IBM

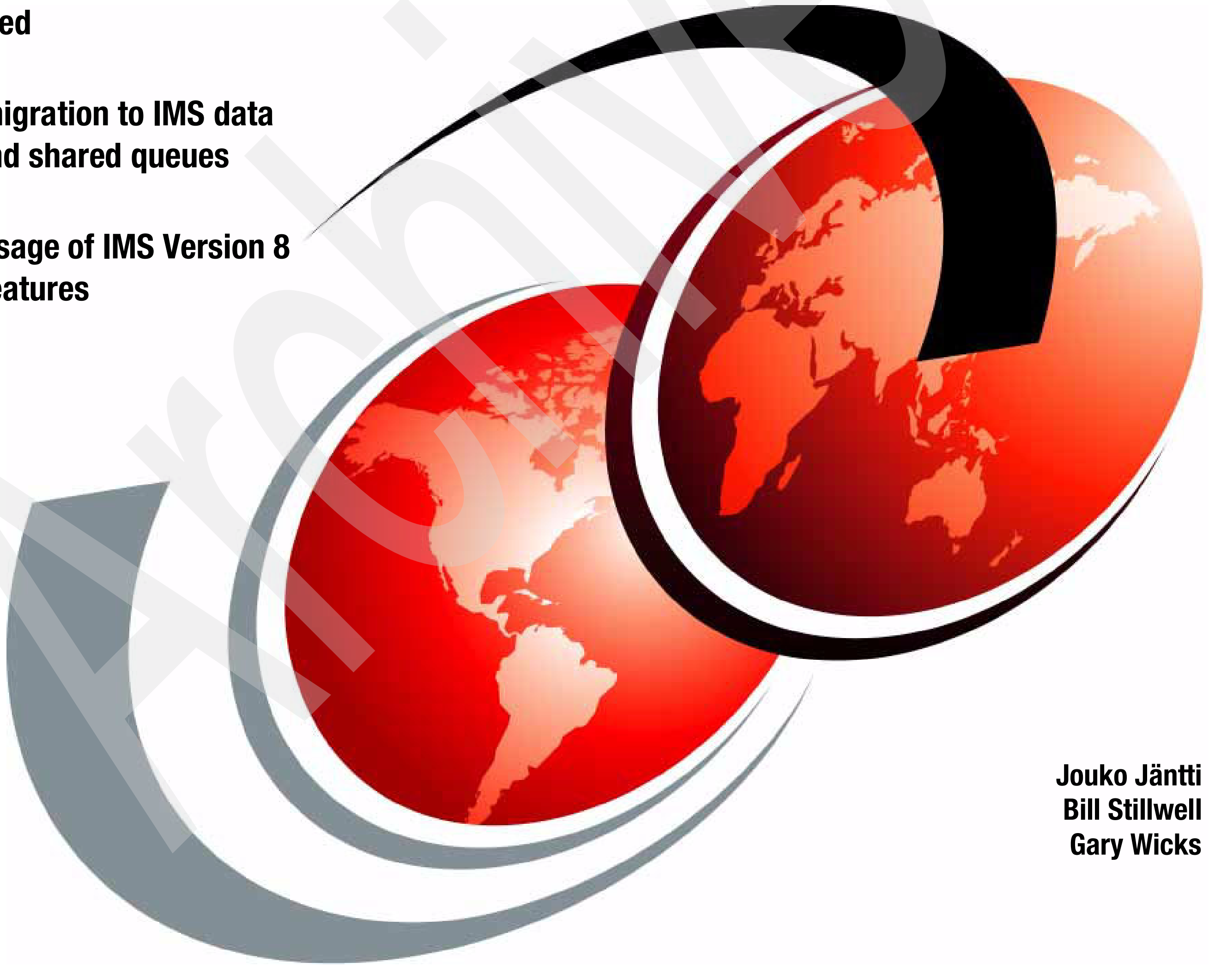
IMS in the Parallel Sysplex

Volume II: Planning the IMSplex

Parallel Sysplex migration planning for IMS updated

Plan the migration to IMS data sharing and shared queues

Plan the usage of IMS Version 8 IMSplex features



Jouko Jäntti
Bill Stillwell
Gary Wicks

ibm.com/redbooks

Redbooks

International Technical Support Organization

IMS in the Parallel Sysplex Volume II: Planning the IMSplex

July 2003

SG24-6928-00

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

First Edition (July 2003)

This edition applies to IMS Version 8 (program number 5655-C56) or later for use with the OS/390 or z/OS operating system.

Contents

Notices

Preface

This IBM Redbook is the second volume of a series of redbooks called IMS™ in the Parallel Sysplex®. These redbooks describe how IMS exploits the Parallel Sysplex functions and how to plan for, implement, and operate IMS systems working together in a Parallel Sysplex. We use the term *IMSplex* to refer to multiple IMSs that are cooperating with each other in a Parallel Sysplex environment to process a common shared workload. Although we generally think of an IMSplex in terms of online environments, an IMSplex can include batch IMS jobs as well as IMS utilities.

IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology, SG24-6908 describes the Parallel Sysplex and how IMS exploits the Parallel Sysplex to provide user services including data sharing, shared queues, VTAM® Generic Resources, Automatic Restart Manager (ARM), and systems management functions. When migrating an IMS system from a single, non-sharing environment to one which invokes some or all of these services, or even when incorporating additional function into an existing IMSplex (for example, upgrading a data sharing system to also use shared queues), the migration process must be carefully planned. *Many decisions and compromises must be made, perhaps even some application or database changes. There will also be changes to system definition and to operational procedures.*

This redbook addresses the development of the migration plan and identifies some of the steps and considerations you might encounter when developing the plan. The result of this exercise is not to *perform* any of the implementation tasks but to *identify* those tasks which must be done, and to *create a plan* for accomplishing them. For example, the plan can identify as a task the establishment of a naming convention for system data sets. The naming convention itself is not a part of the plan, but is a result of implementing the plan.

In this book we present planning considerations for the IMSplex. Separate chapters are devoted to:

- Block level data sharing
- Shared queues
- Connectivity
- Systems management
- The overall IMSplex environment

The other volumes in this series are:

- *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908
- *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Jouko Jäntti is a Project Leader specializing in IMS with the IBM International Technical Support Organization, San Jose Center. He holds a bachelor's degree in Business Information Technology from Helsinki Business Polytechnic, Finland. Before joining the ITSO in September 2001, Jouko worked as an Advisory IT Specialist at IBM Global Services, Finland. Jouko has been working on several e-business projects with customers as a specialist in IMS, WebSphere®, and UNIX on the OS/390® platform. He has also been responsible for the local IMS support for Finnish IMS customers. Prior to joining IBM in 1997, he worked as a Systems Programmer and Transaction Management Specialist in a large Finnish bank for 13 years, and was responsible for the bank's IMS systems.

Bill Stillwell is a Senior Consulting I/T Specialist and has been providing technical support and consulting services to IMS customers as a member of the Dallas Systems Center for 20 years. During that time, he developed expertise in application and database design, IMS performance, Fast Path, data sharing, shared queues, planning for IMS Parallel Sysplex exploitation and migration, DBRC, and database control (DBCTL). He also develops and teaches IBM education and training courses, and is a regular speaker at the annual IMS Technical Conferences in the United States and Europe.

Gary Wicks is a Certified I/T Specialist in Canada. He has 28 years of experience with IBM in software service and is currently on assignment as an IMS Development Group Advocate. He holds a degree in mathematics and physics from the University of Toronto. His areas of expertise include IMS enablement in Parallel Sysplex environments and he has written several IBM Redbooks™ on the subject over the last six years.

Thanks to the following people for their contributions to this project:

Rich Conway
Bob Haimowitz
International Technical Support Organization, Poughkeepsie Center

Rose Levin
Jim Bahls
Mark Harbinski
Claudia Ho
Judy Tse
Sandy Stoob
Pedro Vera
IBM Silicon Valley Laboratory,USA

Juan Jesús Iniesta Martínez
IBM Spain

Knut Kubein
IBM Germany

Suzie Wendler
IBM USA

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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 Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM® Corporation, International Technical Support Organization
  Dept. QXXE Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099

1

Introduction to IMSplex planning

Migrating to an IMSplex environment is a multi-phase process. A comprehensive migration plan must address all of the phases. We have identified four phases which must be considered in your migration plan. You may identify more, or fewer, depending on how you define the components of each phase. The phases we have identified are the following:

- Planning phase
- Preparation phase
- Implementation phase
- Operational phase

This volume deals primarily with the first phase, planning. It tries to identify for you in the form of questions that you must answer or issues you must address, what you have to consider when migrating from a non-sysplex environment to an IMSplex environment. This volume is also useful when you are upgrading your current IMSplex to take advantage of other IMS Parallel Sysplex capabilities, such as migrating from data sharing only to a shared queues environment, or to one that includes the Common Service Layer available with IMS Version 8.

The first volume in this series of three volumes, *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908, describes the IMSplex, its functionality, and its benefits.

The third volume contains the implementation and operational details. For example, you may decide during the planning phase that you want to migrate to shared queues to meet some business requirement. *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929 describes how you implement shared queues, including parameter changes, JCL changes, new address spaces, Coupling Facility structure definitions, and so forth.

1.1 Planning for migration

The following text identifies four phases of a migration process. Note that the planning phase includes plans for each of the other phases.

1.1.1 Planning phase

This phase includes an analysis of the current environment and what your objectives and expectations are for implementing an IMSplex. This can be done by answering a series of questions, examples of which are included in this volume. It then continues with the definition of tasks to be performed during all phases of the migration, including the planning phase itself. Specific planning activities should include:

- Defining your objectives and expectations

  Sometimes, when implementing a new function or feature, or a new technology, you can lose sight of the reasons why you are doing it in the first place, and get too caught up in the technology. The first step in the planning process should be to *document your reasons* for implementing the feature, whether it is data sharing, shared queues, or the Common Service Layer. What problems does it solve, or how does it make operation of the IMSplex easier? Then, throughout the planning process, continually review your objectives for implementing the feature and what your expectations are for each component. When the migration process is complete, come back one more time and see if it meets your expectations.

- Functional planning

  This includes identifying among the features available in an IMSplex, which ones will be implemented in your environment. For each feature, identify the specific functions you will exploit and your objectives and expectations for that function; what problems it solves; or how it improves performance, capacity, availability, manageability, scalability, connectivity, and any other requirements.

- Configuration planning

  Define what your *final IMSplex configuration* will be, and any *intermediate configurations*. For example, your first configuration may be just data sharing, then shared queues, and finally the Common Service Layer. The configuration should include the Parallel Sysplex hardware and software, subsystem software, connectivity to the IMSplex, and a decision about where applications will run. Of course that decision may be that each application runs on every IMS image.

  You should also include a plan for degraded mode processing if not all components are available. For example, if the LPAR containing one of your IMS control regions is down for an extended period, what will you do?

- Security planning

  There are many aspects of security which have different connotations in an IMSplex. Examples are joining the IMSplex, connecting to structures, and submitting commands through the Operations Manager interface. Shared queues introduce some security issues concerning authorization processing when transaction entered on one IMS executes on another. The migration plan should include tasks which identify how security will be implemented in the IMSplex.

