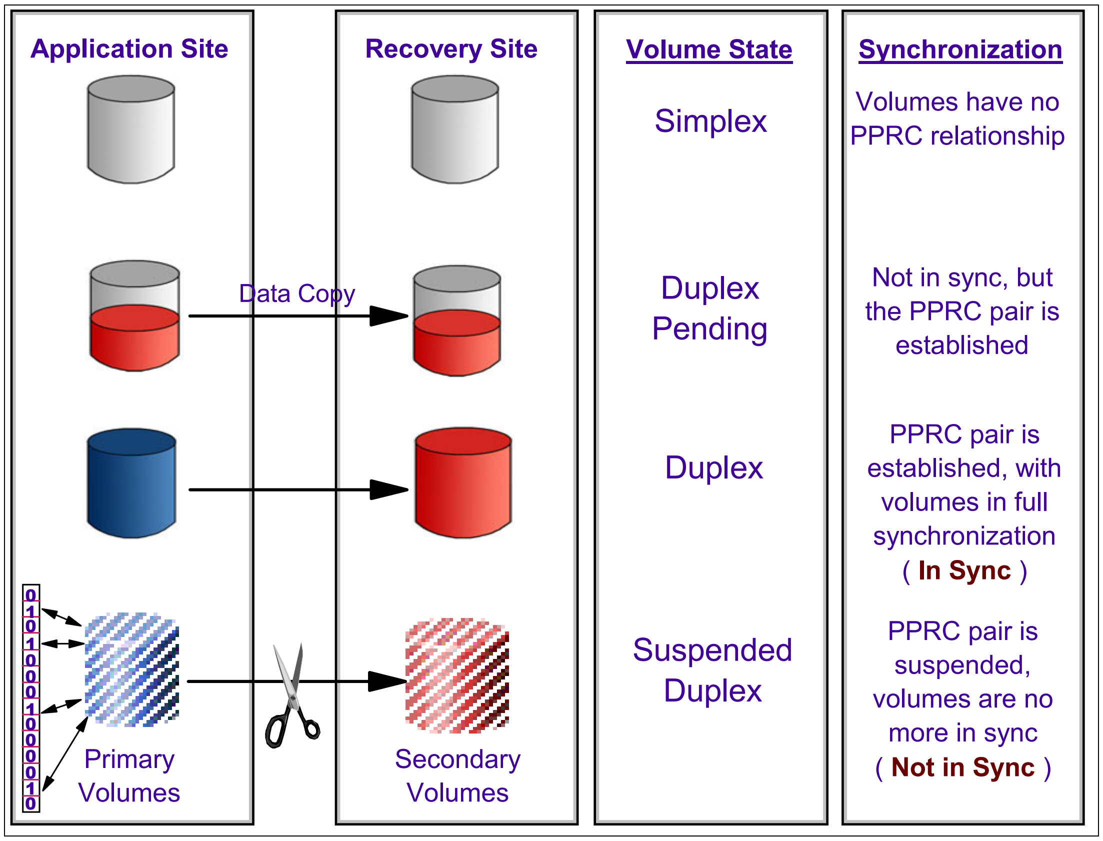

Application Site
Recovery Site
Volume State
Synchronization
Simplex
Volumes have no PPRC relationship
Data Copy
Duplex Pending
Not in sync, but the PPRC pair is established
Duplex
PPRC pair is established, with volumes in full synchronization
(In Sync)
0101000010000010
Suspended Duplex
PPRC pair is suspended, volumes are no more in sync
(Not in Sync)
Primary Volumes
Secondary Volumes

Figure 7-28 PPRC volume states - synchronous mode of operation

PPRC volume states - synchronous mode of operation

In order to manage PPRC configurations, you need to know the state of the PPRC volumes. You can convert PPRCOPY volume pairs from PPRCOPY extended distance mode to synchronous mode, and vice versa, on regular PPRCOPY pairs as well as on cascaded pairs. A switch to synchronous mode can be done using the PPRCOPY `ESTPAIR` command while the volume pair is active or after the volume pair has been suspended. As Figure 7-28 shows, at any given time a volume can be in either of the following states:

Simplex The initial state of a volume. A PPRC volume pair relationship has not been established yet between the primary and the secondary volumes.

Pending The initial state of a defined PPRC volume pair relationship, when the initial copy of the primary volume to the secondary volume is happening. This state is also found when a PPRC volume pair is re synchronized after it was suspended. During the pending period, PPRC is copying tracks from the primary to the secondary volume. The volume pair is not in synchronization.

Duplex The state of a PPRC volume pair after PPRC has fully completed the copy operation of the primary volume onto the secondary volume. At this moment the volume pair is in synchronization, and all write updates to the primary volume are synchronously applied onto the secondary volume.

Suspended In this state of the PPRC pair, the writes to the primary volume are not mirrored onto the secondary volume. The secondary volume becomes out of synchronization. During this time PPRC keeps a bit map record of the changed tracks in the primary volume. Later, the volume pair can be re synchronized, and then only the tracks that were updated will be copied. A PPRC volume pair will automatically go into a suspended state, for instance, when the primary ESS cannot complete a write operation to the recovery system ESS. Also, the operator can suspend pairs by command or using the ESS Copy Services WUI.

Note: You can convert PPRCOPY volume pairs from PPRCOPY extended distance mode to synchronous mode and vice versa on regular PPRCOPY pairs as well as on cascaded pairs. A switch to synchronous mode can be done using the PPRCOPY ESTPAIR command while the volume pair is active or after the volume pair has been suspended.

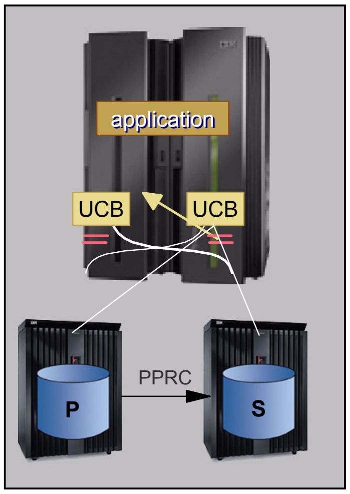
Monitoring PPRCOPY volume pairs

To monitor the copy process of PPRCOPY volumes, issue the `QUERY` command. When PPRCOPY extended distance is enabled, you should monitor volumes that have the most out-of-sync tracks waiting to be transmitted because of the delays introduced before updates are received by the recovery site. The amount of data lost during a disaster increases with the number of out-of-sync tracks.

PPRCOPY FAILOVER and FAILBACK

PPRCOPY FAILOVER and FAILBACK allows for reversing the direction of PPRCOPY pairs. The reason you would want to use FAILOVER is to provide access to the data from the secondary site if the primary site fails. When the primary site access is lost, the PPRCOPY ESTPAIR FAILOVER is issued to the original PPRCOPY secondary volumes. If the volumes receiving this command are PPRCOPY secondary volumes, then they must be in the full duplex or suspended state, or a PPRC Extended Distance secondary in duplex pending state.

7.28 GDPS/PPRC HyperSwap



- ❑ HyperSwap substitutes PPRC secondary for PPRC primary device
  - ➢ No operator interaction - GDPS-managed
  - ➢ Can swap large number of devices - fast
  - ➢ Changes status in secondary disk subsystem
  - ➢ Changes status in primary disk subsystem - if available
  - ➢ Non-disruptive - applications keep using same device addresses
- ❑ GDPS V3.7 functions are:
  - ➢ DCM support extended to Microsoft® Windows® clusters
  - ➢ Linux® on System z® integration with GDPS
  - ➢ HyperSwapTM I/O timing triggers - z/VM® and z/VM guests
  - ➢ SUSE SLES 11 support

Brings different technologies together to provide a comprehensive application and data availability solution

Figure 7-29 GDPS/PPRC HyperSwap

GDPS/PPRC HyperSwap

The GDPS/PPRC HyperSwap function is designed to broaden the continuous availability attributes of GDPS/PPRC solutions by extending the Parallel Sysplex *redundancy to disk subsystems*. The HyperSwap function can help significantly increase the speed of switching sites and switching disk between sites.

Figure 7-29 illustrates how applications can access the PPRC copy after a HyperSwap via the same set of UCBs. After a data set is allocated by a task, the UCB address cannot be changed. With HyperSwap this UCB, instead of mapping the primary device, now maps the secondary, without any application task disruption.

GDPS/PPRC HyperSwap Manager (GDPS/PPRC HM)

GDPS/PPRC HM is designed to extend the availability attributes of a Parallel Sysplex to disk subsystems, whether the Parallel Sysplex and disk subsystems are in a single site, or the Parallel Sysplex and the primary/secondary disk subsystems span across two sites.

It provides the ability to transparently switch primary disk subsystems with the secondary disk subsystems for either a planned or unplanned disk reconfiguration. It also supports disaster recovery capability across two sites by enabling the creation of a consistent set of secondary disks in case of a disaster or potential disaster. However, unlike the full GDPS/PPRC offering, GDPS/PPRC HM does not provide any resource management or recovery management capabilities.

HyperSwap function

The HyperSwap function is designed to deliver complete automation, allowing all aspects of a site or DASD switch to be controlled via GDPS from a single point of control. Prior to the advent of HyperSwap, DASD could be a single point of failure even when DASDs are mirrored.

In the event that there is a problem writing or accessing the primary disk because of either a primary disk hard failure or because the disk subsystem is not accessible or not responsive, then there is a need to swap from the primary disk subsystems to the secondary disk subsystems.

HyperSwap provides the ability to non-disruptively swap from using the primary volume of a mirrored pair to using what had been the secondary volume. Prior to the availability of HyperSwap, an IPL was required on every system if you wanted to switch and run from the secondary volumes, meaning that it was not possible to maintain application availability across a switch from primary to secondary volumes.

With HyperSwap, such a move can be accomplished without IPL and with just a brief hold on application I/O. The HyperSwap function is designed to be completely controlled by automation, thus allowing all aspects of the site switch to be controlled via GDPS.

7.29 GDPS/PPRC HyperSwap goal

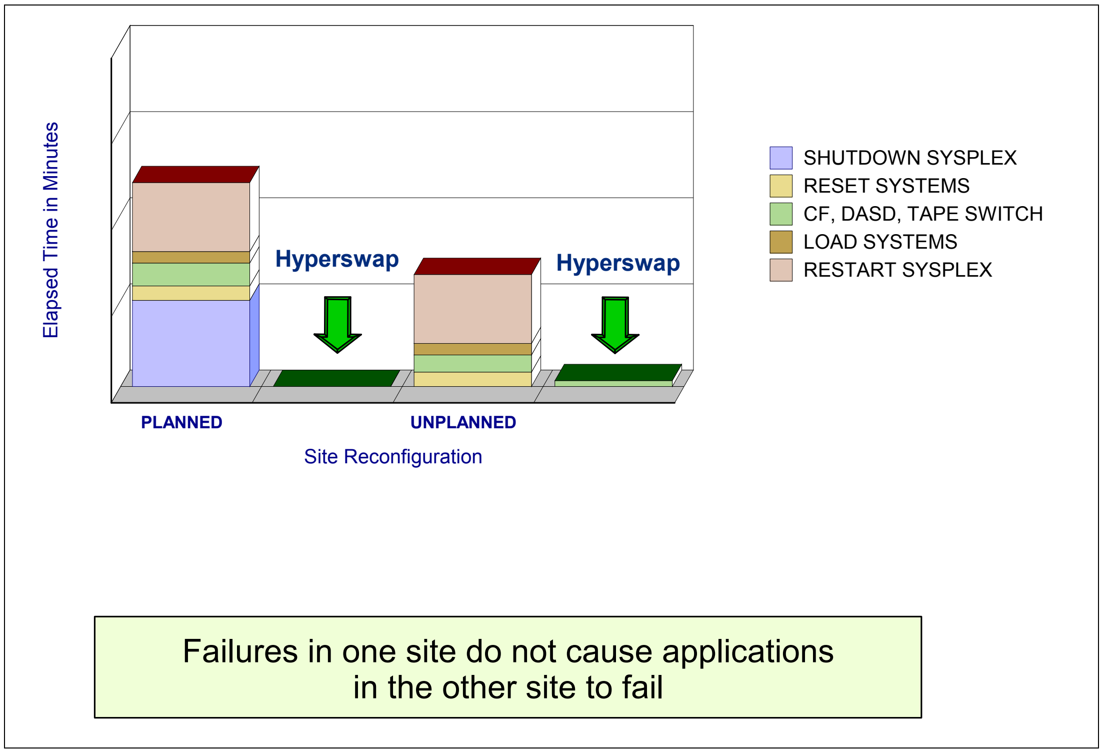


Figure 7-30 GDPS/PPRC HyperSwap goal

GDPS/PPRC HyperSwap goal

The overall availability of your Parallel Sysplex may be significantly improved with HyperSwap. The HyperSwap function delivers significantly faster primary/secondary PPRC DASD swaps both for planned and unplanned DASD reconfiguration activities. The value of HyperSwap can be obtained both in multi-site and single-site environments, as long as DASD is configured to exploit the PPRC function. Prior to HyperSwap, the time to accomplish planned or unplanned switches could take between one and two hours. Most of these activities can virtually be eliminated with HyperSwap, thus reducing the switch time to minutes.