- Implementation planning

  Implementation planning should include both preparing the environment for the IMSplex, and the actual implementation of the IMSplex in that environment.

The preparation phase is where you get the operating system and IMS environment ready for implementation. This includes hardware, software, libraries, and any changes required to procedures, applications, and databases. This goes on while the current environment is still operational and it can get very confusing keeping libraries straight, installing new hardware and software, preparing changes to procedures, changing applications, and so on. A well documented plan for how this will occur can save a lot of problems later.

Define the tasks to implement your configuration (including any intermediate configurations) *through all phases of the migration*. Plans for the implementation phase must include (if necessary) termination of the existing environment and initiation of the new environment (for example, IMS shutdown and a cold start). Do not forget to include the end-users in your implementation plan.

- Operational planning

  Identify and document procedures for the *operation of the IMSplex environment*. This should include network (connectivity), systems (OS/390 or z/OS®), IMS (for example, the MTO or SPOC) and end-user operations. You must also include procedures for recovery where they differ from recovery in the existing (non-sysplex) environment. And do not forget any backout procedures.

1.1.2 Preparation phase

During this phase, and following the plan developed in the planning phase, prepare the environment for the first configuration. This will include making sure that all hardware and software prerequisites are in place, and that the OS/390 or z/OS environment is preconditioned for the IMSplex. Create and update procedures and PROCLIB members, and update operational and end-user procedures. In other words, make the environment *ready for implementation*. Make sure your plan is comprehensive - do not leave anything out.

1.1.3 Implementation phase

This phase is the actual *cutover to a production environment*. It will probably include shutting down the existing environment and restarting the new environment. End-users have to switch too. This could be the tricky part, so everyone has to be on the same page, including the end-users. Make sure your plan is thorough.

1.1.4 Operational phase

Everything is up and running. Hopefully, just as you planned; but in case something goes wrong, this phase will *also include recovery and restart* of the sysplex, operating system, IMS, applications, databases, network sessions, and almost anything that Murphy's Law covers. Make sure your plan covers it all.

Most users will not go directly from a plan to a production environment. They will first create a test environment (perhaps several) and a development environment (perhaps several). Although this document is written as though all of the planning were for the production environment, similar planning, preparation, implementation, and operational phases exist for the test and development environments, with the same kinds of questions and tasks. The answers and tasks may be different, but the questions are the same.

1.2 Planning phase

The purpose of this phase is to identify and document where you are coming from, where you are going, what you will do if there are failures, to determine how the plan will be created, who has responsibility, what will be its content and format, what planning or project management tools will be used, and finally to develop the plan itself.

Throughout the planning process, there are several concepts that must be in the forefront of all your planning activities. You might recognize fewer or more, but each has a purpose, and that purpose must be satisfied in any planning process.

1.2.1 Understand the existing environment

In order to arrive at a target environment that meets your objectives and expectations, you must first understand the existing environment. This includes knowing, for each application system, the resource requirements (such as CPU or I/O), service level requirements, schedules and workloads, connections to the network and to other subsystems, and the *reasons for migrating* (for instance, reduced costs or improved performance, capacity, or availability). You should also identify any *inhibitors to migration*, such as non-sharable resources.

The assumption here is that the existing environment is being replaced (or upgraded) for one or more reasons, such as better performance, availability, operability, or perhaps reduced cost. The target environment must continue to provide the equivalent function with performance and availability at least as good as the existing environment. So, in order to define a target environment which will do this, it is first necessary to understand the existing environment. The last thing you want to do is to get into production in the new environment and then find out that some process, application, or database has some unique requirements that are not met by the new IMSplex environment.

The following describes some of the characteristics of the existing environment that should be known before defining the target.

- What is the current workload?

  This should be documented in terms that will facilitate the *distribution of that workload* over two or more processors and should include transaction volumes as well as batch and BMP and support jobs such as image copies, reorganizations, and so forth.

- Who are the heavy resource users?

  Which of the transactions or batch processes identified in the previous question require the greatest number of CPU or I/O resources, and which transactions are the highest volumes? Are there any existing problems (for example, locking conflicts) that must be accounted for in the new environment? It might be necessary to make special provisions for them in the target environment.

- What are the service level commitments?

  What agreements exist for transaction response times, batch elapsed times, and availability? How about recovery times following database or system failure? Are users billed according to the resources they use?

- What are the network connections to the existing IMS?

  Who are the end-users and how are they connected? VTAM? TCP/IP using OTMA? Any changes expected within the planning horizon?

- To what systems is the existing IMS connected?

  This should include other IMSs, DB2s, CICSs, IMS Connect, WebSphere MQ, DB2® stored procedures, WebSphere Application Server programs, and any other intelligent systems or devices that might be sensitive to the identity of the IMS system to which they are connected.

- What are the online and batch schedules?

What are the hours of availability for online access and what is the batch window (if there is one)? When and how are database maintenance functions (image copy, reorganization, other) performed?

- Are there any batch or online dependencies?

  Are there *sequences of processes* that must be maintained in the target environment. For example, transactions defined as SERIAL, implying a FIFO processing sequence, should be identified. Batch jobs that produce input to other batch jobs must be scheduled in the right order.

- Are any user exits sensitive to the identity of the IMS system on which they execute?

  Look particularly at transaction manager exits and system exits as there will be multiple transaction managers with different IDs, connected, perhaps, to different subsystems (for instance, different CICSs, or different DB2s) and with only part of the original network.

- Do any user-written or vendor-written programs or tools process the IMS logs?

  The logs will be quite different if you are implementing shared queues, with each log containing only part of the information that was on the original single-image log. Database recovery in a data sharing environment requires logs from multiple IMSs.

- What are the business-critical applications?

  If one component of the target environment fails (for instance, one of two IMS systems) and can't be immediately restarted, it might be necessary to quiesce less important work on the surviving system in order to shift the entire critical workload to that system. It might also be necessary to shift part (or all) of the network to the surviving system. This should be addressed by your *degraded mode* environment.

- Are there any non-sharable resources?

  An installation can choose not to share some databases, or some resources may not be sharable (for example, main storage databases). These must be identified and plans made for their migration to the target system.

1.2.2 Define and redefine the target environment

A major part of developing a migration plan is to choose the configuration to which the migration will be done. This configuration is affected by the *reasons* for making the migration. The migration to IMS data sharing and shared queues with Parallel Sysplex can be used to provide multiple benefits. These include, but are not limited to:

- New function
- Improved availability
- Improved performance
- Increased capacity
- Incremental growth capability
- Operational flexibility
- Reduced cost per transaction

Always keep in mind the target environment. You may begin with some idea of what that configuration is, but you should continually review your first ideas to see if that configuration must change as you identify the function that meets your objectives. There will probably be multiple target environments, which include test, development, and production environments.

Defining the environment should include the configuration of the Parallel Sysplex itself, the number of IMS subsystems in the IMSplex, composition of the VTAM generic resource group and/or OTMA group, the OS/390 or z/OS system on which each IMS will run, other subsystems outside of the target IMSplex environment to which the target IMSs will connect (for example, other IMSs and IMSplexes connected by MSC or ISC), the Coupling Facilities to be used, and which systems will be used for various purposes. Will all the IMSs be defined the same (cloned)? On which systems will IMS BMPs be run, or if applications are to be partitioned, on which systems will they run? Where will IMS batch (not BMP) jobs run? The answers to some of these questions may not be known until well into the planning, and perhaps even the implementation phase. Be sure the target environment satisfies the reasons for migration.

The elements of the target configuration, along with some of the questions you must answer, are as follows:

- Processors (CPCs) and LPARs

  How many and which LPARs will IMS be active in?

- OS/390 or z/OS systems

  The OS/390 or z/OS systems in the target configuration should be identified by the processors and LPARs on which they run and the types of work they will support. The types of work include the IMS subsystems and support processes they will handle.

- Coupling Facilities

  The Coupling Facilities and Coupling Facility structures to support IMS should be identified. How many Coupling Facilities are available for IMS structures? What type of processor are they (for example, 967x or zSeries®)? What is the CF Level of the Coupling Facility Control Code (CFCC)? Structures required in an IMSplex may include:

  – Data sharing structures:

    • IRLM (lock structure)
    • VSAM cache structure (directory-only cache structure)
    • OSAM cache structure (directory-only or store-through cache structure)
    • Shared DEDB VSO structure(s) (store-in cache structures)

  – Shared queues:

    • Shared message queue and shared EMH queue primary and overflow structures (list structures)

    • System Logger structures (list structures)

  – Common Service Layer

    • Resource structure (list structure)

- IMS subsystems

  These are IMS online systems (either IMS Transaction Manager, DC Control, or Database Control), IMS batch jobs, and IMS support processes, such as utilities. Identify where IMS batch jobs will execute. Support processes may include database image copies, database recovery, database reorganization, log archiving, and message queue management. The OS/390 or z/OS systems on which they will run should be identified.

- IMS Transaction Manager connectivity

  For work to execute in an IMS environment, the user must be able to *connect* to an IMS control region. When there was only one control region, it was fairly obvious which one to connect to, and what to do when that one control region was not available. BMPs always connected to the one IMS control region. When there are multiple control regions, it is necessary to define how work will flow to *each IMS in the IMSplex*. This may include:

  – SNA network

    These are the "traditional" means of gaining access to an IMS transaction manager. These devices may be real (for example, a 3270 terminal) or simulated (for example, Telnet (TN3270)). End-users may gain access through VTAM Generic Resources, or log on directly to one IMS. Different types of logical units that are able to access IMS are the following:

  - LU0 (SLUTYPEP and FINANCE)
  - LU1 and LU2
  - LU6.1 (ISC)
  - LU 6.2 (APPC)

- Open transaction manager access (OTMA)

  Non-SNA access to an IMS TM is provided through open transaction manager access (OTMA). This usually means through IMS Connect or WebSphere MQ, but may also be other OTMA clients; for example, applications invoking the callable OTMA interface (OTMA C/I).

- Multiple systems coupling (MSC)

  IMS may be connected to other IMSs or other IMSplexes through multiple systems coupling (MSC). In some cases, when those connected IMSs are in close proximity, the MSC links may be replaced by shared queues. When they are remote, one of the IMSs must have the active MSC connection while others must be able to take over in case of a failure.

- Session managers

  If users are currently connected to one IMS through a session manager, connection to multiple IMSs must be provided for.

► IMS Database Manager connectivity

IMS provides connectivity to its databases through means other than the submission and execution of a transaction. The target configuration must make provisions for this, and may include:

- IMS batch

  If you currently run IMS batch jobs in a non-data sharing mode by taking the databases offline, you may want to consider changing those jobs to either data sharing or BMP.

- BMP

  BMPs which currently run under a single IMS may now execute under one of several. Since you may not always know on which LPAR a BMP may be scheduled, the IMSID may need to be changed to a group name, allowing the BMP to connect to whichever IMS is on the same LPAR.

- CICS® w/DBCTL

  If CICS application owning regions (AOR) are currently connected to an IMS DBCTL control region, then the target environment may require new AORs to connect to the IMSs on the new LPARs.