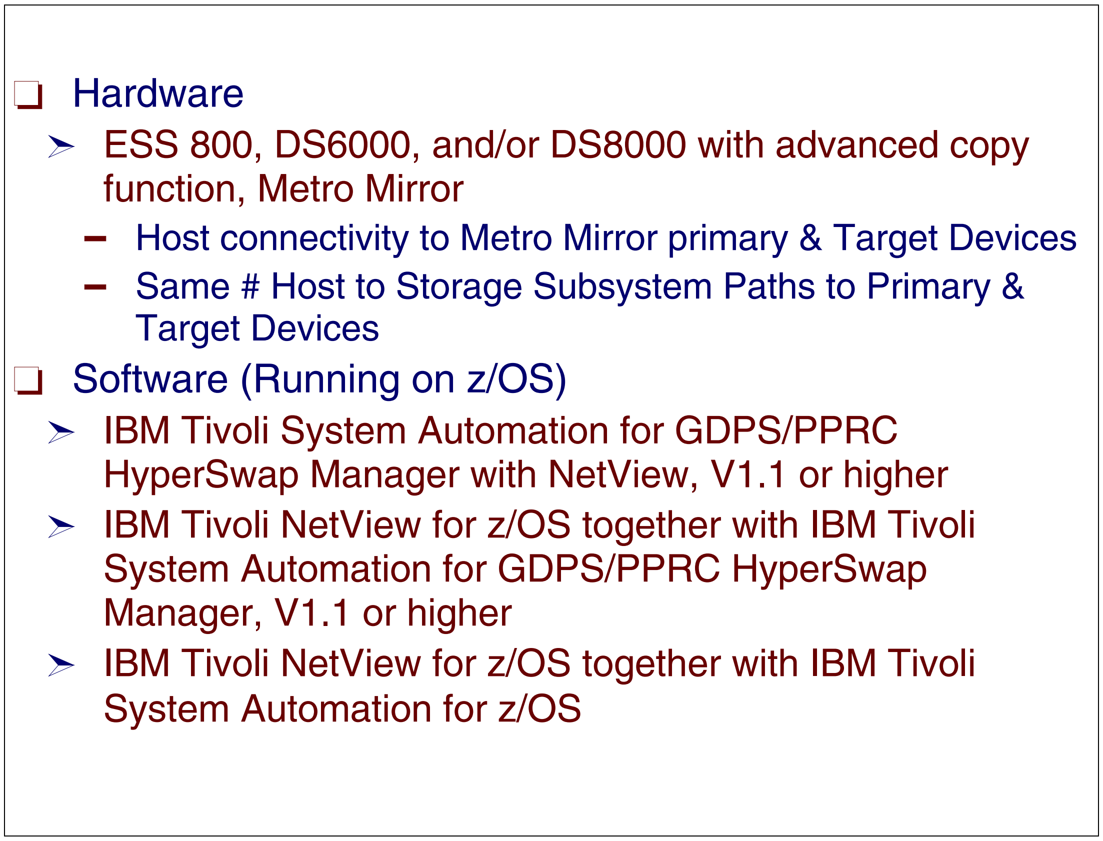
Planned HyperSwap

This allows you to transparently (that is, without having to quiesce the applications) switch primary PPRC disk subsystems with the secondary PPRC disk subsystems for a planned reconfiguration. Planned HyperSwap provides the ability to perform DASD configuration maintenance and planned site maintenance. Large configurations can be supported, because HyperSwap is designed to scale to swap large numbers of disk devices within a few minutes. A planned HyperSwap is invoked manually by operator action using GDPS facilities. One example of a planned HyperSwap would be where a HyperSwap is initiated in advance of planned disruptive maintenance to a disk subsystem.

Unplanned HyperSwap

This delivers the ability for GDPS/PPRC to transparently switch to the secondary PPRC disk subsystems in the event of unplanned outages of the primary PPRC disk subsystems, without data loss and without requiring an IPL. An unplanned HyperSwap is invoked automatically by GDPS, triggered by events that indicate the failure of a primary disk device.

7.30 GDPS/PPRC HyperSwap prerequisites



- Hardware
  - ESS 800, DS6000, and/or DS8000 with advanced copy function, Metro Mirror
    - Host connectivity to Metro Mirror primary & Target Devices
    - Same # Host to Storage Subsystem Paths to Primary & Target Devices
- Software (Running on z/OS)
  - IBM Tivoli System Automation for GDPS/PPRC HyperSwap Manager with NetView, V1.1 or higher
  - IBM Tivoli NetView for z/OS together with IBM Tivoli System Automation for GDPS/PPRC HyperSwap Manager, V1.1 or higher
  - IBM Tivoli NetView for z/OS together with IBM Tivoli System Automation for z/OS

Figure 7-31 GDPS/PPRC HyperSwap prerequisites

GDPS/PPRC HyperSwap prerequisites

GRS star mode must be implemented and all reserves must be converted to global enqueues using conversion RNL definition.

The DASD controller must support PPRC level 3 (extended query function). All 3390 volumes must be in PPRC duplex mode with the exception of the ones containing couple data sets. The PPRC configuration must be symmetric, which means each primary PPRC SSID must have a one-to-one secondary PPRC SSID counterpart.

The HyperSwap devices cannot be shared outside of the sysplex, and all CPCs channels must assure a sufficient bandwidth to primary and secondary subsystems.

The GDPS automation code relies on the runtime capabilities of Tivoli NetView and Tivoli System Automation (SA).

Hardware (ESS and DS8000)

Because Global Mirror is built on Global Copy (PPRC Extended Distance function), any platform that is supported by IBM Metro Mirror is also supported by Global Mirror. Currently this support is limited to the IBM Enterprise Storage Server (ESS) and the IBM DS8000 family of products.

FlashCopy SE is functionally not much different from the standard FlashCopy. The concept of space efficiency with FlashCopy SE relates to the attributes or properties of a DS8000 volume. As such, a space efficient volume could be used like any other DS8000 volume. However, the intended and only recommended use is as a target volume in a FlashCopy relationship.

Additionally, each of the GDPS solutions relies on IBM-developed disk replication technologies: PPRC for GDPS/PPRC, XRC for GDPS/XRC, and Global Mirror for GDPS/GM. These architectures are, of course, implemented on several IBM enterprise storage products. Specifically, PPRC has been implemented and branded as IBM System Storage Metro Mirror for the IBM Enterprise Storage Server (ESS) and the IBM DS8000 family of products. Similarly, the XRC technology has been implemented on the same storage servers under the brand name of IBM System Storage z/OS Global Mirror.

Note: References to Metro Mirror were replaced by PPRC when discussing IBM's synchronous mirroring architecture. The brand name of IBM Metro Mirror continues to be used for the implementation of the PPRC architecture included on the IBM Enterprise Storage Server (ESS) and DS8000 family of storage products.

7.31 Extended Remote Copy (XRC)

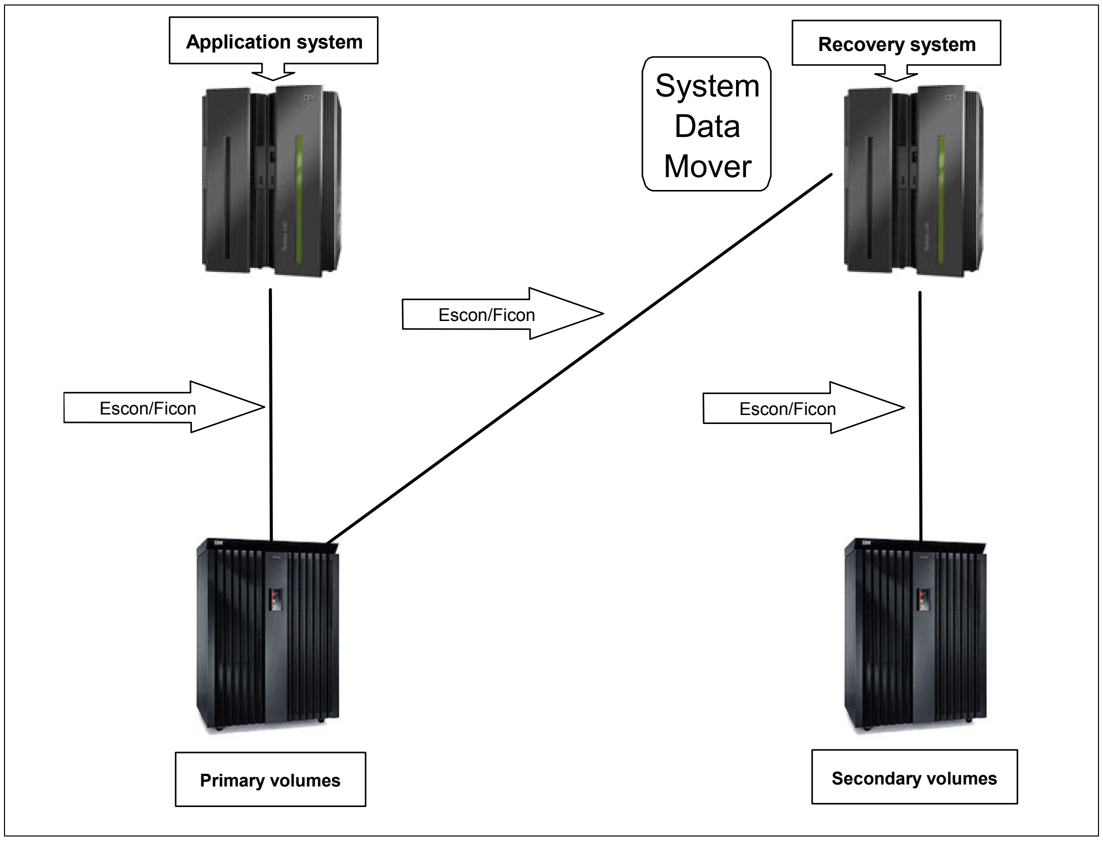


Figure 7-32 Extended Remote Copy (XRC)

Extended Remote Copy (XRC)

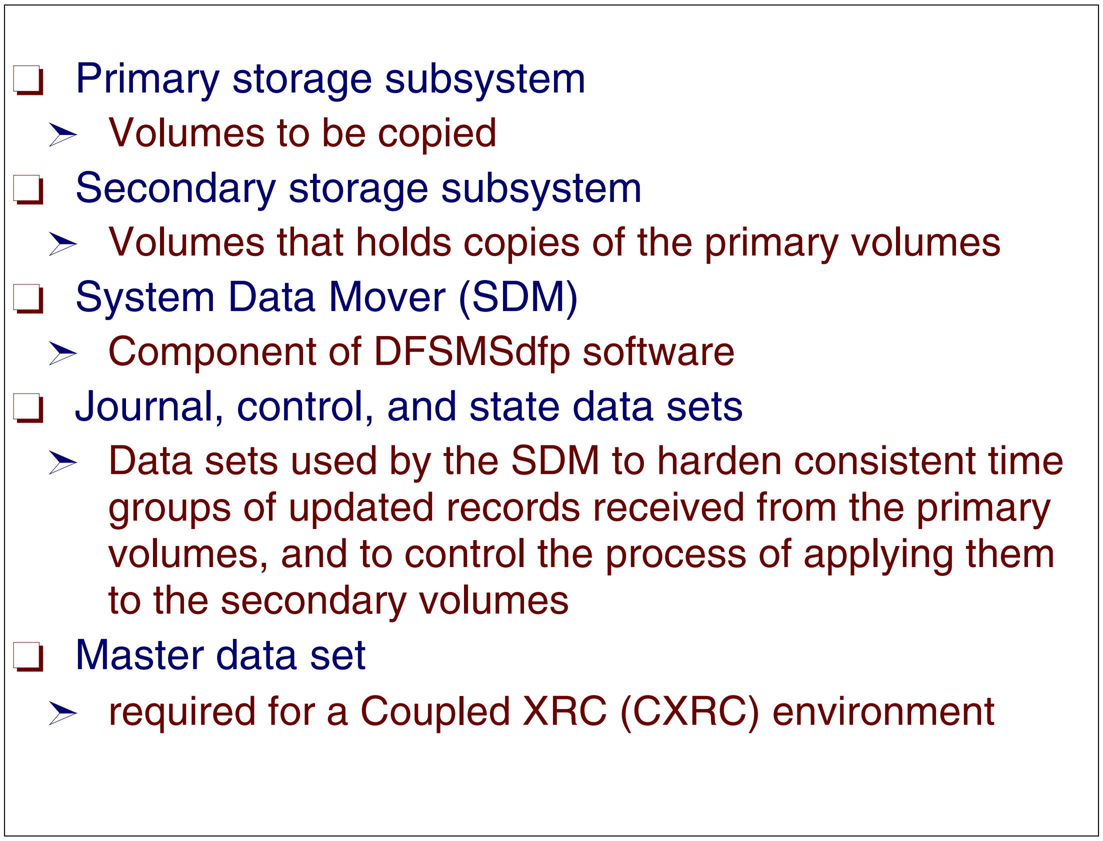
XRC (or Global Mirror in DS8000 terms) is a combined hardware and software asynchronous remote copy function available for the z/OS. It involves a System Data Mover (SDM), a DFSMSdfp component that asynchronously, on a timely basis, copies (using normal channels) changed data from the primary 3390s to the secondary ones. Then, XRC maintains a copy of the data asynchronously at a remote location over unlimited distances. It is an optional feature offering data integrity and data availability that can be used as part of business continuance solutions, for workload movement, and for data migration.

When ESS/DS8000 is used as the primary storage subsystem, XRC also supports unplanned outages. Such controllers maintain a hardware bitmap of the tracks changed on primary volumes by the primary systems. Whether an outage occurs or not, only changed tracks need to be copied to the secondary volumes when the connection between the controller and the SDM is established; then there is no need for a full resynchronization of the volumes. The SDM also exploits new ESS/DS8000 CCWs for improved performance.

The SDM benefits greatly from the higher bandwidth FICON channels provided to read the updates from the primary controller. A FICON channel can support up to five or more times the throughput of an ESCON channel.

The improved bandwidth of FICON together with the longer unrepeated distance support of up to 10 km (and up to 100 km with repeaters) results in a considerable improvement over ESCON channels. With ESCON, the recommendation was to use channel extenders through telecommunication lines beyond 25 km.

7.32 XRC components



- ❑ Primary storage subsystem
  - ➢ Volumes to be copied
- ❑ Secondary storage subsystem
  - ➢ Volumes that holds copies of the primary volumes
- ❑ System Data Mover (SDM)
  - ➢ Component of DFSMSdfp software
- ❑ Journal, control, and state data sets
  - ➢ Data sets used by the SDM to harden consistent time groups of updated records received from the primary volumes, and to control the process of applying them to the secondary volumes
- ❑ Master data set
  - ➢ required for a Coupled XRC (CXRC) environment

Figure 7-33 XRC components

XRC components

The components of XRC are described in this section.

Primary storage subsystem

The primary storage subsystem is a collection of volumes that is designated to be copied to a secondary site. These collections of volumes might be all the volumes at the primary site, or a subset of them. An XRC primary volume must be part of a primary storage subsystem that is XRC capable, such as the ESS/DS8000 with the optional XRC feature enabled. Attaching the primary volumes are one or more host applications running on one or more z/OS (or OS/390) system images.

Secondary storage subsystem

The secondary site storage subsystem is a collection of volumes that holds copies of the primary volumes. Each XRC primary volume has a corresponding secondary volume. The XRC secondary volumes can be part of *any* storage subsystem supported by the secondary z/OS, where the SDM is running.

System data mover (SDM)

The system data mover (SDM) is part of DFSMSdfp software, and must have connectivity to the primary and secondary controllers' volumes. When primary systems write to the primary volumes, the SDM manages the process of copying those updates asynchronously to secondary volumes.

To improve parallelism, multiple XRC sessions can be in effect per z/OS system. They can be coupled XRC (CXRC) or not. In a CXRC environment, update sequence integrity is managed across the coupled SDM sessions using the XRC master data set.

Coupled Extended Remote Copy (CXRC) expands the capability of XRC so that very large installations that have configurations consisting of thousands of primary volumes can be assured that all their volumes can be recovered to a consistent point in time. Up to 14 XRC sessions can be coupled to a single master session. The multiple SDMs coordinate their consistency group processing such that the recovery of all secondary volumes can be done to a single point of consistency.

Each SDM will have one XRC session that is responsible for a group of volumes. SDM maintains the updates' sequence consistency for the volumes participating in the XRC session, across LCUs in the ESS/DS8000 and across ESS/DS8000 (as well as with other primary storage subsystems that support XRC). Multiple instances of SDM on separate z/OS images are also possible.

The SDM for XRC operates in at least two system address spaces: ANTAS000, which is automatically started during IPL, and ANTASnnn (an ANTASnnn session will be started for each XSTART command). These address spaces are non swappable.

ANTAS000 handles TSO commands that control XRC. If this address space is cancelled, it is automatically restarted with no impact on XRC operations. ANTASnnn manages the movement of data from primary to secondary volumes. It manages the journal data sets and controls the application of updates to secondary volumes. If this address space is cancelled, the XRC session for this SDM is terminated.

Journal, control, and state data sets

The journal data set, control data set, and state data set are used by the SDM to harden—on disk—consistent time groups of updated records received from the primary volumes, and to control the process of applying them to the secondary volumes, thus maintaining sequence consistency. SDM creates consistency groups and writes them to the journal data sets.

The control data sets have pointers into the journal data sets indicating the last set of consistency groups written to the secondary volumes and the amount of data written to the journal.

The state data set maintains a list of XRC primary and secondary volumes, and is updated whenever a volume pair status for that session is changed.

Master data set

The master data set is only required for a coupled XRC (CXRC) sessions environment. It ensures recoverable consistency among all XRC subsystems contained within the CXRC system. At regular intervals the SDM writes data to the master data set. This data includes the last consistency group written to the secondary volumes, and the last consistency group written to the journal data sets. In the same I/O operation, the SDM reads the status of the last consistency group written to the secondary volumes, and the last consistency group written to the journal data set by all other SDMs in the coupled sessions. It also writes information about its own journal and write consistency time.

7.33 XRC data flow

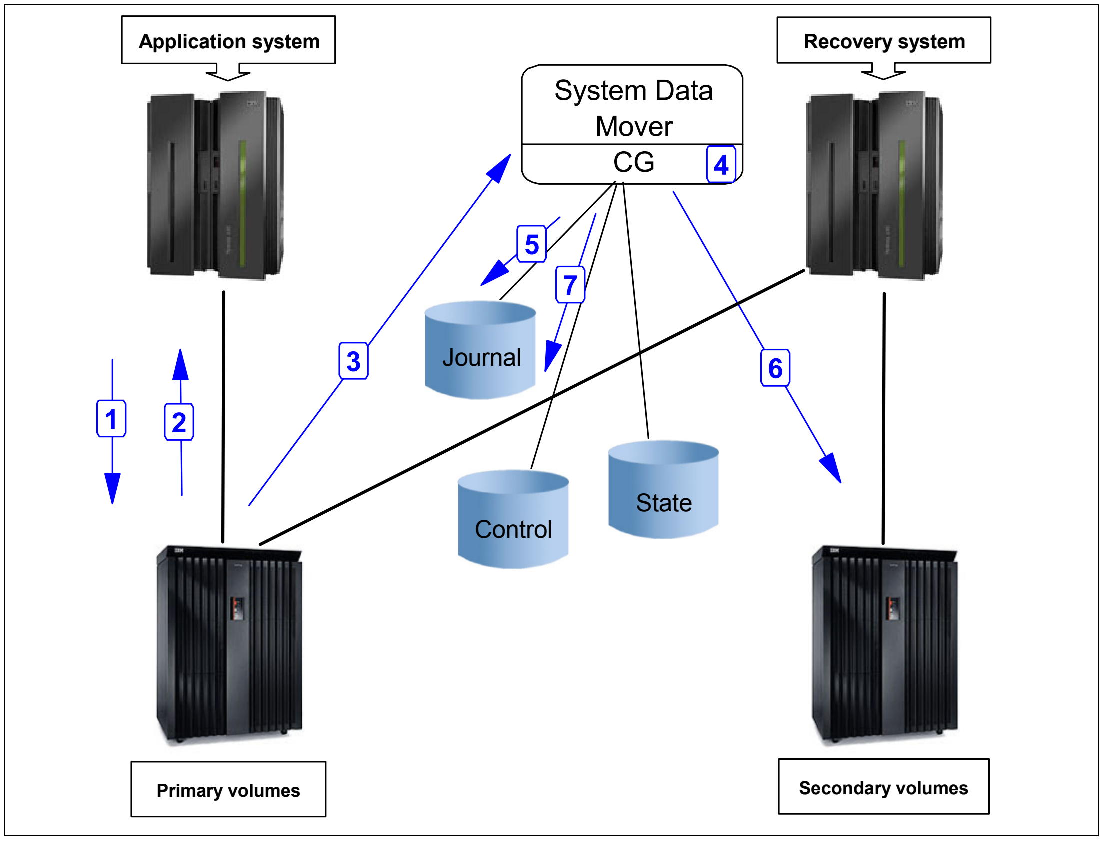


Figure 7-34 XRC data flow

XRC data flow

Figure 7-34 presents a simplified view of the XRC components and the data flow logic. But before discussing the logic, let us understand a key issue for the characteristic consistency of this process: *timestamping*.

Common time reference

When an XRC pair is established, this is signalled to the primary system, and the host system DFSMSdfp software starts to timestamp all write I/Os to primary volumes. This is necessary to provide data consistency across multiple LCUs. If those primary volumes are shared by systems running on different CECs, an IBM Sysplex Timer (or STP server) is required to provide a common time reference. If all the primary systems are running in different LPs on the same CEC, the system time-of-day clock can be used.

XRC is implemented in a cooperative way between the ESS/DS8000s on the primary site, Input Output Supervisor (IOS, a z/OS component) and the DFSMSdfp host system software component System Data Mover (SDM). The logic for the data flow is as follows (refer to Figure 7-34):

1. The primary system writes to the primary volumes; IOS includes a time stamp with data.
2. The application I/O operation is signalled completed when the data is written to primary ESS/DS8000 cache and NVS; this is when channel end and device end are returned to the primary system. Thus, the application write I/O operation has completed, and later the updated data is mirrored asynchronously through the following steps.

3. The ESS/DS8000 groups the updates into record sets (sorted by the time stamp) that are asynchronously off-loaded from the cache due to an SDM request. Because XRC uses this asynchronous copy technique, there is no performance impact on the primary applications' I/O operations.
4. The record sets, perhaps from multiple primary storage subsystem controllers, are processed into consistency groups (CGs) by the SDM sessions. The CG contains records that have their order of update preserved across multiple LCUs within an ESS/DS8000, across multiple ESS/DS8000s, and across other storage subsystems participating in the same XRC session. This preservation of order is absolutely vital for dependent write I/Os such as databases and their logs. The creation of CGs guarantees that XRC copies data to the secondary site with update sequence integrity.
5. When a CG is formed, it is written from the SDM real storage buffers to the journal data sets.
6. Immediately after the CG has been hardened on the journal data sets, the records are written to their corresponding secondary volumes. Those records are also written from SDM's real storage buffers. Because of the data in transit between the primary and secondary sites, the currency of the data on secondary volumes lags slightly behind the currency of the data at the primary site.
7. The control data set is updated to reflect that the records in the CG have been written to the secondary volumes.

7.34 GDPS error recovery

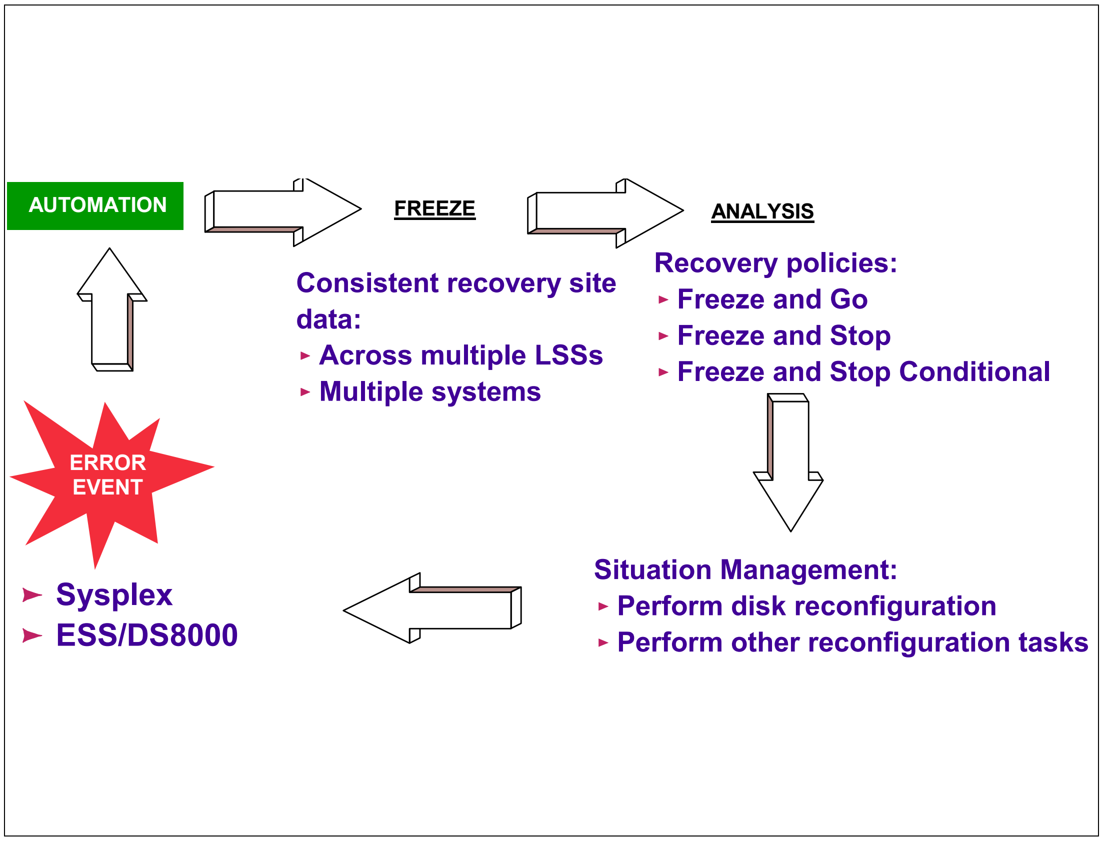


Figure 7-35 GDPS error recovery

GDPS error recovery

GDPS monitors many types of error events, the most significant category being changes in the remote copy configuration. This category is so important because this type of error may affect the time consistency of the recovery data. Figure 7-35 shows an example of the automatic procedure that a GDPS-based implementation could follow. The errors could be in Sysplex elements or in the remote copy activity.