- ODBA

  Application programs which do not run in an IMS batch or online region can access IMS databases using the open database access (ODBA) interface. Typically, these applications run as:

  - DB2 stored procedures
  - WebSphere Application Server programs
  - Other ODBA connectors

► DB2

If IMS is connected to DB2, then multiple IMSs on multiple LPARs will require connectivity to multiple DB2s. DB2 data sharing may be required.

- Other subsystems and address spaces

  These are any other subsystems or address spaces to which IMS will connect or communicate.

Identify basic changes to the existing IMS environment

The current definitions of the IMS systems must, obviously, be modified to accommodate the requirements of the target environment. However, there may be other changes that are not so obvious. For example, transactions defined as SERIAL in a single IMS may require changes when run in a shared queues environment. Scheduling classes, parallel limit counts, and dependent region scheduling parameters may have to be changed for shared queues. Changes to application processing may be required when running in a data sharing environment. The effects of resource management when IMS is running with the Common Service Layer will dictate changes. Much of this document is aimed at helping you identify these changes.

Identify changes to operational procedures

Operators (both IMS and OS/390), database and system support personnel, and end-users will all have changes to their daily procedures when operating in the target environment. These changes and the people they affect must be identified and included in the migration plan.

1.2.3 Define degraded mode environment

You must also decide what you will do if something fails. Since a Parallel Sysplex consists of multiple OS/390 or z/OS and IMS systems, an installation should plan what it will do if one or more components fail. For example, there may be certain applications or systems that are more critical to the business, and therefore should have preference when there is an outage of part of the system. This is called *degraded mode* processing.

During this planning phase, you should determine both the *processing and business impact* of the failure of any component of the target environment. Identify those applications which must be given priority in a degraded processing environment. You must also consider what users who are connected to a failed component should do (for example, log on to another IMS, or wait for the IMS restart).

Some tasks which should be included are:

- Perform a *component failure impact analysis* (CFIA) for critical components
- Prioritize applications for degraded mode processing
- Identify network terminals and connections to be reconnected to another system if one system fails

1.2.4 Develop the plan

The plan should recognize all phases of the migration process: planning (yes, you need a plan to plan), preparation, implementation, operation. Although this document does not prescribe a format for the migration plan, the following elements should be included:

Tasks	What must be done?
Responsibility	Who is responsible to see that it gets done?
Dependencies	Is any task a prerequisite to another task?
Duration	How long should each task take?
Schedule	When must it be done - start/complete/drop-dead dates?
Status	A mechanism for monitoring progress?

During this phase you should develop a list of tasks that you are able to identify, and this list should be a part of the migration plan.

1.3 Preparation phase

Many of the tasks identified in the migration plan are implemented during the preparation phase. The plan may say, for example, that a naming convention must be established for system data sets. During this phase, that naming convention will be developed. Or the plan may say that operational procedures must be updated. During this phase, those procedures are updated.

Some of the tasks in this phase will be *decision*-type tasks (for instance, how many copies of SDFSRESL do I want?). Others will be *implementing* some of these decisions (such as, making two copies of SDFSRESL). At the conclusion of this phase, you are ready to migrate to your target environment.

1.4 Implementation phase

The next phase in the migration process is the actual implementation. That is, the existing environment will be converted to an operational IMSplex environment. The actual *implementation plan* will probably be produced as part of the preparation phase as it is unlikely that enough information will be available during planning to generate this plan.

1.5 Operational phase

Finally, after implementation, we have the operational phase. Hopefully, if everything before has been successfully and completely done, this will be business-as-usual. However, if this is not the case, then the operational plan should include procedures for falling back to some previous level, or to some less-than-full IMSplex implementation.

1.6 Now what?

The remainder of this document provides an overview of the Parallel Sysplex and the services offered to authorized programs such as IMS. It then continues with the planning considerations and tasks when migrating from a single IMS environment to an IMSplex environment. Each function supported in the IMSplex is presented in one or more chapters, and include:

- Introduction to the Parallel Sysplex

  This is a brief high level description of the objectives and benefits of a Parallel Sysplex, the hardware and software components, and the services provided by cross-system coupling facility (XCF) and cross-system extended services (XES). We also identify (again at a high level) which of these services IMS exploits to provide the user with the functionality described in the following chapters. It is assumed that you, as the IMS user, are responsible for the implementation of the IMSplex, not the Parallel Sysplex.

- Block level data sharing

  The first step in migrating to a Parallel Sysplex environment is usually to implement block level data sharing. When enabled, IMS batch and/or online applications running anywhere in the IMSplex can access all of the data in the shared IMS databases, allowing the user to distribute the workload according to the business requirements.

- Connectivity

  Once shared data is available from multiple IMS members in a Parallel Sysplex, the user must plan for and implement techniques for distributing the batch and transaction workload across the multiple IMSs in the IMSplex.

- Shared queues

  Shared queues is one technique for distributing the IMS transaction workload across multiple members of the IMSplex. When enabled, any transaction can be entered from the network to any IMS in the IMSplex and be executed by any IMS in the shared queues group. When one IMS is not available, or too busy to process a transaction, others can step in and pick up the work.

- Common Service Layer

  The Common Service Layer (CSL), introduced in IMS Version 8, improves the user's ability to manage an IMSplex. It includes improvements in both operations and recovery, as well as a capability to coordinate the online change process across all IMSs in the IMSplex.

2

Introduction to the Parallel Sysplex

This chapter provides an overview of the Parallel Sysplex and includes brief descriptions of the objectives and benefits of a Parallel Sysplex, its hardware components, including the Coupling Facility hardware itself, and its software components, including XCF and XES, and the sysplex services provided by these components. The descriptions in this chapter, or even in this book, are by no means comprehensive, and at times the description of the sysplex services are abbreviated for ease of understanding. Some of the services available to sysplex users are not described at all if they do not play a role in the IMSplex.

Throughout the description of the Parallel Sysplex services, we have tried to use IMS's implementation of Parallel Sysplex services as examples.

2.1 Introduction

IMS/ESA® Version 5 was the first release of IMS to exploit the Parallel Sysplex. Version 5 dramatically improved the way block level data sharing was implemented by taking advantage of Parallel Sysplex services for communications and data sharing offered by the OS/390 components cross-system coupling facility (XCF), cross-system extended services (XES), and a new hardware component called the Coupling Facility (CF). This hardware is not to be confused with XCF, which is software.

Since that time, each release of IMS has more fully exploited Parallel Sysplex services. There is every indication that future releases of IMS will add even more Parallel Sysplex exploitation to enable a more dynamic, available, manageable, scalable, and well performing environment for database, transaction, and systems management. To be able to take advantage of these functions, and any future enhancements, it is important that you have at least a cursory understanding of what a Parallel Sysplex is, how it works, and why it is important to IMS and to your installation.

While this is simply a very high level overview of the Parallel Sysplex, more detailed information is provided in a series of IBM publications on the Parallel Sysplex and Parallel Sysplex services. These publications can be found by searching for the term *sysplex* on the IBM publications center Web site:

http://ehone.ibm.com/public/applications/publications/cgibin/pbi.cgi

2.2 What is a Parallel Sysplex?

In 1990, IBM announced the *sysplex* as the strategic direction for large systems computing environments and described it as "... *a collection of MVS systems that cooperate, using certain hardware and software products, to process work*." The term sysplex is derived from the words ***SYS***tems com***PLEX***. At the time of this first announcement, and for several years thereafter, there was no Parallel Sysplex - only a *base sysplex*, which provided improvements in inter-processor communications between systems and any subsystems wishing to exploit those services, but no data sharing services.

The Parallel Sysplex was introduced later in the 1990s and added hardware and software components to provide for *sysplex data sharing*. In this context, data sharing means the ability for sysplex member systems and subsystems to store data into, and retrieve data from, a common area. *IMS data sharing* is a function of IMS subsystems which exploits sysplex data sharing services to share IMS databases. Since IMS did not exploit the base sysplex, from this point on we will use the term sysplex as shorthand notation for a Parallel Sysplex.

In this series of redbooks, the term sysplex, when used alone, refers to a Parallel Sysplex. When the sysplex is not a Parallel Sysplex, and the distinction is important, we use the term base sysplex. We use the term IMSplex to refer to one or more IMSs cooperating in a single data sharing environment.

2.2.1 Some components and terminology of a Parallel Sysplex

Parallel Sysplex implementation requires a number of hardware, software, and microcode components, and has its own terminology. Here are some of the more visible components of a Parallel Sysplex, and some of the Parallel Sysplex terminology you will encounter:

- Central processing complex (CPC)

  A typical sysplex includes multiple CPCs (sometimes known as a CEC, or central electronic complex), although it is possible to have a single-CPC sysplex. These include selected models of the ES/9000®, S/390®, and z900 processors. For the sake of brevity, in these redbooks we use the term S/390 to refer to any processor which is sysplex capable. As well, we use the term OS/390 to refer to an MVS/ESA™, OS/390, or z/OS operating system.

- Common time reference (sysplex timer)

  Because the OS/390 systems and subsystems running in the sysplex cooperate with other instances of the same subsystem, or with other subsystems and programs, it is critical that they all share a common time of day. For example, two IMS systems sharing data include the time that data is updated in their respective but separate logs. If those times were not synchronized, database recovery using these times would be difficult. All CPCs in a sysplex are connected to a sysplex timer (IBM 9037) which provides this synchronization across all CPC time-of-day (TOD) clocks. The sysplex extended TOD architecture also guarantees that two applications requesting a TOD value from any CPC in the sysplex will each receive unique values — also important for IMS data sharing recovery.

- Signaling paths

  Many of the services provided by the sysplex require communications between participating OS/390 systems and subsystems. To do this, at least two signaling paths are required between the CPCs in the sysplex: one inbound path and one outbound path. These paths may be defined using one or more of the following:

  – ESCON® channels operating in CTC mode
  – The 3088 Multisystem Channel Communication Unit
  – Coupling Facility with list structures

  Structures will be the subject of much discussion later in this document. The signaling structures are a Parallel Sysplex implementation requirement, not specific to IMS, and so no more is said about the signaling structures themselves.

- Coupling Facility (CF)

  The Coupling Facility hardware, not to be confused with cross-system coupling facility (XCF) software, is itself a processor that provides the medium on which shared data may be stored and retrieved within a Parallel Sysplex. The CF processor itself can be a logical partition (LPAR) in a S/390 processor but without the channels (and peripherals), which are part of the LPARs running your OS/390 operating systems.

  There are two basic types of Coupling Facility: *internal* and *stand-alone*. The stand-alone CF is a 9674 processor and is external to the CPC. During the last few years, most CPCs have been shipped with internal CFs, which need only be activated to be used. A z900 box, for example, may come with multiple general use CPUs, plus additional CPUs which can be configured and activated only as Coupling Facility LPARs. It is also possible to configure one or more S/390 LPARs as a Coupling Facility. From an end-user (for example, IMS) functional point of view, there is no difference between the different types of Coupling Facility.