When a volume is suspended because of a remote copy error condition, the automation routines immediately freeze the image of all the required secondary volumes, applying the CGROUP FREEZE command to all application-related LSSs that are known to GDPS. This preserves the consistency of the recovery data at that specified point in time. This action is followed by an analysis step. The analysis is driven by a predetermined policy, which can be either FREEZE and GO, or FREEZE and STOP, or FREEZE and STOP CONDITIONAL.

FREEZE and GO

With this option the automation routines allow applications to continue after the secondary volumes have been frozen. The implication is that updates to primary volumes after the freeze will be lost if a disaster occurs, and thus applications will have to be restarted in the recovery site. The other implication is that the availability of the application is favored, because it is not stopped by this error circumstance.

FREEZE and STOP

With this option the automation routines stop all applications right where they are at the time of freezing the secondary volumes. Application availability will be impacted, but the currency of the recovery site data is preserved.

FREEZE and STOP CONDITIONAL

With this option the automation routines freeze the image of the secondary volumes, and then analyze the reason for the suspend condition. If it turns out to be non-threatening, for instance, because a secondary piece of equipment has failed, then the applications are allowed to continue. However, if there is any indication of a serious problem, the automation routines then stop all the required applications.

7.35 GDPS/PPRC multi-site example

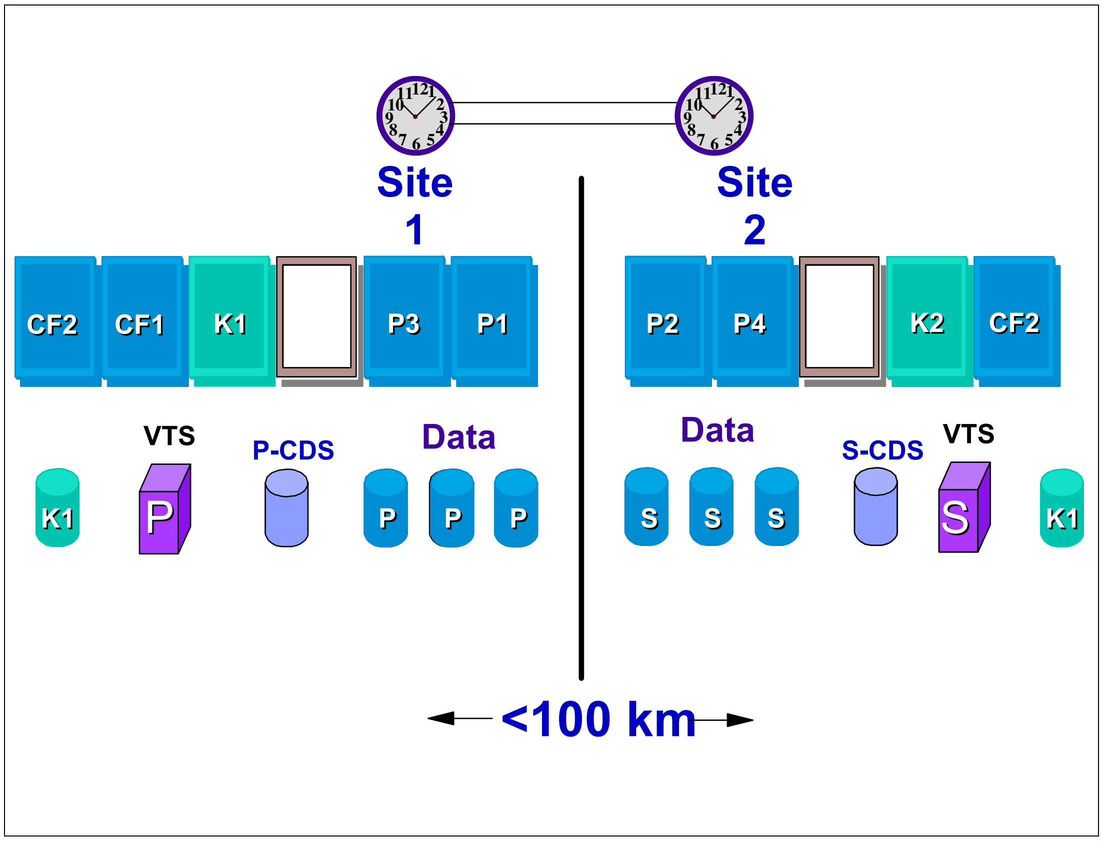


Figure 7-36 GDPS/PPRC multi-site example

GDPS/PPRC multi-site example

Figure 7-36 shows a GDPS/PPRC multi-site configuration. There is one Sysplex with production systems on both sites sharing data, for example a 50/50 distribution. There is a controlling system (K2) in Site 2. Both sites have the capability of increasing CPU resources through CBU to take over each other's work, and DASD data and tape data are mirrored (primary and secondary site). HW and SW must be configured in terms of continuous availability, which includes data sharing across two sites and cloned applications and z/OS systems.

Other GDPS/PPRC configurations

This configuration is also called 100/0 distribution.

- GDPS/PPRC single-site
  - One sysplex, all production systems in Site 1
  - Controlling system in Site 2
  - Expandable workload or CPU capacity backup (CBU) capable processors in Site 2
  - Primary DASDs and tapes in Site 1, secondary DASDs and tapes in Site 2
  - Site recovery by restarting failing system images in 2

GDPS/PPRC business recovery service (BRS) configuration

This is very similar to the single-site configuration, with the exception that Site 2 is owned by a third party and the controlling system will always be located in Site 1. Primary DASDs and tapes are located in Site 1and secondary DASDs and tapes in Site 2. K1 needs to be IPLed in Site 2 when Site 1 has a failure. This configuration can be used if:

- There is no second site but a disaster recovery (D/R) facility exists within maximum allowed PPRC distance.
- Dark fiber cannot be installed between company sites.
- You need to minimize cross-site link connectivity.
- An increased recovery time objective (RTO) is acceptable.

8

Availability in the zSeries and z9 environment

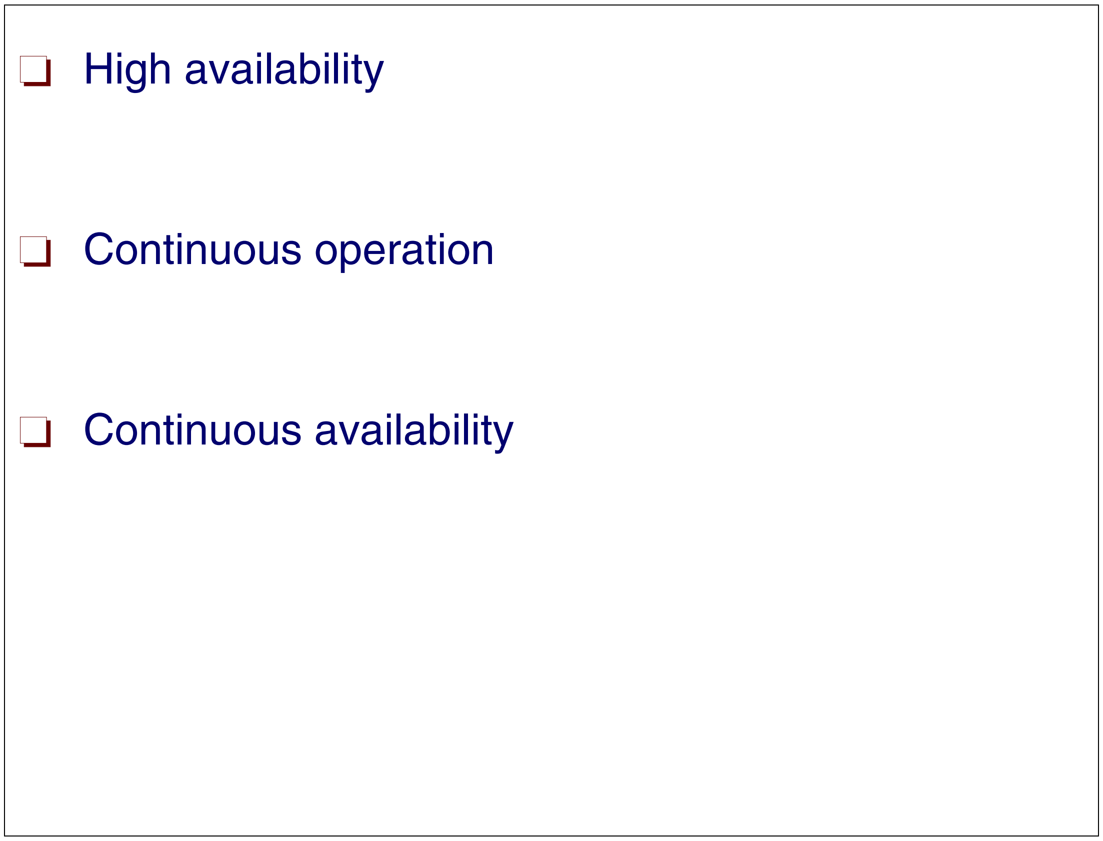
In this chapter we give you an overview of system availability in the zSeries environment. We discuss the following topics:

- High availability
- Continuous operation
- Continuous availability

We discuss what has to be done to plan and implement a continuous availability solution and how the IBM zSeries platform and its components can be used to accomplish this.

Even though you can use the general guidelines for other platforms also, we focus here on the zSeries platform only.

8.1 System availability terminology



- High availability
- Continuous operation
- Continuous availability

Figure 8-1 System availability terminology

High availability

The term *high availability* means the ability of a complex system, its subsystems, and its applications (independent of its complexity) to provide service to its users during defined service periods, without errors (hardware, software, or human) that might cause disruption of such services. In other words, no unplanned outages due to errors should occur. Many installations consider availability of 99.7 percent as an accepted minimum level for highly available systems. Service periods are normally defined in service level agreements (SLAs), a contract between the end users and the service providers (for example, a data center).

Continuous operation

Continuous operation means the ability of a complex system, its subsystems, and its applications to provide service to its users at any time, even when a partial planned outage is performed. This partial planned outage is necessary to perform maintenance for the system (programs) and its data. At first view it seems to be very difficult to perform change and maintenance activities on a system while keeping it in continuous operation mode.

Continuous availability

Continuous availability (CA) refers to the ability of a complex system, its subsystems, and its applications to provide both high availability *and* continuous operation at the same time. In other words, continuous availability is going to produce the famous “24 by 7.” You can imagine that it might be difficult to approach this level of operation because hardware and software components are not entirely free of errors and maintenance, and there are also many changes in the computing environment. Therefore, it is important to make use of hardware, software, and operational procedures that hide outages from the end user. Thus it is essential for many zSeries processors and z/OS that changes and maintenance be possible while the systems are up and running. These continuous availability attributes eliminate the need for switching the workload to an alternate system to perform changes and maintenance on the primary system.

The lack of continuous availability is caused by the existence of single points of failure (SPOF) in the elements composing the computational environment. The technique to fight back the SPOFs is redundancy, which implies additional capacity. One type of redundancy is cloning. One example of cloning is all the z/OSs in a Parallel Sysplex fully exploiting data and program sharing. If one z/OS has a planned or unplanned outage, the other z/OSs execute the arriving transactions, because due to cloning the functionality is the same.

However, the most serious problem affecting CA is the multiple concurrent failures scenario. All the redundancy that we can put together might not be enough to maintain the 24 by 7 goal. This scenario belongs in the realm of disaster recovery, where the target is just to reduce the mean time to repair (MTTR).

An example of multiple concurrent failures could be the outage of the primary sysplex couple data set together with one z/OS. In this case, because of integrity issues, all the z/OSs need to be IPLed again despite the existence of an alternate sysplex couple data set.

8.2 Continuous availability as a target

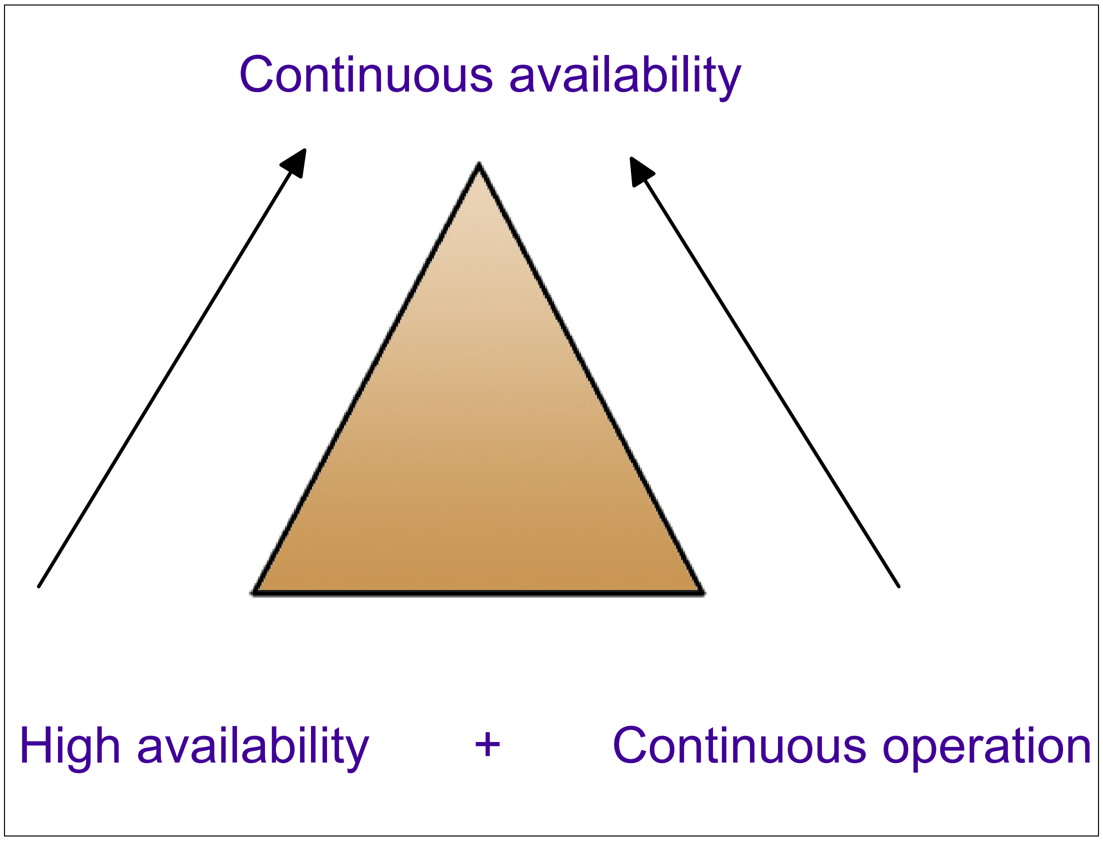
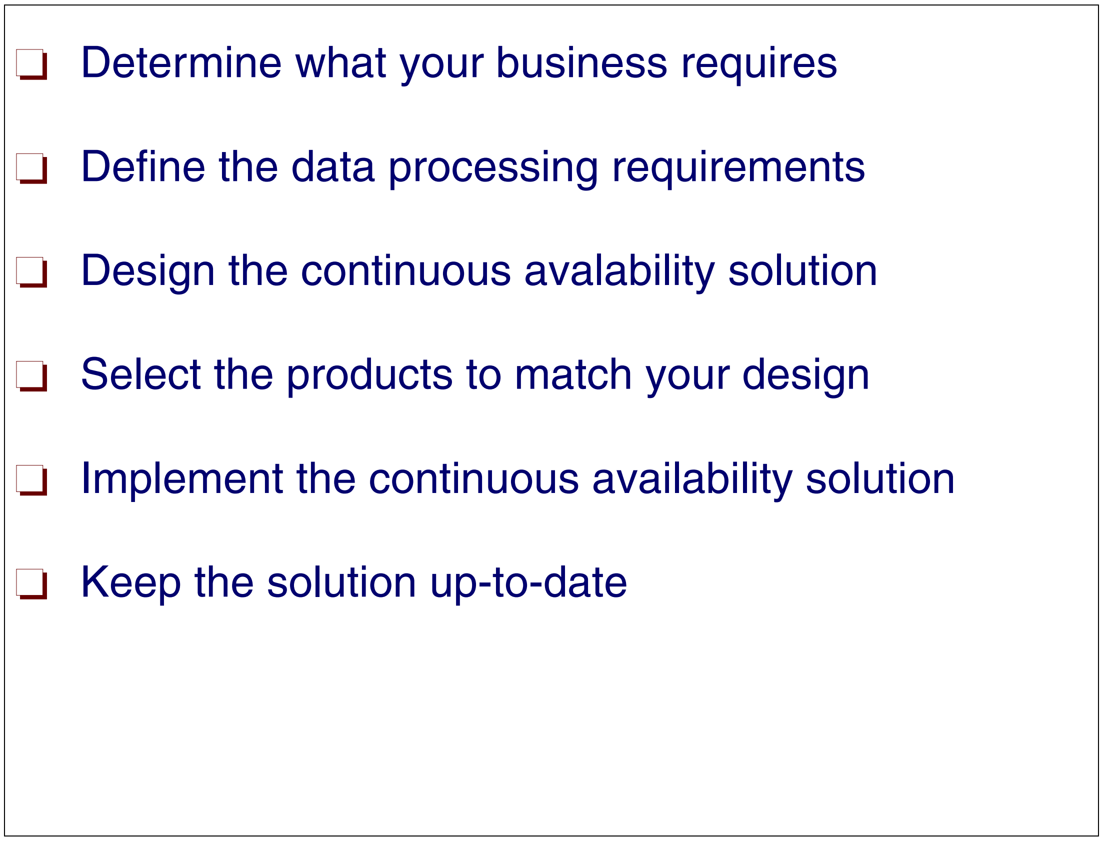


Figure 8-2 Continuous availability as a target

Continuous availability as a target

Figure 8-2 shows the dependencies between the three terms we introduced. A system can be either highly available or continuously operational, or it can be a composition of both. In the third case it will achieve the highest level. But before you decide to make a system continuously available, you first have to determine what your business requirements are. Then you will have to take further steps in the implementation of your solution. We discuss these next.

8.3 A structured approach to continuous availability design



- ❑ Determine what your business requires
- ❑ Define the data processing requirements
- ❑ Design the continuous avalability solution
- ❑ Select the products to match your design
- ❑ Implement the continuous availability solution
- ❑ Keep the solution up-to-date

Figure 8-3 A structured approach to continuous availability design

Determine what your business requires

First you have to determine which of the aspects of continuous availability are required for which business processes. In today's world a huge number of business processes depend on information technology and available resources. That means if an outage occurs, they might come to a complete standstill, which means loss of business and revenue. The impact of such outages has to be evaluated by performing a *business impact analysis*.

Define the data processing requirements

Once you have established your business requirements, you must translate them into data processing terms for use during system design. The results of these data processing requirements are *service level objectives*.

Design the continuous availability solution

After the data processing requirements have been established and agreed upon, you must design a technical solution that will satisfy these requirements. Designing means that you define the characteristics and major elements of your solution. It is a generic description of hardware, software, and systems management functions required for the desired level of availability. Very often, you can estimate the cost of your solution only after you have done a high-level design.

Select the products to match your design

Once you have completed the design for your availability solution, you can then select the appropriate zSeries products. These products will, of course, have a major influence on the cost of your overall solution.

Implement the continuous availability solution

You are now ready to implement your availability solution. You have to install the required zSeries hardware and software components, introduce systems management procedures, and develop appropriate system documentation.

Keep the solution up-to-date

After implementing the continuous availability solution, it is very important that you put procedures in place to ensure that your solution remains viable regardless of changes in your computing environment. You do this through systems management procedures that help you define, verify or implement availability measures at each system change or introduction of new applications.

Tip: Today it is very common to align the systems management procedures with ITIL® (*IT Infrastructure Library*®). ITIL in a sense is just a new name for the traditional systems management tasks. We recommend that you get familiar with the concepts of this most widely accepted approach to IT service management. It is not zSeries specific, but it is used on this platform. For more information about ITIL, refer to:

http://www.itil.co.uk

8.4 Decision criteria for a continuous availability design

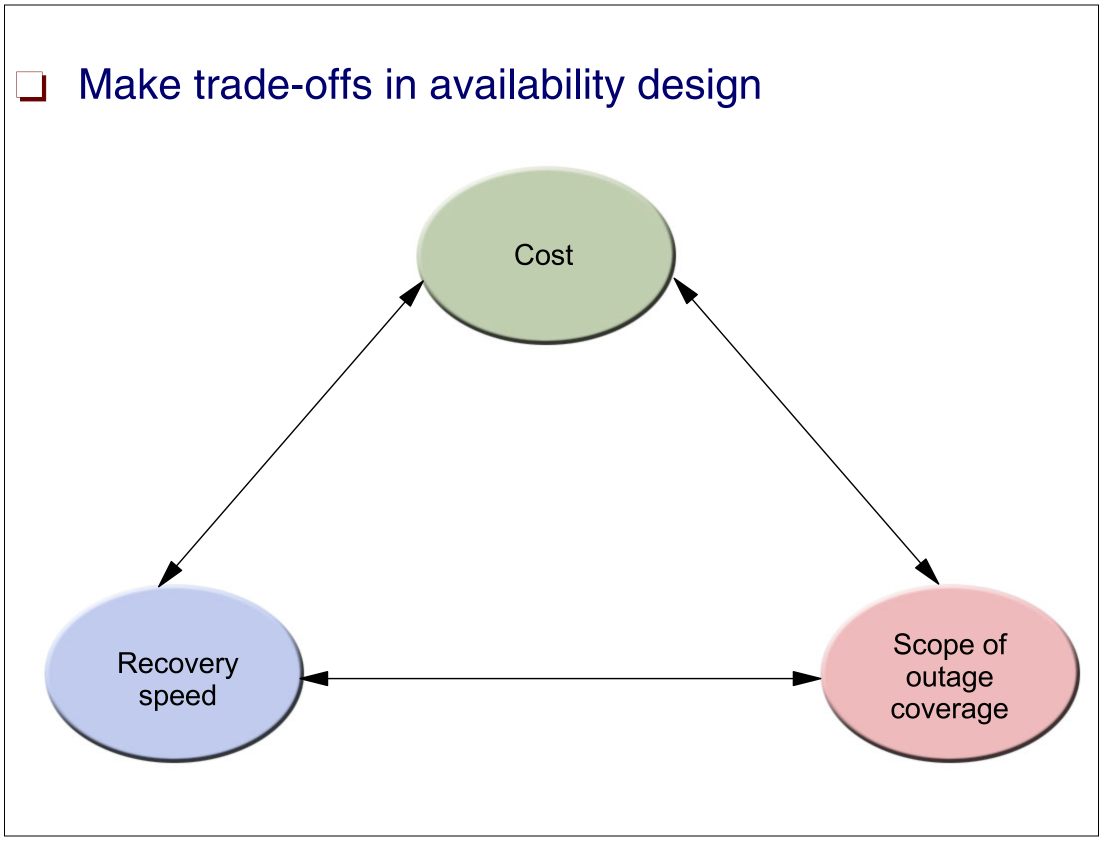


Figure 8-4 Decision criteria for a continuous availability design

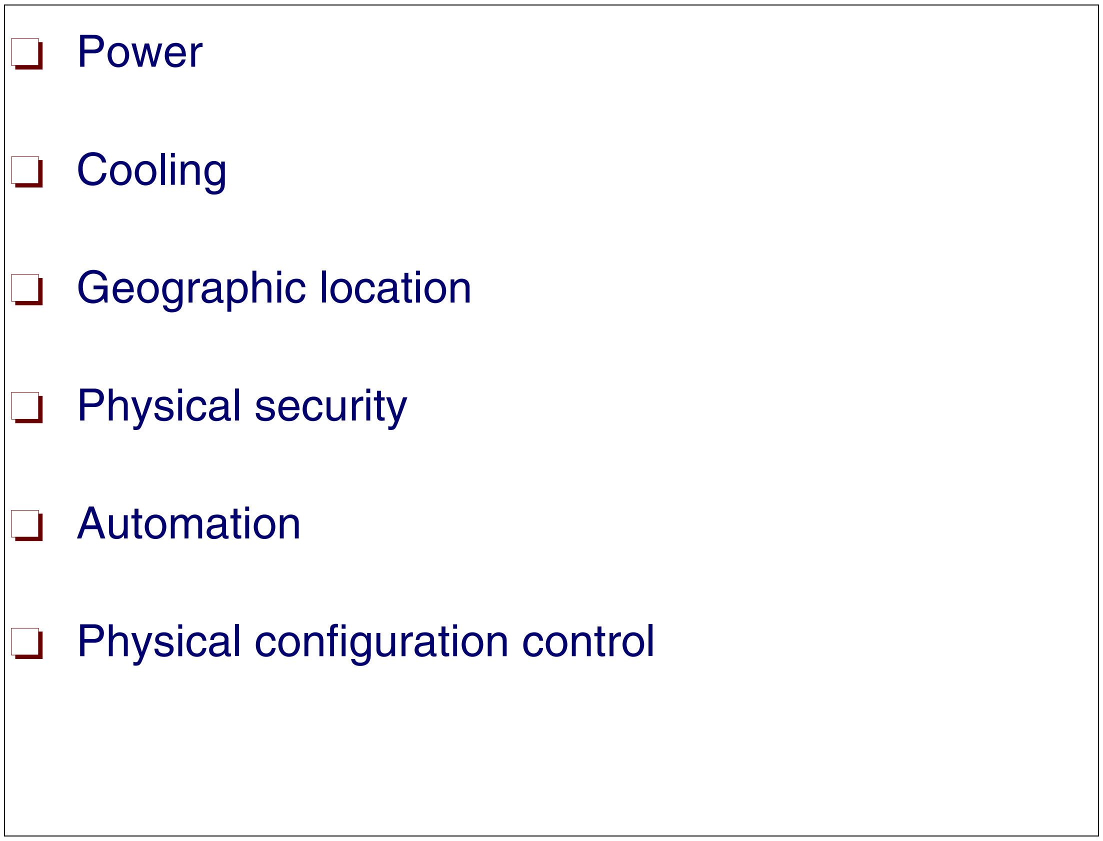
Make trade-offs in availability design

Although you want to develop an availability solution that exactly matches the requirements of your business, you must be aware that some requirements are trade-offs, or are even mutually exclusive. In addition, the more stringent the requirements are, the higher the cost of the designed solution will be.