- Coupling Facility Control Code (CFCC)

  The Coupling Facility equivalent of an operating system is the Coupling Facility Control Code (CFCC) which is implemented only as microcode in the CF itself and runs only in LPAR mode. No software runs in the CF processor. Different releases of the CFCC are referred to as levels. For example, the latest CFCC level (at the time of this writing) is CFCC Level 12. Some data sharing functions have minimum CFCC level requirements, similar to some OS/390 or IMS functions that have minimum version and release level requirements.

- CPC-to-CF links

  The CF link is the functional equivalent of a CPC channel connecting the CPC to its peripheral devices, such as DASD. It provides the physical path for sending data to, and receiving data from, the Coupling Facility. Sometimes you will see a physical link referred to as a *path* or a *Coupling Facility channel.* RMF™, for example, may report *path busy* conditions.

- Coupling Facility link adapters

  CF link adapters are microcode on either end of the CF link which provide the functionality to send and receive messages (data) between the CPC and the CF. CF link adapters have buffers into which data is placed for sending. These buffers will often be referred to as *subchannels*. For example, some of the RMF reports may show *subchannel busy* conditions. This means that the buffers were full at the time a request was made and the requestor had to wait.

- CF-to-CF links

  Recent versions of the sysplex configuration may include another type of link called a CF-to-CF link. These links are used by some sysplex data sharing services, such as structure duplexing, to synchronize updates.

- Couple data sets (CDS)

  Although all CPCs have DASD containing many system data sets, there is one set of data sets required to define the sysplex and its services. These data sets are called *couple data sets* (CDSs). These data sets, which are shared by all members of the Parallel Sysplex, describe the sysplex environment and contain *policies* which are used to describe how the Parallel Sysplex is to be managed by OS/390 or z/OS. In a Parallel Sysplex, there are six couple data sets (CDSs), each with an optional (but recommended) alternate.

  A base sysplex has only a single CDS (with an optional alternate for availability) called the sysplex couple data set. The sysplex CDS defines the sysplex and contains (for example) information related to the members of the sysplex and members of specific groups within the sysplex (called XCF groups). An IMS data sharing group is one example of an XCF group:

  – Sysplex CDS

    Contains information about the (parallel) sysplex, such as the OS/390 or z/OS members of the sysplex and the members of XCF groups. An IMS data sharing group is one example of an XCF group. There are many in an IMSplex. This is the only CDS required in a base sysplex.

  – CFRM CDS

    Contains the Coupling Facility resource management (CFRM) policy which defines the Coupling Facilities and the structures that may be allocated in the CFs by connectors.

  – LOGR CDS

    Contains the System Logger (LOGR) policy used by the System Logger to build log streams for its clients. The Common Queue Server (CQS) is a client of the System Logger and writes to two log streams - one for full function shared message queue structures, and one for Fast Path shared EMH queue structures.

  – SFM CDS

    Contains the sysplex failure management (SFM) policy which determines how the system is to manage certain types of failures, such as connectivity to a structure, or loss of communications between sysplex members.

  – ARM CDS

    Contains the Automatic Restart Manager (ARM) policy which defines the strategy for restarting jobs or tasks which have failed.

  – WLM CDS

Contains the Workload Manager (WLM) policy used by the WLM to manage system resources to meet performance objectives.

- Cross-system coupling facility (XCF)

  In 1990, MVS™ was enhanced to include a set of services provided by system software called cross-system coupling facility (XCF) services. XCF has since been a basic part of all supported MVS, OS/390 and z/OS systems ever since the base sysplex was made available and provides sysplex services to OS/390 and to subsystems such as IMS. Examples of such services include:

  – Group services
  – Signalling services
  – Monitoring services
  – Sysplex services for recovery

  These will be described in more detail later, but they all basically deal with communications between OS/390 components and/or subsystem with the sysplex.

- Cross-system extended services (XES)

  With the Parallel Sysplex came another set of OS/390 services known as cross-system extended services. XES provides OS/390 components and subsystems with sysplex services for data sharing — a comprehensive set of services to access shared data in the Coupling Facility structures. Examples, which will be discussed in a little more detail later, include:

  – Connection services
  – Lock services
  – Cache services
  – List services

- CF structures

  Within the Coupling Facility are blocks of storage referred to as structures. These structures contain all of the user-related data that is to be shared across the sysplex. These will be discussed in more detail later, but there are three types of structures available to the user, each offering a different set of functions:

  – Cache structures
  – Lock structures
  – List structures

  A full-blown IMSplex uses all three types of structures to store information about shared data, shared message queues, and IMSplex resources. Defining, sizing, and managing these structures is a critical part of implementing and managing the IMSplex.

- Hardware system area (HSA)

  Some sysplex services allocate and use storage within each CPC to signal the occurrence of certain events within the Coupling Facility. This storage is allocated as bit vectors in the hardware system area (HSA) by connectors to cache and list structures, with each bit signaling the occurrence of an event associated with that structure. There are two types of bit vectors:

  – *Local cache vectors* signal that a database buffer has been invalidated.
  – *List notification vectors* signal that a message has arrived on a list structure.

  IMS uses these bit vectors for data sharing and for shared queues.

- Policies

  Finally, there are the sysplex *policies* mentioned above. These policies are in the various Parallel Sysplex CDSs and describe how the sysplex is to manage certain resources, conditions, or events. They are typically created by the sysplex administrator or systems programmer and are tailored to the needs of the installation. For example, an Automatic Restart Manager (ARM) policy in the ARM CDS describes the actions ARM (XCF's sysplex service for recovery) should take when a registered subsystem fails, or if the OS/390 system on which that subsystem is running fails. The Coupling Facility resource management (CFRM) policy in the CFRM CDS defines the structures which may be allocated in the Coupling Facilities. There are other policies for Workload Manager, System Logger, and System Failure Management, each in their own CDS.

2.2.2 OS/390 component and subsystem software

To take advantage of the XCF and XES services available in a Parallel Sysplex, OS/390 components and subsystems such as IMS must explicitly invoke these services. The number of sysplex exploiters has grown dramatically since it was first announced, and now includes, among many others:

- IMS, IRLM, CQS, IMS Connect, WebSphere MQ, DB2, CICS T/S
- XCF (uses XES services), JES2, RACF®, VSAM, VTAM, System Logger

There are surely many more and each new product provided by IBM, as well as other software vendors, may take advantage of these sysplex services. How to use them is documented in a series of IBM publications. Examples include:

- *z/OS MVS Programming: Sysplex Services Guide*, SA22-7617
- *z/OS MVS Programming: Sysplex Services Reference*, SA22-7618
- *z/OS MVS Setting Up a Sysplex*, SA22-7625

Figure 2-1 shows some of the major components of a possible Parallel Sysplex configuration.

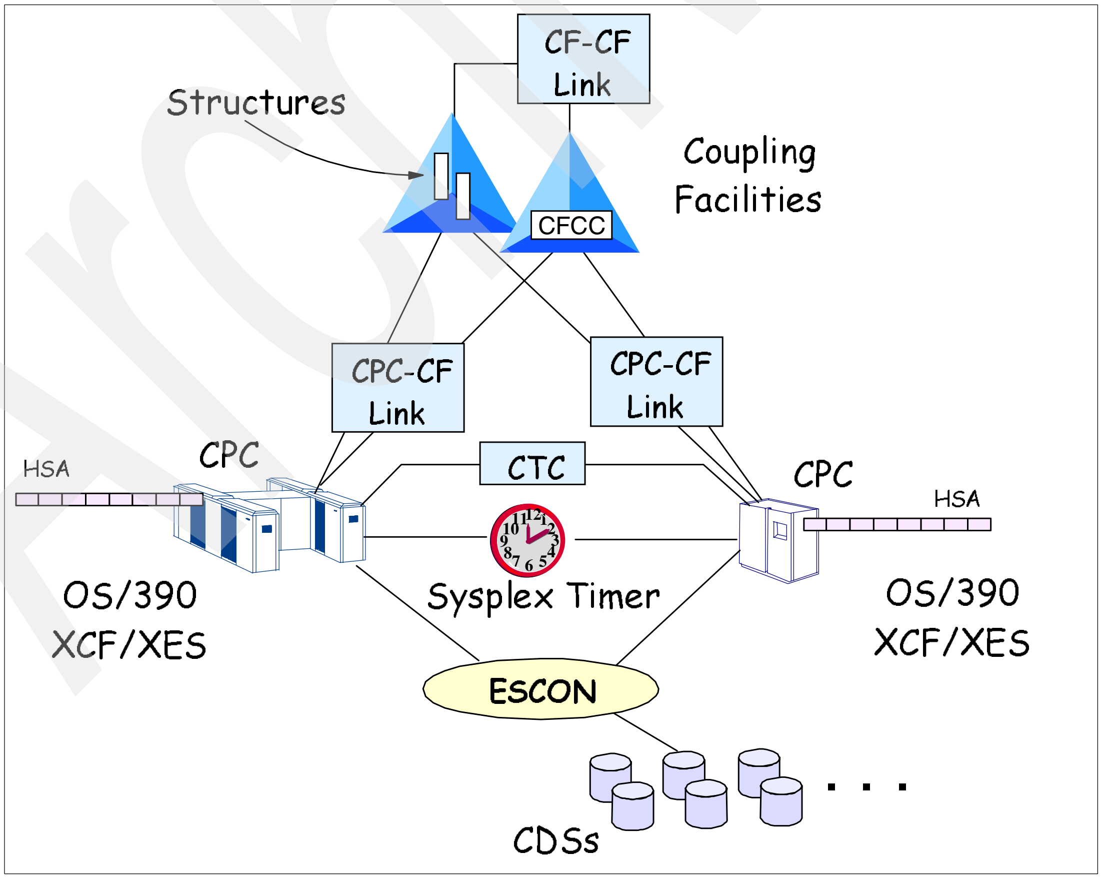


Figure 2-1 Parallel Sysplex configuration

2.3 Sysplex services

This section provides an overview of the sysplex services provided by XCF and XES, and identifies where IMS invokes these services. As always, a more complete and detailed explanation can be found in the IBM sysplex documentation, but many of the terms and concepts we discuss here, you will hear in discussions about the Parallel Sysplex with IMS. The following sections in this chapter should give you a good understanding of how IMS exploits these services, and help you in the planning, implementation, and operational phases of an IMS migration to the Parallel Sysplex environment.

We briefly discuss three types of services:

- Sysplex services for communications
- Sysplex services for recovery
- Sysplex services for data sharing

2.4 Sysplex services for communications

Cross-system coupling facility (XCF) offers a set of communication services which allows multiple programs or subsystems, running on different systems in a sysplex, to share status information and to communicate with each other. XCF communications services are available only between members of an XCF group. Before explaining what each of these services offers, it is first necessary to describe the concept of an XCF group.

2.4.1 XCF groups

According to *z/OS V1R3.0 MVS Sysplex Services Guide,* SA22-7617:

> "*A group is a set of related members defined to XCF by a multisystem application in which members of the group can communicate (send and receive data) between MVS systems with other members of the same group.*"

There are many XCF groups in a Parallel Sysplex, and IMS and IMSplex-related programs are members of many XCF groups. Groups are dynamically created when the first member joins the group. For example, when the IRLMs are initiated, they join an XCF group using the name specified in the IRLMGRP= parameter. The group is dynamically created by XCF when the first member joins. Information about the group is maintained in the sysplex couple data set. We refer to this as a data sharing group, but it is also an XCF group. Being a member of a group entitles that member to use XCF signaling and monitoring services.

There are many other groups in an IMSplex. Examples are:

- Data sharing group (IRLMs)
- Shared queues group (CQSs)
- OTMA group (IMSs, IMS Connect, WebSphere MQ)
- Fast Database Recovery group (1 IMS and 1 FDBR)

In addition to these explicitly joined groups, each *connector to a CF structure* becomes a member of a group associated just with that structure. For example, each IRLM is not only a member of a data sharing group, but is also a member of a group associated with the IRLM lock structure. Being a member of the lock structure group allows each member to be notified when there is a change in structure status, such as a structure failure. There is another group that IMSs in an IMSplex may belong to, but it is not an XCF group. That is the VTAM generic resource group. While the various VTAMs in the sysplex belong to a group, the members of a VTAM generic resource group do not.

2.4.2 XCF group services

XCF group services allow authorized programs to join (create, if necessary) and leave XCF groups, and to be informed when other members join or leave the group. By belonging to a group, the member is authorized to invoke other XCF services, such as signaling and monitoring services. When joining a group, the member can identify a *group user routine*. When a new member joins the group, or an existing member leaves the group, this routine in each remaining member is driven by XCF to notify that member of the change in group status. For example, IMS and Fast Database Recovery (FDBR) join an XCF group using the name specified in the GROUPNAME parameter in the DFSFDRxx PROCLIB member. If IMS fails, XCF drives FDBR's group user routine to inform FDBR that the IMS system it was monitoring failed so that it can back out in-flight database updates and release the locks held by the failed system. If an IRLM fails, other IRLMs are notified through their group user routine that one has failed, and each can then take action to read the retained locks held by the failed IRLM.

2.4.3 XCF signaling services

XCF signaling services provide the means for members of a group to send messages to, receive messages from, and respond to messages received from, other *members of the same XCF group*. XCF macros are provided to communicate with other group members. The XCF macros invoke user message routines to send messages, receive messages, and respond to received messages. It is not necessary for any member to know which OS/390 system any other member is on, in order to send a message. No special sessions are established between the group members. For example, IMS Connect and IMS both join an OTMA (XCF) group. When IMS Connect wants to send a message to IMS OTMA, it uses XCF to send the message without caring where that IMS is located. OTMA's message user routine is then driven to inform IMS that a message has arrived from IMS Connect. IMS sends the response to IMS Connect using the same techniques.

2.4.4 XCF monitoring services

XCF monitoring services allow members to monitor the operational status of other members of the same XCF group. When a member joins a group, it identifies to XCF a *group user routine, a status user routine, a 32-byte member state field, and a status checking interval*. That member must update this status field periodically (and within the specified status checking interval). XCF monitors this status field based on the specified status checking interval and, if not updated within the specified interval, drives the status user routine to determine the status of the member. If the status user routine indicates that the member's status has changed, or if the status user routine just does not respond, then XCF drives the group user routines of the remaining members to inform them of the changed status of the member. Although there are five different states that can exist for a member, we will consider only two - ACTIVE and FAILED.

An example of this is the IRLMs in a data sharing group. Each IRLM updates a status field in CSA indicating that it is still alive and kicking (active). If one IRLM fails, XCF will detect that the IRLM has stopped updating its status field and (try to) drive that IRLMs status user routine. If the IRLM has truly failed, then the status user routine will not respond and XCF will notify the other IRLMs of the failure, at which time they will begin action to read the retained locks from the IRLM structure.

OS/390 also has a status field, but it is on the sysplex CDS instead of in CSA. If OS/390 fails, or the processor fails, then this status field will not be updated. XCFs running on other members of the sysplex monitor each other's status field and will detect the status missing condition and take action to isolate the failed system. XCF will then drive the group user routines of every group member on the surviving OS/390s that had a member on the failed OS/390 system. For example, if the OS/390 that IRLM1 was running on failed, all the other IRLMs would be informed.

2.5 Sysplex services for recovery

When running in a Parallel Sysplex, there are two types of sysplex services that address availability and recovery, and in particular, address problems when a system (for example, OS/390 or the CPC) fails, when an application (for example, IMS) fails, and even when a Coupling Facility or the link to a Coupling Facility fails.

2.5.1 Sysplex failure management

Sysplex failure management (SFM) allows you to define an SFM policy in the SFM couple data set (CDS) in which you can describe the actions you want OS/390 or z/OS to take when certain failures occur within the sysplex. Responsibility for this policy will not lie with the IMS people, but with the systems folks. Therefore, this description will be very brief and is intended to make you aware that such a policy exists and how it might affect your IMSplex.

The failures addressed by the SFM policy include:

- System failures

  Members of a Parallel Sysplex must periodically update a status field in the sysplex couple data set (CDS). The SFM policy can specify an INTERVAL within which a system must update this status field or trigger a *status update missing condition.* When an OS/390 or z/OS system fails, or the LPAR on which it is running fails, that OS/390 or z/OS can no longer update its status in the sysplex CDS and a status update missing condition exists. Other systems detect this and can use this to initiate recovery action. For SFM, recovery actions might be to PROMPT an operator to take action, or automatically ISOLATE the failed system from the sysplex without operator intervention. Isolation terminates any I/O and Coupling Facility access, resets channel paths, and loads a non-restartable wait state on the isolated system. In effect, isolation ensures that the isolated system (which might not really have failed, it may just not be able to update the status or to communicate) cannot corrupt shared resources.

- Signalling connectivity failures

  All systems in the Parallel Sysplex must be able to communicate. When they cannot, then one or more of them must be removed from the sysplex (isolated). When you want SFM to handle this condition, the SFM policy must specify CONNFAIL(YES). The SFM policy can also assign a WEIGHT to each system. When there is a choice about which system to isolate, it will use these weights to maximize the aggregate value of the surviving system in the sysplex.

- Coupling Facility connectivity failures

  Loss of connectivity to a Coupling Facility from a system can occur because of a link failure or because of a failure to the CF itself. When this happens, that system can no longer access the structures on the CF. For example, if the CF contained the IRLM lock structure, any IRLM on that system could no longer make lock requests and data sharing would stop.

  The CFRM policy contains definitions for each structure. That structure definition may have a REBUILDPERCENT parameter which says that when connectivity to this structures falls below this percent, to automatically rebuild that structure on another Coupling Facility which has the required connectivity. Since the loss of any IMS structure is serious business, users generally set REBUILDPERCENT for IMS structures to a very low value, for example:

```
REBUILDPERCENT(1) <<< this is also the default
```

which would cause the structure to be rebuilt whenever there is any connectivity failure. When there is no SFM policy active, then rebuild is always done for those structures that support it (all IMS structures support rebuild).

2.5.2 Automatic Restart Manager (ARM)

Automatic Restart Manager (ARM) provides a complementary service to sysplex failure management. SFM deals primarily with systems and communications, while ARM addresses specific work running on those systems. If a program (such as IMS) fails, or the system on which that program is running fails, ARM can restart that program either on the same system (if it is a program failure) or on another system (if it is a system failure).

To take advantage of ARM, the program must register with ARM as a restartable *element*. Most of the system-type address spaces in an IMSplex register with ARM as a default, but can be overridden by an execution parameter (for example, ARMRST=Y|N). IMS dependent regions are not supported by ARM - they must be restarted by operator command or automation. Specifically, the following address spaces which may be a part of, or connected to, your IMSplex may register with ARM:

- IMS control regions (DB/DC, DBCTL, DCCTL)
- Fast Database Recovery region (FDBR)
- Internal Resource Lock Manager (IRLM)
- Common Queue Server (CQS)
- CSL address spaces (SCI, RM, and OM)
- IMS Connect
- DB2
- CICS
- VTAM

It is the responsibility of the application (for example, IMS) to enable itself to be restarted by ARM. This is done by registering as a restart *element* to the automatic restart management service of the Parallel Sysplex. ARM is activated in the sysplex by defining an *ARM policy* in the ARM CDS and then activating that policy.

```
SETXCF START,POLICY,TYPE=ARM,POLNAME=yourpolicyname
```

The following are highlights of the automatic restart management services. Most can be selected as options or overridden in the ARM policy.

- Abends

  The registered element will be restarted on the same OS/390 image. When registering, IMS identifies some abend types as not restartable. In general, these are abend types for which you would not want an automatic restart, for example:

  – U0020 - IMS is modified down
  – U0028 - /CHE ABDUMP
  – U0624 - /SWITCH SYSTEM (XRF planned takeover)
  – U0758 - message queues are full
  – U0759 - message queue error

  Following an abend, IMS will be restarted with AUTO=Y causing /ERE.

- System failures

The registered element will be restarted on another candidate system in the sysplex. Candidate systems must be identified in the ARM policy. This is called a cross-system restart. Following a system failure, IMS will be restarted with AUTO=Y and with /ERE OVERRIDE or /ERE BACKUP (for XRF alternate).

- Restart groups

  Elements that must execute together on the same image can be grouped for cross-system restart; for example, IMS and DB2, or IMS and CICS AORs.

- Restart order

  These parameters apply to groups and can specify the order in which elements are to be restarted. For example, you will probably want to restart IMS before restarting a CICS AOR which is going to connect to IMS.

- Restart method

  You can specify how you want the element to be restarted. There are three choices:

PERSIST	Restart the same way it was started
JOB	Specify a library containing the restart JCL
STC	Specify a started task

  For IMS, PERSIST is generally the right choice, unless there is some reason you want to use a different restart procedure when restarting, for example, on a different OS/390 image.

- Restart attempts

  You can specify in the policy how many times you want restart to be attempted within a specified time frame. This will avoid continual restart/failure scenarios. Setting this value to zero (0) will bypass restart even if IMS is registered for restart. Note that this is the only way you can bypass ARM restart for the IRLM and DB2. There is no execution parameter for them to enable or disable ARM.

The above is a sampling of the capabilities and options you have with ARM. There are several other parameters you can specify in the ARM policy which have not been identified here. Refer to OS/390 documentation for a complete description of ARM and the ARM policy and parameters (see *z/OS MVS Programming: Sysplex Services Guide,* SA22-7617).

2.6 Sysplex services for data sharing

Cross-system extended services (XES) is a set of sysplex services for data sharing which allows subsystems, system products, and authorized programs to use Coupling Facility structures for high performance, high availability data sharing. In this context, data sharing means the sharing of *any type of data* by programs running on any system in the sysplex. It does not mean data sharing only in the IMS (or DB2 or VSAM) sense of the term. IMS data sharing utilizes these sysplex services for data sharing to manage its own block level data sharing of IMS databases. Other examples of shared data used within the IMSplex include shared messages (IMS shared queues), shared log data (CQS and the System Logger), and shared information about IMSplex resources (IMS Resource Manager).

Shared data resides in one of three structure types in the Coupling Facility — cache structures, lock structures, and list structures. XES provides a set of services to connect to and define the attributes of these structures (connection services), and another set of services unique to each structure type (cache services, lock services, and list services). Connectors use these services to invoke the functions supported by that structure type.