You need to ask the following questions:

- Cost - What will the cost of the targeted solution be?
- Recovery speed - How fast must the recovery from an outage be accomplished? How long can a downtime last without affecting the critical business?
- Scope of outage coverage - What type of outage does the designed solution cover?

8.5 Hardware - Environmental aspects



- ❑ Power
- ❑ Cooling
- ❑ Geographic location
- ❑ Physical security
- ❑ Automation
- ❑ Physical configuration control

Figure 8-5 Hardware - Environmental aspects

Power

There are two basic requirements for your power infrastructure:

- ▶ The normal main power supply should be as reliable and robust as possible, with a minimal number of outages.
- ▶ The infrastructure you provide must be able to keep at least your essential equipment up and running during any power interruption. There are no-breaks options covering such requirements.

In order to satisfy these requirements, the following has to be carefully considered:

- ▶ Ensure that critical components have redundant power connections and that these components are actually connected to redundant power sources.
- ▶ Provide an uninterruptable or redundant power supply for critical components.
- ▶ Establish procedures to test failover to the redundant power source on a regular basis.

Cooling

You have to provide sufficient cooling capacity and redundancy to ensure that environment conditions can be maintained in case of an equipment failure. Even though modern mainframe equipment no longer requires water cooling, a temperature-controlled environment *is* still required.

Therefore, make sure that the cooling capacity is sufficient and install a redundant cooling unit that can take over in the event of equipment failure. Establish procedures to ensure speedy takeover in the event of an environmental failure.

Geographic location

When deciding where to place a data center, you should also consider the following:

- Neighboring establishments. For example, a chemical leakage at an adjoining plant can mean that physical access to your installation is denied, even if there is nothing wrong with your site.
- At a minimum, you should ensure that all your equipment can be operated, monitored, and controlled from a remote location that is distant enough so that access will not be denied to your facility in case of a local disaster.
- Avoid, if possible, choosing a place with intense seismic activity.

Physical security

There has always been a need for strict security at computer sites because of the value and sensitivity of the equipment there. However, in the modern world, there is the additional risk of terrorism.It is extremely expensive, maybe even not possible, to protect a computer installation from all potential forms of attack.

The best line of defense is to use remote copy technology and GDPS (discussed in detail in Chapter 7, "Geographically Dispersed Parallel Sysplex" on page 293) to mirror all data to a remote site that can take over operations at very short notice in the event of a disaster impacting one of the sites. While it may not be possible to have a 100% secure primary site, the chances of losing two adequately equipped and protected sites should be low enough to constitute an acceptable business risk.

Certainly one of the ways of improving security is through anonymity and protection of confidential information. Many companies will no longer divulge the location of their computer facilities. Another simple recommendation is to prevent removal from the site of disks that are assumed to be defective (as customer engineers from DASD storage controller vendors might try to do).

Automation

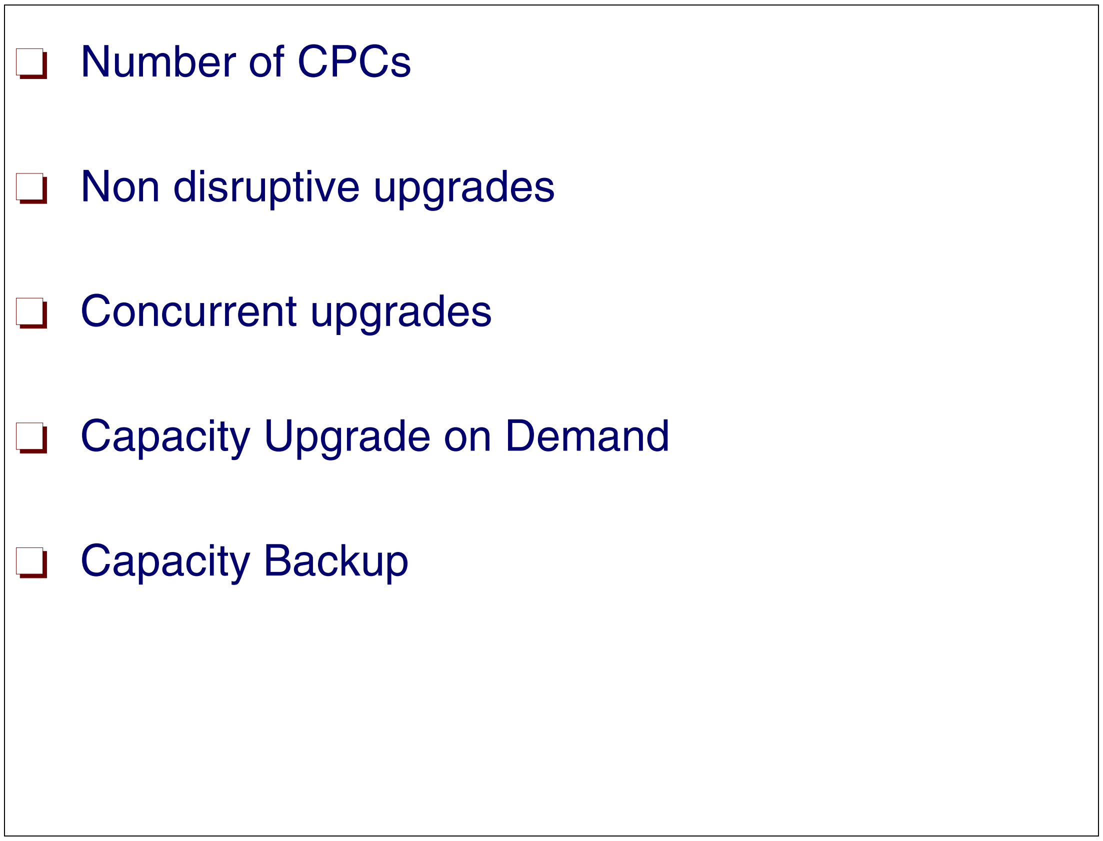
Nearly all hardware is designed so that some form of message or alert is given in case of a problem or failure. However, there is no point issuing these alerts unless they reach the person responsible for that piece of equipment. It is not unknown for a redundant piece of hardware (like a Sysplex Timer, for example), to be broken for months without anyone noticing.

Therefore, automation should be put in place to trap and forward such messages and alerts to the responsible party, and also to at least one other individual (to avoid single points of failure).

Physical configuration control

In order to ensure that devices are actually connected according to your plans—and most especially to ensure that devices and cables can be easily identified in case of a problem—it is vital that all equipment and cables are clearly labelled. Ideally, an integrated cable management system is employed. An example is Hardware Configuration Management (HCM), an optional component of z/OS, which lets you control not only your hardware devices, but also the cables used to connect them, cable cabinets, and so on.

8.6 Hardware - Central processing complexes



- Number of CPCs
- Non disruptive upgrades
- Concurrent upgrades
- Capacity Upgrade on Demand
- Capacity Backup

Figure 8-6 Hardware - Central processing complexes

Number of CPCs

In order to avoid a single point of failure, you should always configure a minimum of two central processing complexes (CPCs), regardless of the fact that a single CPC might be able to provide sufficient processing power to handle the whole workload. Even though the reliability of CPCs is increasing all the time, there are still a few planned changes that require a shutdown or Power-On-Reset of a whole CPC, so having at least two CPCs allows you to maintain application availability at all times.

For availability reasons, the CPCs should be placed in different locations, or if in the same location, at least separated from each other by a fireproof/flood-proof wall. When placing CPCs in different locations, it is important to be aware of the distance limitations of CF links, ESCON channels, Sysplex Timer links, and so on. Also be aware of the performance impact the distance can have on your applications.

Each CPC should have *at least one*, and preferably two, LPs that participate in the production sysplex. By doing this, it is possible to lose one CPC and still have the possibility to keep your applications available. By having two production LPs on each CPC, you still have access to the MIPS of that CPC even if you need to shut down one of those LPs.

Non-disruptive upgrades

CP, IFL, ICF, and/or zAAP processors can be concurrently added to a z9 CPC if there are spare PUs available on any installed book. The number of zAAPs cannot exceed the number of CPs plus unassigned CPs on a z9 CPC.

Additional books can also be installed concurrently, allowing further processor upgrades. A processor upgrade cannot be performed when CBU or On/Off CoD is activated. Concurrent upgrades are not supported with CPs defined as additional SAPs. If reserved processors are defined to a logical partition, then z/OS, OS/390, and z/VM operating system images can dynamically configure more processors online, allowing nondisruptive processor upgrades. The Coupling Facility Control Code (CFCC) can also configure more processors online to Coupling Facility logical partitions using the CFCC image operations panel.

Concurrent upgrades

The z9 CPCs have the capability of concurrent upgrades, providing additional capacity with no server outage. In most cases, with prior planning and operating system support, a concurrent upgrade can also be nondisruptive, that is, without system outage (Power-on Resets (PORs), logical partition deactivations, and IPLs do not have to take place).

Given today's business environment, the benefits of the concurrent capacity growth capabilities provided by z9 servers are plentiful, and include:

- Enabling exploitation of new business opportunities
- Supporting the growth of e-business environments
- Managing the risk of volatile, high-growth, and high-volume applications
- Supporting 24x365 application availability
- Enabling capacity growth during "lock down" periods

This capability is based on the flexibility of the z9 system design and structure, which allows configuration control by the Licensed Internal Code (LIC) and concurrent hardware installation.

Licensed Internal Code (LIC)-based upgrades

The LIC - Configuration Control (LIC-CC) provides for server upgrade with no hardware changes by enabling the activation of additional, previously installed capacity. Concurrent upgrades via LIC-CC can be done for:

- Processors (CPs, IFLs, ICFs, and zAAPs) - Requires available spare PUs on installed books.
- Memory - Requires available capacity on installed memory cards.
- I/O cards ports (ESCON channels and ISC-3 links) - Requires available ports on installed I/O cards.

Concurrent hardware installation upgrades

Configuration upgrades can also be concurrent by installing additional:

- Books (which contain processors, memory, and STIs) - Requires available book slots in the installed CEC cage.
- I/O cards - Requires available slots on installed I/O cages. I/O cages cannot be installed concurrently.

The concurrent upgrade capability can be better exploited when a future target configuration is considered in the initial configuration. Using this Plan Ahead concept, the required number of I/O cages for concurrent upgrades, up to the target configuration, can be included in the z9 CPC initial configuration.

Capacity Upgrade on Demand

Capacity Upgrade on Demand (CUoD) is a function available on z9 CPCs that enables concurrent and permanent capacity growth.

The CUoD function is based on the Configuration Reporting Architecture, which provides detailed information on system-wide changes, such as the number of configured Processor Units, system serial numbers, and other information.

CUoD provides the ability to concurrently add processors (CPs, IFLs, ICFs, and zAAPs), memory capacity, and I/O ports. The concurrent upgrade can be done by Licensed Internal Code Configuration Control (LIC-CC) only or also by installing additional books and/or I/O cards.

CUoD is ordered as a “normal” upgrade, also known as Miscellaneous Equipment Specification (MES). CUoD does not require any special contract, but requires IBM service personnel for the upgrade. In most cases, a very short period of time is required for the IBM personnel to install the LIC-CC and complete the upgrade.

To better exploit the CUoD function, an initial configuration should be carefully planned to allow a concurrent upgrade up to a target configuration. You need to consider planning, positioning, and other issues to allow a CUoD nondisruptive upgrade. By planning ahead, it is possible to enable nondisruptive capacity and I/O growth for the z990 with no system power down and no associated POR or IPLs.

The Plan Ahead feature involves pre-installation of additional I/O cages, as it is not possible to install an I/O cage concurrently. CUoD for processors can add, concurrently, more CPs, IFLs, ICFs, and zAAPs to a z9 CPC by assigning available spare PUs via LIC-CC. Depending on the quantity of the additional CPs, IFLs, ICFs, and zAAPs in the upgrade, additional books may be required and can be concurrently installed before the LIC-CC enablement.

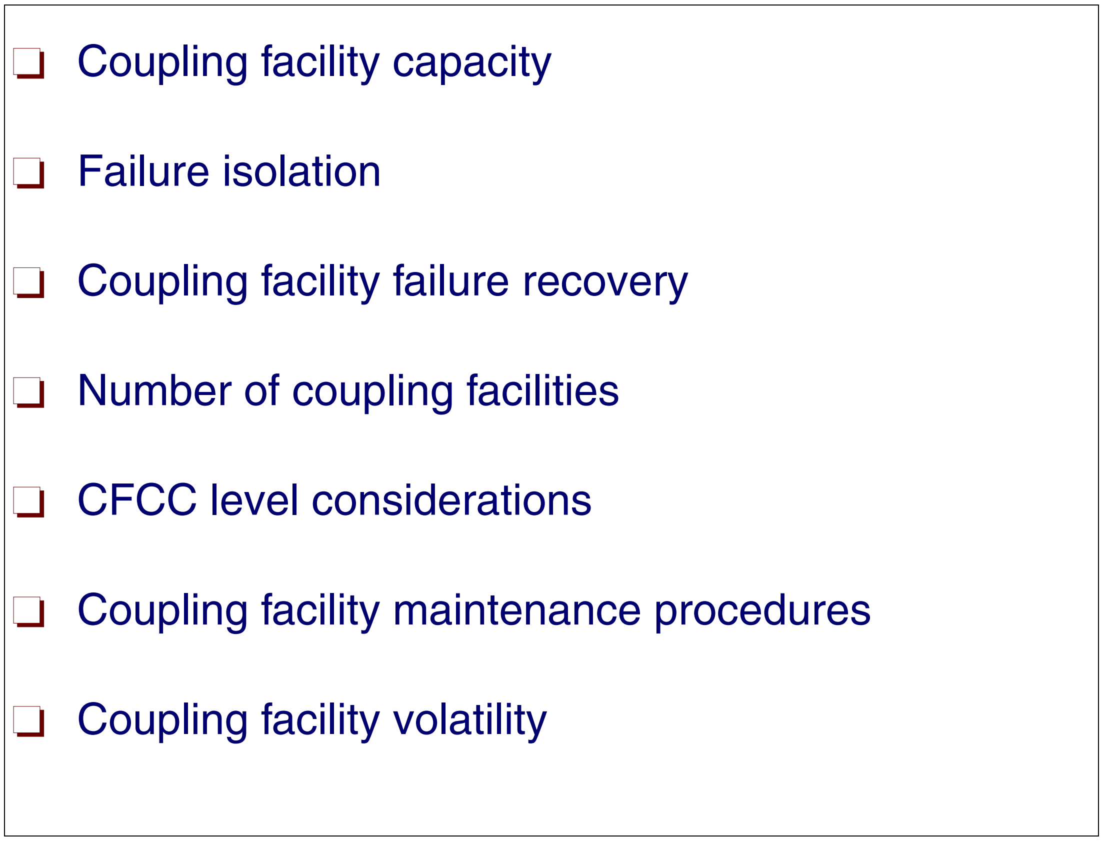
Capacity Backup

Capacity BackUp (CBU) is offered with the z9 CPC servers to provide reserved emergency backup processor capacity for unplanned situations where customers have lost capacity in another part of their establishment and want to recover by adding the reserved capacity on a designated z9 CPC.

CBU is the quick, temporary activation of Central Processors (CPs), up to 90 days, in the face of a loss of customer processing capacity due to an emergency or disaster/recovery situation.

When the CBU activated capacity is no longer required, its removal is nondisruptive. If CBU is activated on a z9 CPC, other hardware upgrades/MES are restricted. With the exception of memory and channels, LIC-CC enabled features, such as CPs, ICFs, IFLs, and zAAPs, can be ordered, but not enabled until the CBU upgrade is deactivated.

8.7 Hardware - coupling facilities



- ❑ Coupling facility capacity
- ❑ Failure isolation
- ❑ Coupling facility failure recovery
- ❑ Number of coupling facilities
- ❑ CFCC level considerations
- ❑ Coupling facility maintenance procedures
- ❑ Coupling facility volatility

Figure 8-7 Hardware - coupling facilities

Coupling facility capacity

In order to be able to deliver acceptable response times, you must have enough capacity in your CFs. In addition to CF PU utilization, you must also consider response times. It is possible for CF PU utilization to be low, but CF response times to be so long that the cost of using the CF becomes unacceptable (this is especially the case if there is a large disparity in speed between the CF CPC and the CPCs the z/OSs are running on).

Failure isolation

One of the most important characteristics of a CF is its location in relation to the z/OS systems that are connected to it. There are some advantages to it being in a stand-alone processor. However, the most important question is whether a single failure can impact both the CF and one or more z/OSs connected to it. A CF in a CPC where none of the other images in that CPC are connected to that CF can provide nearly as good availability as one running in a stand-alone processor.

Coupling facility failure recovery

CFs are different from z/OS LPs in that if a CF fails, it is possible that the CF contents can be re-created in an alternate CF, allowing operations to continue with just a pause in processing.

When planning for CF recovery, you need to consider whether the structures in the failed CF support recovery from a CF failure, and whether the remaining CFs have sufficient capacity to take over from the failed CF.

Number of coupling facilities

As a general rule, you should always have at least two CFs in a production Parallel Sysplex. There are a small number of structures whose loss does not have a significant impact on the sysplex (OPERLOG and LOGREC are two examples), but in most cases, the loss of a structure will have some negative or disabling effect on the sysplex connected systems.

As a result, it is vital that there always be an alternative CF available for a structure to rebuild into in the case of a planned or unplanned CF outage. This is why we recommend that every Parallel Sysplex has at least two CFs, even if you are only exploiting the Resource Sharing capability of the sysplex.

Some larger customers, especially those doing extensive exploitation of data sharing and those that have very high availability requirements are now starting to deploy three CFs in their production Parallel Sysplexes. This provides greater capacity to cope with unexpected workload spikes and rebuild flexibility, and also ensures that there is still no single point of failure, even if one CF is unavailable for some reason.

CFCC level considerations

The level of Coupling Facility Control Code (CFCC) running in your CF determines the functions that CF supports. In 1.13, "Coupling facility LPARs and CFCC code" on page 23 you can see a table with functions and their integration into a specific CFCC level. Prior to z990, CF level upgrades were always disruptive. Therefore, make sure to plan all CF level upgrades well in advance. In fact, it is a good idea to upgrade to the latest CF level available on your CPCs as they become available and as you get the opportunity to do a Power-On-Reset. This ensures that should a need arise for the functionality in that CF level, there is a good chance that it will already be installed and available.

Coupling facility maintenance procedures

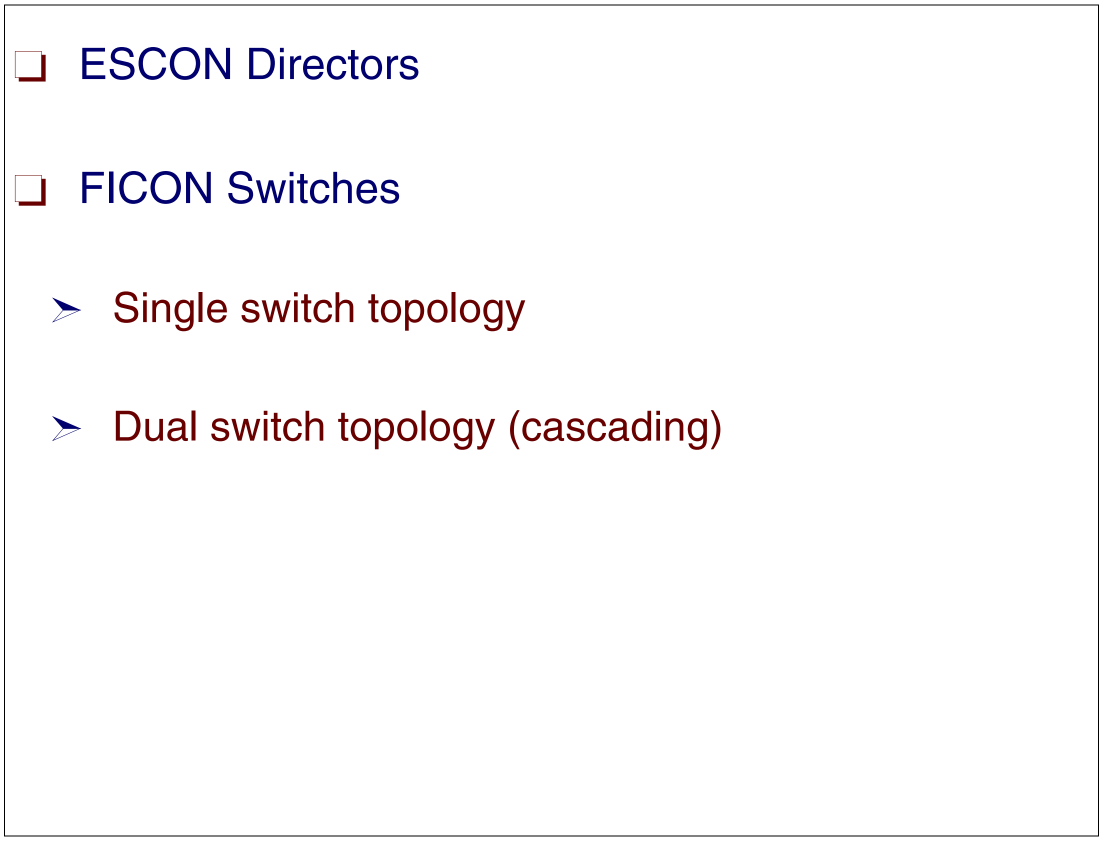
There are various approaches for maintenance of coupling facilities:

- Some customers maintain three CFRM policies: One that contains both CFs, one that contains just one CF, and a third that contains just the other. To empty a CF, they switch to the policy containing just the CF that will remain in service, and then rebuild all the structures that now have a POLICY CHANGE PENDING status.
- Another method is to simply issue a **`SETXCF START,RB,CFNM=cfname,LOC=OTHER`** command for the CF that is to be emptied. If you do this, you will need to move any XCF structures out of the CF with individual **`SETXCF START,RB`** commands.

Coupling facility volatility

An exploiter application that is using the coupling facility might require nonvolatility of the coupling facility storage. Depending on the CPC on which the coupling facility is defined, you might have to provide a backup power supply to make the contents of coupling facility storage nonvolatile across utility power failures. PR/SM Planning Guide describes the processor requirements for coupling facility nonvolatility and the coupling facility control code **`MODE`** command in HMC. This command must be set so that applications using the coupling facility can monitor its nonvolatility status. Coupling facilities also have the capability of a *power save state* for preserving coupling facility storage. Power save state allows the coupling facility storage to be maintained during a utility power outage so that the structures in the storage remain intact. In the server, other LPs that are not coupling facilities are automatically system-reset when the power is lost. The PR/SM Planning Guide provides two scenarios that describe recovery actions in a Parallel Sysplex environment with a coupling facility enabled for the power save state.

8.8 Hardware - switches



- ESCON Directors
- FICON Switches
  - Single switch topology
  - Dual switch topology (cascading)

Figure 8-8 Hardware - switches

ESCON Directors

The ESCON Director (IBM 9032-5) offers any-to-any connectivity for its full range of 248 ESCON ports. All ESCON ports support attachment of channels, control units, serial CTCs, converters (for attachment to convertors two 9032-5 ESCON ports are defined as having a dedicated connection) and other ESCON Directors.

The 9032-5 ESCON also supports up to 16 FICON Bridge cards. These cards convert FICON protocol to ESCON because of the existence of old controllers. Each FICON Bridge card can support 1 FICON port. The internal matrix connections from the FICON Bridge card can be to any ESCON port that logically connects to a ESCON interface control unit or an ESCON serial CTC channels. The external connection to the FICON Bridge card can only be to a S/390 FICON channel in FCV mode. FCV mode is FICON ConVersion mode.

The 9032-5 supports both LED and XDF ESCON ports.

FICON Switches

FICON channels implement the switched point-to-point topology by using FICON switches (F_Ports) to connect FICON channels (N_Ports) to control unit FICON adapters (N_Ports). These connections are established as the server FICON channel Login Link Service FLOGI discovers that its N_Port is connected to a F_Port (switch port) and the control unit FICON port Login Link Service, FLOGI, discovers that its N_Port is also connected to an F_Port (switch port).

This topology allows a much more flexible and dynamic environment and is required to exploit the I/O frame multiplexing capability of FICON channels. All FICON switch connections are dynamic. So static connections, which are possible in an ESCON Director, are not supported in a FICON Director. FICON protocol initially keeps its channel-to-CU path definition approach, which provides controlled access. It does not use a fabric port address discovery (N_Port) approach and requires a known fabric port address, which is the switch destination port.

Switch cascading is a switched point-to-point configuration with more than one dynamic connection in the path. Note that ESCON switched point-to-point configurations also do not support switch cascading, as one of the two possible switch connections must be static. This is a switch chaining extension for ESCON devices. FICON channels do not require any repeater or switch to reach up to 10 km (20 km with RPQ). So initially, only one FICON switch can be used between a FICON channel and a FICON control unit adapter.