The details of each structure type and the XES services supporting them is documented in the z/OS library of manuals — one good reference is *z/OS MVS Programming: Sysplex Services Guide,* SA22-7617. The following is a very brief overview of the structure types, their components, what they are used for (by IMS), and what services are provided by XES.

2.6.1 Structures, connectors, and services

A *structure* is a Coupling Facility construct whose purpose is to support the sharing of data (for example, data in an IMS database, or messages in an IMS message queue) among multiple *connectors* to the structure through a set of *XES services* rather than require individual connectors to manage that storage directly. This not only simplifies and standardizes the access to the CF storage, but allows for enhancements to CF storage management functions without having to modify individual programs using that storage. All structures in a Parallel Sysplex must be defined in a Coupling Facility resource management (CFRM) *policy* in the CFRM couple data set.

There are three types of structures that can be allocated in a Coupling Facility - *cache structures, lock structures, and list structures*. Each type is designed to support certain data sharing functions and has its own set of XES services. Programs such as IMS, which want to access a structure and invoke these XES services for data sharing, connect to it using XES connection services. After connecting to the structure, the program then accesses data in the structure using XES cache, lock, or list services.

2.6.2 Connectors and connection services

XES provides connection services for a program to connect to (and disconnect from) a structure, and to define the type and attributes of the structure. That program is called a *connector* and, when connected to a structure, is authorized to use other XES services appropriate to the type of structure to which it is connected. The first connector to a structure determines the structure attributes. Other users can connect but cannot change those attributes. Some of the attributes of a structure are:

- Structure information, such as type and structure persistence
  - *Structure types* are cache, lock, and list structures. Note that while structures must be pre-defined in the CFRM policy, the type of structure is not determined until the first user connects to the structure.
  - A *persistent structure* is one which, when all connectors have disconnected from the structure, the structure remains allocated in the Coupling Facility. It is still there the next time a program connects to it. Examples of persistent IMS structures are the shared queues structures, the IRLM lock structure, and the resource structure. OSAM, VSAM, and shared VSO structures are not persistent.
- Connection information, such as connection persistence
  - A *persistent connection* is one which persists if the connector fails. A non-persistent structure will not be deleted if there are any active or failed persistent connections to that structure. An IMS example is a shared VSO structure. It has connection persistence but not structure persistence. If all IMSs terminate their connection to a shared VSO structure normally, then it is deleted. However, if one IMS fails, then its connection becomes a failed persistent connection and the structure will not be deleted even if there are no other active connectors.
- Structure modification information determines how a structure may be modified after it is allocated. For example, does the connector support structure rebuild, structure alter, structure duplexing, and system managed processes.

Also during connection to a cache or list structure, each connector provides information about the *local cache vector* (required for cache structures) or *list notification vector* (optional for list structures). These vectors, located within HSA, define a relationship between each connector and the structure and are used to signal the occurrence of some structure event, such as the invalidation of a data buffer, or the arrival of a message on a list structure. For more information about structure attributes, refer to *z/OS MVS Programming: Sysplex Services Reference,* SA22-7618.

2.6.3 Cache structures and cache services

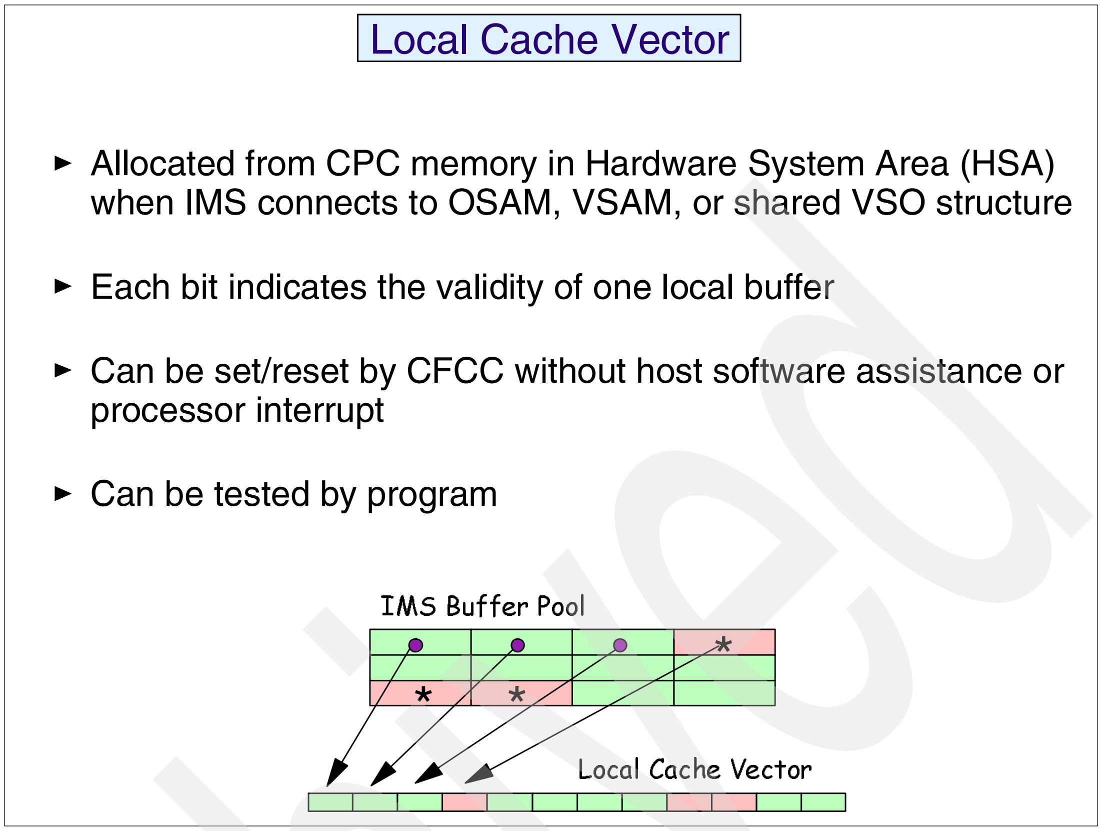
A cache structure and its related cache services provide connectors with data consistency and (optionally) high speed access to data. Data consistency ensures that if one connector updates a block of data, any other connector with that same block of data in its buffer pool will be notified that its copy is invalid and should be refreshed if it is re-used. This is called *buffer invalidation*. If the connector opts to store data in the structure, then any connector can read that data from the cache structure (at CF speeds) instead of reading it from DASD (at the much slower DASD speeds). Whether or not a cache structure contains data entries is determined by the first user to connect to the structure — that is, the one that sets the structure type and attributes. There are three types of cache structures that may be allocated in the Coupling Facility. These are:

- **Directory-only** cache structures do not contain any data. They have only directory entries which are used to register interest by a connector in a particular block of data.
- **Store-through** cache structures contain both directory entries and data entries.
- **Store-in** cache structures contain both directory entries and data entries.

The difference between store-in and store-through structures is explained later.

Local cache vector

For cache structures, each connector identifies the length of the local cache vector for that structure in HSA. Each bit in the local cache vector indicates the validity of the contents of one local buffer in one connector's buffer pool(s). IMS uses one cache structure for VSAM and another cache structure for OSAM, each with its own local cache vector. For shared DEDB VSO AREAs, there is one cache structure for each AREA, each with its own local cache vector. One bit in the local cache vectors is associated with each VSAM, OSAM, or shared VSO buffer. Figure 2-2 is an illustration of an IMS buffer pool and the association of each buffer with a bit in the local cache vector. Note that all cache structures have local cache vectors with relationships to local buffers whether there are data entries in the structure or not.



Local Cache Vector

- ► Allocated from CPC memory in Hardware System Area (HSA) when IMS connects to OSAM, VSAM, or shared VSO structure
- ► Each bit indicates the validity of one local buffer
- ► Can be set/reset by CFCC without host software assistance or processor interrupt
- ► Can be tested by program

Figure 2-2 IMS local cache vector

Directory-only cache structures

Directory-only cache structures do not have data entries — only directory entries. *Each directory entry identifies a block of data* which is in a connector's buffer pool. For example, a block of IMS data is identified by its global DMB number (determined by DBRC), its data set ID (which data set within the database), and the relative byte address (RBA) within the data set. This is sufficient to uniquely identify that OSAM block or VSAM control interval (CI).

The directory also identifies which connectors (IMSs) have that block in their local buffer pool, and a *vector index* relating the directory entry to a buffer in the local buffer pool. When one connector updates a block of data on DASD, the CFCC uses the vector index in the directory entry to invalidate that buffer in other connectors having the same block in their buffer pools. Note that all cache structures have directory entries, and all directory entries perform the same functions.

Figure 2-3 is an illustration of a directory only IMS cache structure (for example, the VSAM cache structure). In this example, BLK-A is in IMSX's buffer pool in a buffer represented by vector-index-1 in IMSX's local cache vector. BLK-A is also in IMSY's buffer pool in a buffer represented by vector-index-4 in IMSY's local cache vector. The bits in the local cache vector were assigned to the buffers by each IMS when it connected to the structure.

IMS uses this type of structure for all shared VSAM databases, and has the option to use it for OSAM.

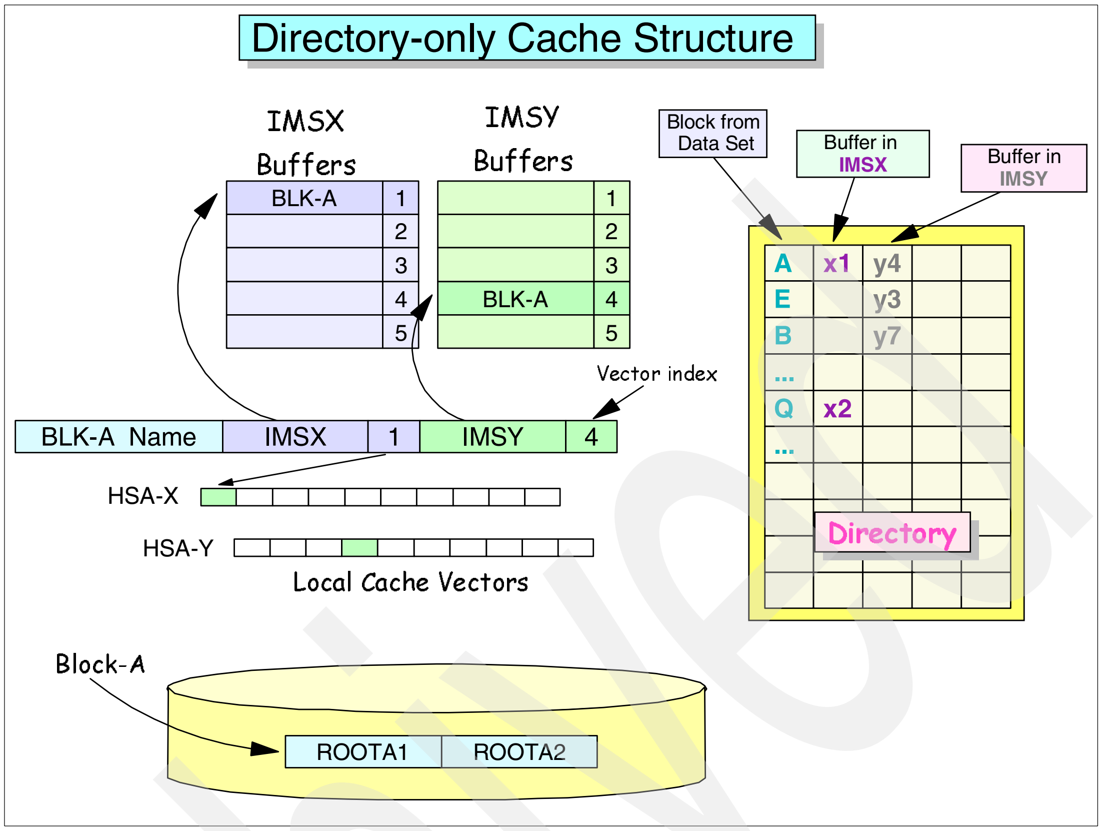


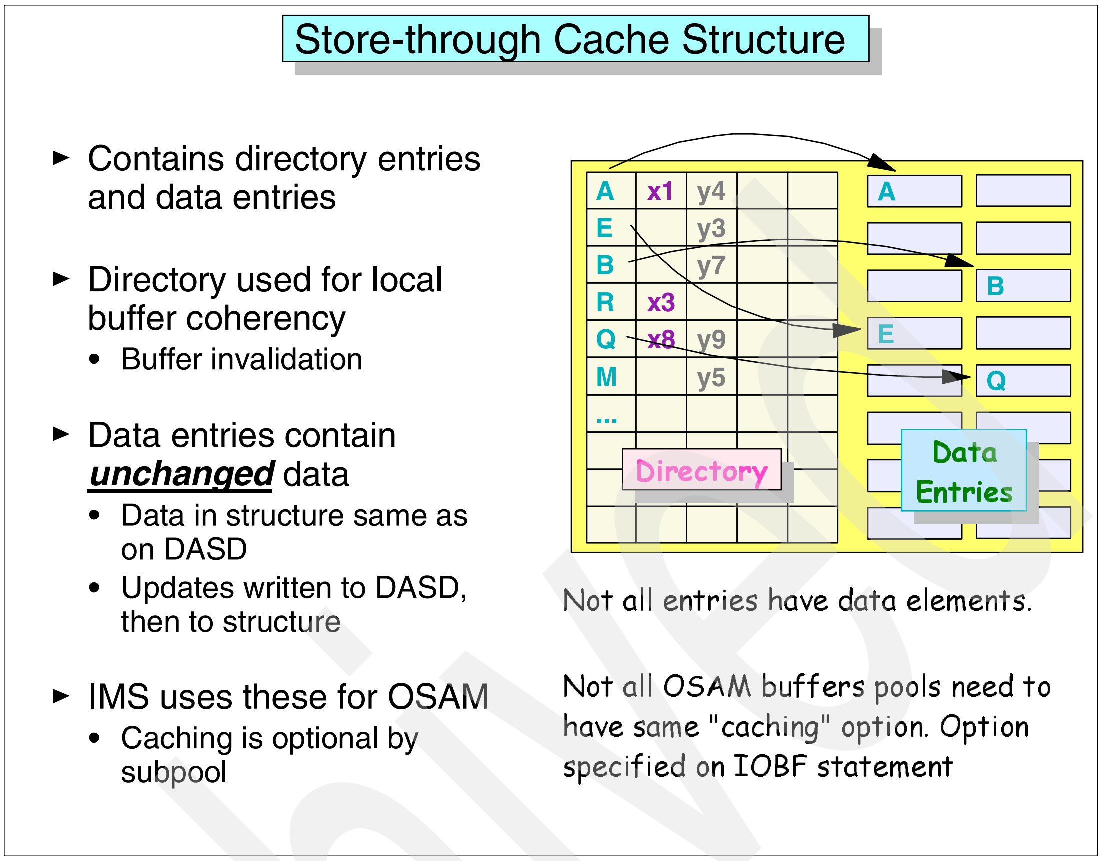
Figure 2-3 Directory-only cache structure

Store-through cache structures

A store-through cache structure contains both directory entries and data entries. The directory entry for the block indicates whether there is data for that block in the structure (not all directory entries have data in the structure) and where that data is in the structure. The data in the structure is considered *unchanged* data. That is, any updates to that block of data *already exist on DASD* before being written to the structure. IMS may use this type of structure for OSAM. The other option for OSAM is a directory-only cache structure. As a user of shared OSAM databases, you must choose between directory-only or store-through cache structures.

Figure 2-4 illustrates a store-through cache structure. Note that not all directory entries have data entries. This may happen when IMS reads a block of data into a buffer but does not write it to the structure. OSAM has an option, for example, to not write data to the OSAM cache structure unless it has been changed. If it is changed, OSAM will write it first to DASD and then to the structure.

Although there is only one OSAM cache structure, whether or not to cache data in that structure is an option set at the subpool level. When this type of structure is used, but not all OSAM data is cached, for blocks that are not cached there will be directory entries for the blocks in the buffers but no data entries.



Store-through Cache Structure

- Contains directory entries and data entries
- Directory used for local buffer coherency
  - Buffer invalidation
- Data entries contain ***unchanged*** data
  - Data in structure same as on DASD
  - Updates written to DASD, then to structure
- IMS uses these for OSAM
  - Caching is optional by subpool

Not all entries have data elements.

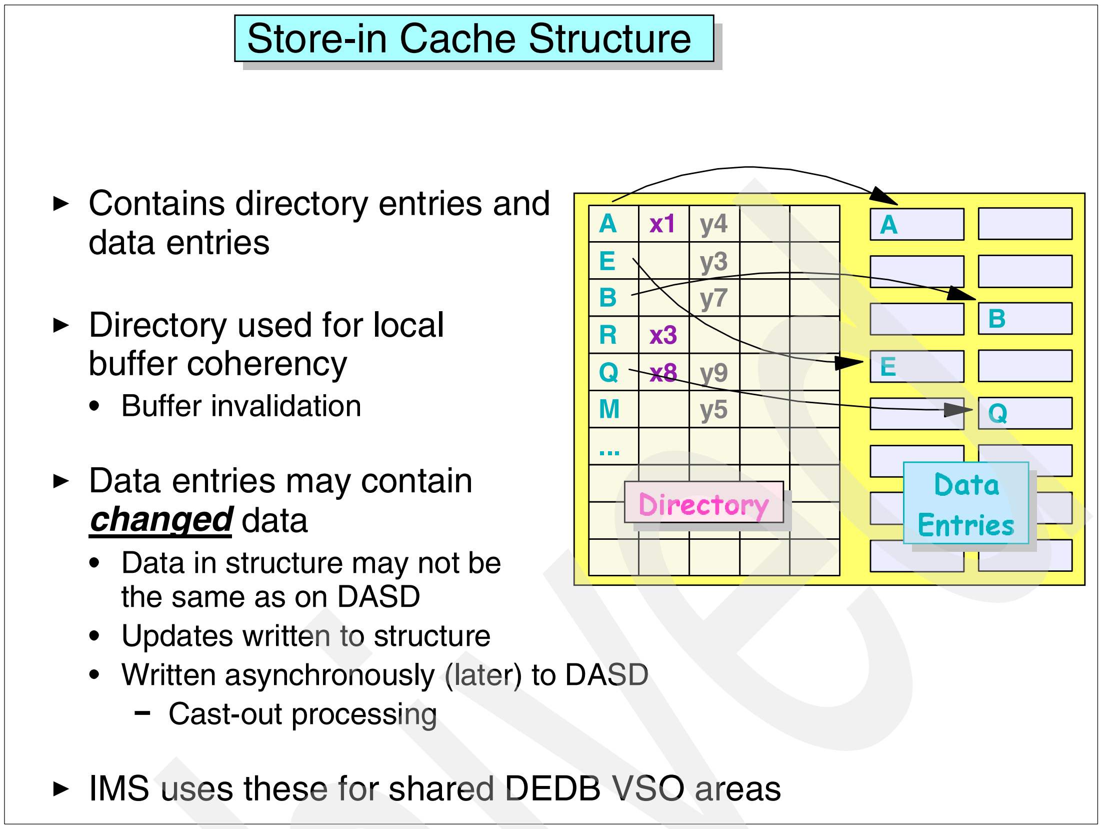
Not all OSAM buffers pools need to have same "caching" option. Option specified on IOBF statement

Figure 2-4 Store-through cache structure

Store-in cache structures

A store-in cache structure may contain *changed* data. Changed data is data which has been written to the structure but not yet written to DASD. When a connector writes a block of data to the structure, it indicates whether the data has been changed since it was last written to the structure. A flag in the directory entry indicates that the data entry contains changed data. Periodically, the connector requests a list of all changed data entries, reads them from the structure, and writes them out to DASD, turning off the changed flag in the directory entry. This process is called *cast-out processing*. IMS uses store-in cache structures for shared VSO AREAs. It invokes cast-out processing at every system checkpoint, and whenever the VSO AREA is closed.

IMS uses store-in cache structures only for shared VSO AREAs in a data entry database (DEDB). Store-in cache structures can offer a very significant performance benefit over store-through structures because there is no I/O. If a shared VSO CI is updated many times between two IMS system checkpoints, then with a store-through structure (such as, OSAM), that block or CI is written to DASD each time it is committed — perhaps hundreds or thousands of times between system checkpoints. If it is in a store-in cache structure (such as, shared VSO), then it gets written to the structure each time it is committed, but only gets written to DASD once per system checkpoint, saving perhaps hundreds or thousands of DASD I/Os.


A
x1
y4
E
y3
B
y7
R
x3
Q
x8
y9
M
y5
...
Directory
A
B
E
Q
Data Entries

Store-in Cache Structure

- Contains directory entries and data entries
- Directory used for local buffer coherency
  - Buffer invalidation
- Data entries may contain ***changed*** data
  - Data in structure may not be the same as on DASD
  - Updates written to structure
  - Written asynchronously (later) to DASD
    - Cast-out processing
- IMS uses these for shared DEDB VSO areas

Figure 2-5 Store-in cache structure

Comparing store-through and store-in cache structures

Each type of cache structure has its advantages

- Directory only cache structures contain no data entries. They can be smaller and they do not have the overhead of writing data both to DASD and to the structure.
- Store-through cache structures contain data entries that represent unchanged data. They are larger than directory-only cache structures and there is the overhead of writing the updated data to the structure each time it is updated on DASD. Other connectors may benefit from the data written earlier by the same or another connector. If the structure fails, since the data is unchanged, there is no database recovery required.
- Store-in cache structures have the same size and overhead considerations as store-through structures. They have similar read advantages while allowing the connector to avoid much of the DASD I/O. However, if a store-in cache structure fails, then database recovery will be required (changed data was in the failed structure that is not on DASD). Because of this, Fast Path has the option of using dual structures for shared VSO. Beginning with z/OS 1.2 and CF Level 11, structure duplexing is supported which may be an alternative to Fast Path maintaining dual structures.