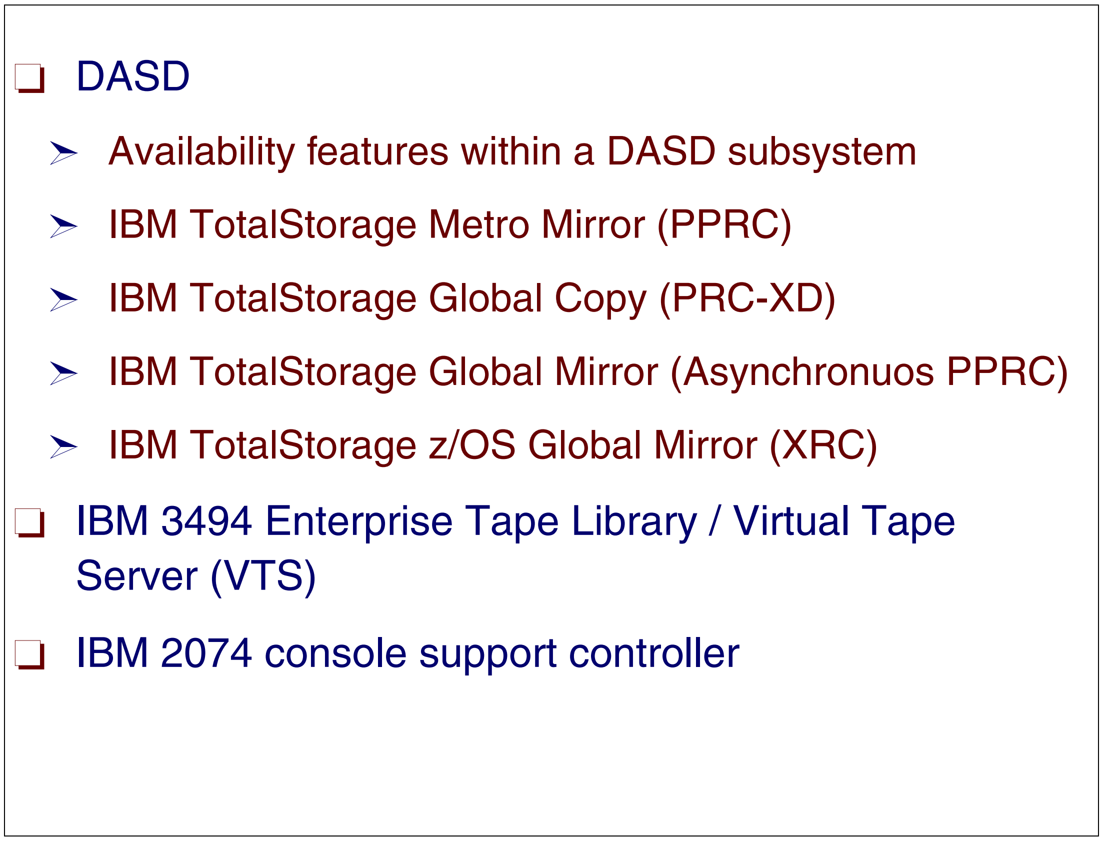
FICON support of Cascading Directors is a switched point-to-point configuration with only two switches in the path. Therefore, all the rules discussed for FICON switched point-to-point also apply for the FICON support for cascaded Directors. FICON channels implement the FICON support for cascaded switches topology by using two dynamic FICON switches (F_Ports) to connect FICON channels (N_Ports) to control unit FICON adapters (N_Ports). These connections are established as the server FICON channel Login Link Service FLOGI to the entry switch of the defined fabric and discovers that its N_Port is connected to an F_Port (switch port) and the control unit FICON port Login Link Service, FLOGI, discovers that its N_Port is also connected to an F_Port (switch port) to the cascaded switch of the fabric.

A connection between both switches in the fabric is performed in such a way that a switch's F_Port (in the entry switch) discovers that it is connected to another F_Port in the other switch (the cascaded switch). The result is that both ports will change their role to support the E_Port functions and form the Inter-Switch Link (ISL) connection.

More than one ISL can be established between both switches in the FICON-supported cascaded director environment. In that case balancing will be done between the multiple ISLs. Since balancing methodologies are vendor-specific, refer to the FICON Director vendor documentation for details.

To ensure data security in the extended environment, both switches have to be in a High Integrity Fabric, which is created by configuring Fabric Binding and Insistent Domain ID in the FC switches. This is checked during channel initialization time. If a 2-byte link address is found for a CU connected to a particular channel, a Query Security Attribute (QSA) is sent to the switch to check whether both are in a high integrity fabric. If it is found, normal channel initialization continues. If the high integrity fabric is not present, no further action is performed.

8.9 Hardware - DASD and other devices



- DASD
  - Availability features within a DASD subsystem
  - IBM TotalStorage Metro Mirror (PPRC)
  - IBM TotalStorage Global Copy (PRC-XD)
  - IBM TotalStorage Global Mirror (Asynchronuos PPRC)
  - IBM TotalStorage z/OS Global Mirror (XRC)
- IBM 3494 Enterprise Tape Library / Virtual Tape Server (VTS)
- IBM 2074 console support controller

Figure 8-9 Hardware - DASD and other devices

DASD

In a continuous availability environment you must configure a DASD subsystem in two ways:

- Create a configuration that will survive the loss of any single component within the subsystem.
- Configure your storage subsystem such that your data can survive, or at least recover from, a complete loss of the whole subsystem.

Features in today's storage subsystems like the IBM DS8000, DS6000™ or Enterprise Storage Server (ESS) support this. Some of them are standard features, some are optional, and some are not available on all storage subsystem types. They are:

- Independent dual power feeds
- N+1 power supply technology/hot swappable power supplies, fans
- N+1 cooling
- Battery backup
- Non-Volatile Subsystem cache, to protect writes that have not been hardened to DASD yet
- Nondisruptive maintenance
- Concurrent LIC activation
- Concurrent repair and replace actions

- RAID architecture
- Redundant microprocessors and data paths
- Concurrent upgrade support (that is, ability to add disks while subsystem is online)
- Redundant shared memory
- Spare disk drives
- FlashCopy/SnapShot/Concurrent copy
- Remote Copy (PPRC and/or XRC)
- HyperSwap support (GDPS/PPRC function)

IBM TotalStorage Metro Mirror (Synchronous PPRC)

Metro Mirror is a remote-mirroring technique for all zSeries and z9 servers. It is designed to constantly maintain an up-to-date copy of the local application data at a remote site that is within the metropolitan area (typically up to 20 km away using DWDM). With synchronous mirroring techniques, data currency is maintained between sites, though the distance can have some impact on performance. Metro Mirror is used primarily as part of a business continuance solution for protecting data against disk storage system loss or complete site failure.

IBM TotalStorage Global Copy (Asynchronous PPRC-XD)

Global Copy is an asynchronous remote copy function for z/OS systems for longer distances than are possible with Metro Mirror. With Global Copy, write operations complete on the primary storage system before they are received by the secondary system. This capability is designed to prevent the primary system's performance from being affected by wait-time from writes on the secondary system. Therefore, the primary and secondary copies can be separated by any distance. This function is appropriate for remote data migration, off-site backups, and transmission of inactive database logs at virtually unlimited distances.

IBM TotalStorage Global Mirror (Asynchronous PPRC)

Global Mirror copying provides a two-site extended distance remote mirroring function for z/OS and servers. With Global Mirror, the data that the host writes to the storage unit at the local site is asynchronously shadowed to the storage unit at the remote site. A consistent copy of the data is then automatically maintained on the storage unit at the remote site. This two site data mirroring function is designed to provide a high-performance, cost-effective global distance data replication and disaster recovery solution.

IBM TotalStorage z/OS Global Mirror (Extended Remote Copy XRC)

z/OS Global Mirror is a remote data mirroring function available for z/OS. It maintains a copy of the data asynchronously at a remote location over unlimited distances. z/OS Global Mirror is well suited for large zSeries server workloads and can be used for business continuance solutions, workload movement and data migration.

IBM 3494 Enterprise Tape Library/Virtual Tape Server (VTS)

The IBM TotalStorage 3494 tape library is an excellent solution for large storage requirements. The 3494 tape library consists of individual frame units for modular expansion that provides a wide range of configurations. This flexibility enables organizations to start small and grow in an affordable and incremental manner without disregarding the aspects of availability.

- When configuring your Tape Library or VTS for high availability, make sure you configure enough paths to the device. If possible, configure paths through different ESCON Directors. If running in an LPAR environment, use EMIF to reduce the number of required channels, but be aware of implications of using EMIF: if one channel fails, it could affect more than one LPAR.

- When installing a 3494 Tape Library, plan for expansion of the tape library with additional frames. Plan to use the High Availability feature of 3494 Tape Library.

IBM 2074 console support controller

The IBM 2074 is a replacement for local, non-SNA 3174 Control Units. The 2074 is a shared control unit that can be used simultaneously by multiple LPARs, using EMIF-capable channels. For example, a single 2074 might provide z/OS consoles (and TSO sessions) for ten LPARs. The LPARs might be in a single zSeries system or spread across multiple zSeries systems. Configuring a high-availability environment with 2074s would mean that you need at least two 2074s. You can find more information about the functions of a 2074 in 5.5, “2074 console support controller configuration” on page 245.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 372. Note that some of the documents referenced here may be available in softcopy only.

- *Server Time Protocol Planning Guide*, SG24-7280-02
- *Systems Programmer's Guide to z/OS System Logger*, SG24-6898
- *zSeries Connectivity Handbook,* SG24-5444
- *Systems Programmer's Guide to Resource Recovery Services (RRS),* SG24-6980
- *Introducing the IBM 2074 Control Unit,* SG24-5966
- *Achieving the Highest Levels of Parallel Sysplex Availability,* SG24-6061
- *OSA-Express Integrated Console Controller Implementation Guide,* SG24-6364
- *Systems Programmer's Guide to: z/OS System Logger,* SG24-6898

Other publications

These publications are also relevant as further information sources:

- *z/OS MVS Setting Up a Sysplex,* SA22-7625
- *z/OS MVS Programming: Resource Recovery,* SA22-7616
- *Processor Resource/Systems Manager Planning Guide,* SB10-7036
- *HCD Planning,* GA22-7525
- *System-Managed CF Structure Duplexing,* GM13-0103
- *z/OS MVS Planning: Operations,* SA22-7601
- *z/OS MVS Sysplex Services Guide*, SA22-7617
- *z/OS Parallel Sysplex Services Reference*, SA22-7618
- *z/OS MVS Routing and Descriptor Codes,* SA22-7624
- *z/OS System Commands,* SA22-7627
- *z/OS and OS390 Managed System Infrastructure for Operations: Setting Up and Using,* SC33-7968
- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS MVS: Installation Exits*, SA22-7593
- *z/OS MVS Diagnosis: Reference,* GA22-7588
- *Server Time Protocol Planning Guide*, SG24-7280
- *Server Time Protocol Implementation Guide*, SG24-7281
- *IBM System z Connectivity Handbook*, SG24-5444

- *z/OS MVS Programming: Authorized Assembler Services Guide*, SA22-7608

Online resources

These web sites are also relevant as further information sources:

- z/OS home page

  http://www-1.ibm.com/servers/eserver/zseries/zos/
- For GDPS information, refer to the GDPS home page at:

  http://www-1.ibm.com/servers/eserver/zseries/gdps/
- The following URL provides CF considerations and CFLEVEL information, including z9 functions:

  http://www-1.ibm.com/servers/eserver/zseries/pso/cftable.html
- It is highly recommended to run the coupling facility structure sizer tool (CFSIZER) after doing a CFCC upgrade. The CFSIZER is provided via the following URL:

  http://www-1.ibm.com/servers/eserver/zseries/cfsizer/
- For details to the technical white paper: *System-Managed CF Structure Duplexing,* GM13-0103, which is available at:

  http://www-1.ibm.com/servers/eserver/zseries/library/techpapers/gm130103.html
- For information on availability for a Parallel Sysplex, see:

  http://www-1.ibm.com/servers/eserver/zseries/pso/availability.html
- There are several tools, wizards, and tech papers linked from the IBM Parallel Sysplex URL:

  http://www-1.ibm.com/servers/eserver/zseries/pso/
- Removing a z/OS System from a sysplex on the IBM Parallel Sysplex home page at:

  http://www.ibm.com/s390/pso/removing.html

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

IBM

Redbooks

ABCs of z/OS System Programming: Volume 5

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

ABCs of z/OS System Programming: Volume 5

Base and Parallel Sysplex, GRS, RRS, ARM, sysplex failure management

System Logger, z/OS system operation

GDPS, zSeries availability

The ABCs of z/OS System Programming is an 13-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information that you need to start your research into z/OS and related subjects. If you would like to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection will serve as a powerful technical tool.

The contents of the volumes are as follows:

Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation

Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, SMP/E, Language Environment

Volume 3: Introduction to DFSMS, data set basics storage management hardware and software, catalogs, and DFSMStvs

Volume 4: Communication Server, TCP/IP, and VTAM

Volume 5: Base and Parallel Sysplex, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), Geographically Dispersed Parallel Sysplex (GDPS)

Volume 6: Introduction to security, RACF, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries firewall technologies, LDAP, and Enterprise identity mapping (EIM)

Volume 7: Printing in a z/OS environment, Infoprint Server and Infoprint Central

Volume 8: An introduction to z/OS problem diagnosis

Volume 9: z/OS UNIX System Services

Volume 10: Introduction to z/Architecture, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and HMC

Volume 11: Capacity planning, performance management, WLM, RMF, and SMF

Volume 12: WLM

Volume 13: JES3

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6985-02 ISBN 0738435511