XES cache services

Connectors to cache structures invoke connection services to connect and define the type of structure and its attributes, but then use cache services to make use of the structure and its functions. Connectors request cache services using the IXLCACHE macro and specifying a specific function on the request. We talk briefly about four cache service request types.

READ_DATA request

Before IMS reads a block of data into its buffer pool, it must first tell the Coupling Facility which block of data it will read and which buffer it will put that data in. This is called *registering interest* in a data item and is necessary so that, if another IMS updates that block, the other IMS's buffer can be invalidated. This XES cache service is the READ_DATA request. It causes a directory entry to be created (or updated, if it already exists). If an existing directory entry points to a data entry in the structure when the READ_DATA request is made, that block will be automatically read from the CF into the buffer identified on the request without further action by the connector. For directory-only cache structures, of course, there is never a data entry in the structure, but the READ_DATA request is still issued to register interest.

WRITE_DATA request

IMS issues a WRITE_DATA request to write a block of data to a store-through or store-in cache structure. If IMS has updated that block of data, then the request will contain an indicator that the data has been changed. If any other IMS has registered interest in the data item, its buffer will be invalidated by flipping a bit in the local cache vector. Which bit to flip is determined by looking at the directory entry. A signal is then sent to the processor to flip that bit in the local cache vector, indicating the buffer is invalid. The IMS which had its buffer invalidated is removed from the directory entry. Note that this IMS does not know its buffer has been invalidated until it tests the bit setting in the local cache vector on a refetch (explained below).

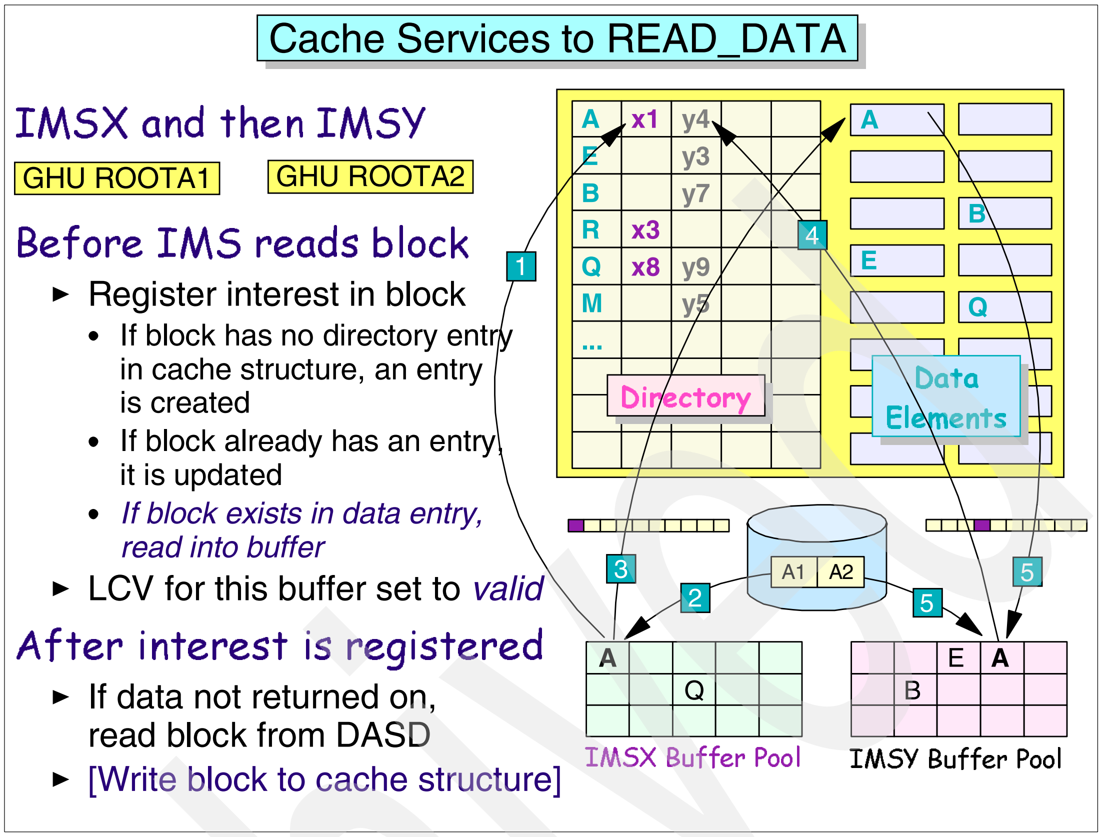
CROSS_INVALidate request

When IMS updates a block of data on DASD and does not write it to the structure (for example, if it is a directory-only structure), then of course there is no WRITE_DATA request to invalidate the buffers. In this case, IMS will issue a CROSS_INVALidate request to tell the CFCC to invalidate any other IMS's buffer with registered interest in the data item.

TESTLOCALCACHE request

Before any IMS uses data in any buffer, it must first test to determine if the contents of that buffer are still valid. To do this, it must test the bit in the local cache vector to see if some other IMS has cross-invalidated it. This is a TESTLOCALCACHE request and must be issued before each access to the buffer. If a buffer is found to be invalid, then that IMS must re-register interest in the data item and, if not returned on the READ_DATA request, then reread it from DASD.

Figure 2-6 and Figure 2-7 are examples of IMS using cache services to read and update data.



Cache Services to READ_DATA

IMSX and then IMSY

GHU ROOTA1 GHU ROOTA2

Before IMS reads block

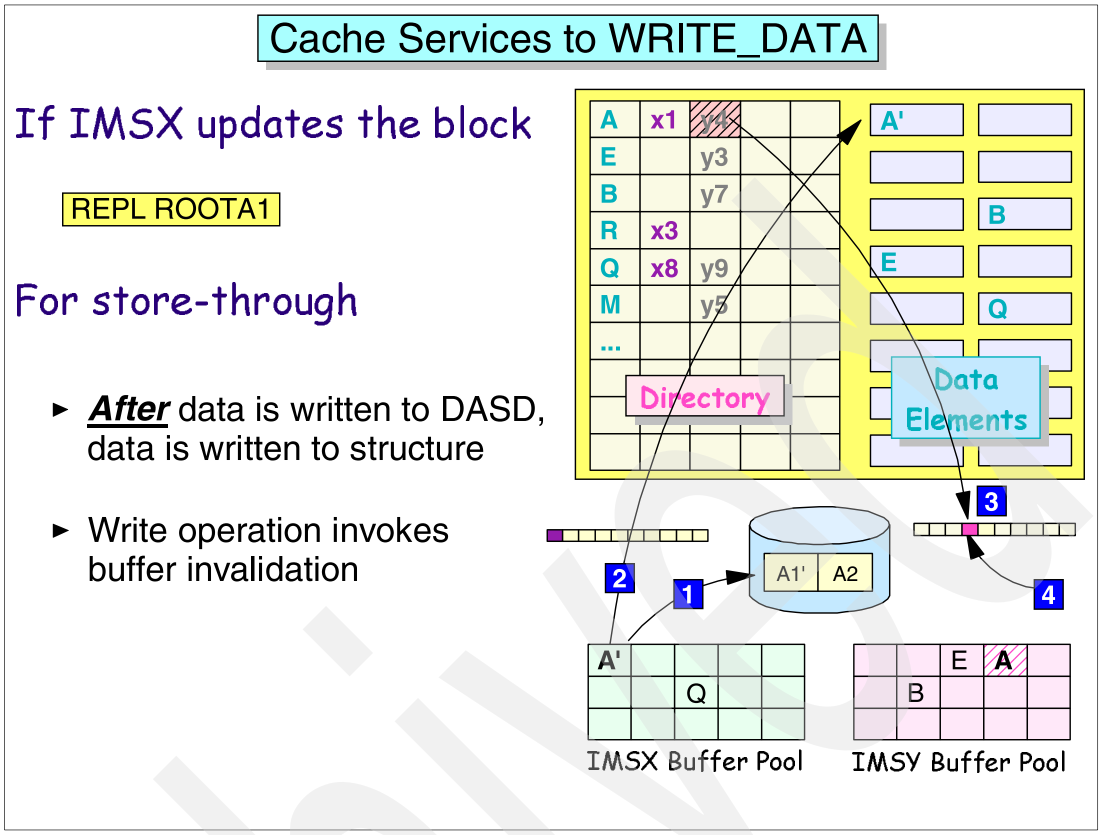
- Register interest in block
  - If block has no directory entry in cache structure, an entry is created
  - If block already has an entry, it is updated
  - *If block exists in data entry, read into buffer*
- LCV for this buffer set to *valid*

After interest is registered

- If data not returned on, read block from DASD
- [Write block to cache structure]

Figure 2-6 Example of a READ_DATA cache service request

1. A program in IMSX issues a get call to ROOTA1 which is in BLK-A. IMSX issues a READ_DATA request to register interest in BLK-A and provides the vector index into the local cache vector. A directory entry for BLK-A is created in the cache structure.
2. IMSX reads BLK-A from DASD.
3. IMSX writes BLK-A to DASD using a WRITE_DATA request and indicating that the data is *not changed*.
4. A program in IMSY issues a call to ROOTA2, which is also in BLK-A. IMSY must register interest in BLK-A using a READ_DATA request. This will update the directory entry for BLK-A to indicate IMSY's interest and its vector index.
5. If BLK-A is in the structure when IMSY issues the READ_DATA request, it will be moved to IMSY's buffer. If it is not in the structure, then IMSY would read it from DASD.



Cache Services to WRITE_DATA

If IMSX updates the block

REPL ROOTA1

For store-through

- ***After*** data is written to DASD, data is written to structure
- Write operation invokes buffer invalidation

Figure 2-7 Example of a WRITE_DATA request to a store-through structure

1. If the program on IMSX updates ROOTA1, IMSX will first write it to DASD.
2. IMSX will then issue a WRITE_DATA request with the changed data indicator on.
3. The CFCC will send a signal to IMSY's processor to flip the bit pointed to by IMSY's vector index in the directory entry and remove IMSY's interest from the directory entry.
4. If IMSY needs to use the buffer again, it must test the validity of the buffer by issuing the TESTLOCALCACHE request. When it finds that it is invalid, it will re-register interest in BLK-A. At this time, since BLK-A (with the updated data) is in the structure, it will be read into IMSY's buffer.

If this were a store-in cache structure (shared VSO), the only difference would be that step-1 would be skipped. Updated blocks in the structure would be written to DASD later (at IMS system checkpoint time) using cast-out processing (another function of XES cache services).

Cache structure sizing considerations

It is important that the cache structure have at least enough space so that every buffer in every connected IMS can have a directory entry. If there are not enough directory entries to support all the buffers, then directory entries will be *reclaimed* when a new data item is registered and the associated buffer(s) will be invalidated.

When the cache structure also contains data, then enough space must be allocated to hold as much data as the user wants. Space for data entries is used just like space in a buffer pool - the least recently used data entry will be purged to make room for a new one. This is unavoidable, unless the database being cached is very small and will fit entirely within the cache structure, so like a buffer pool, the sizing of the structure will affect its performance.

CFSIZER, a tool for helping users size IMS structures, can be found on the Internet at:

http://www.ibm.com/servers/eserver/zseries/cfsizer

2.6.4 Lock structures and lock services

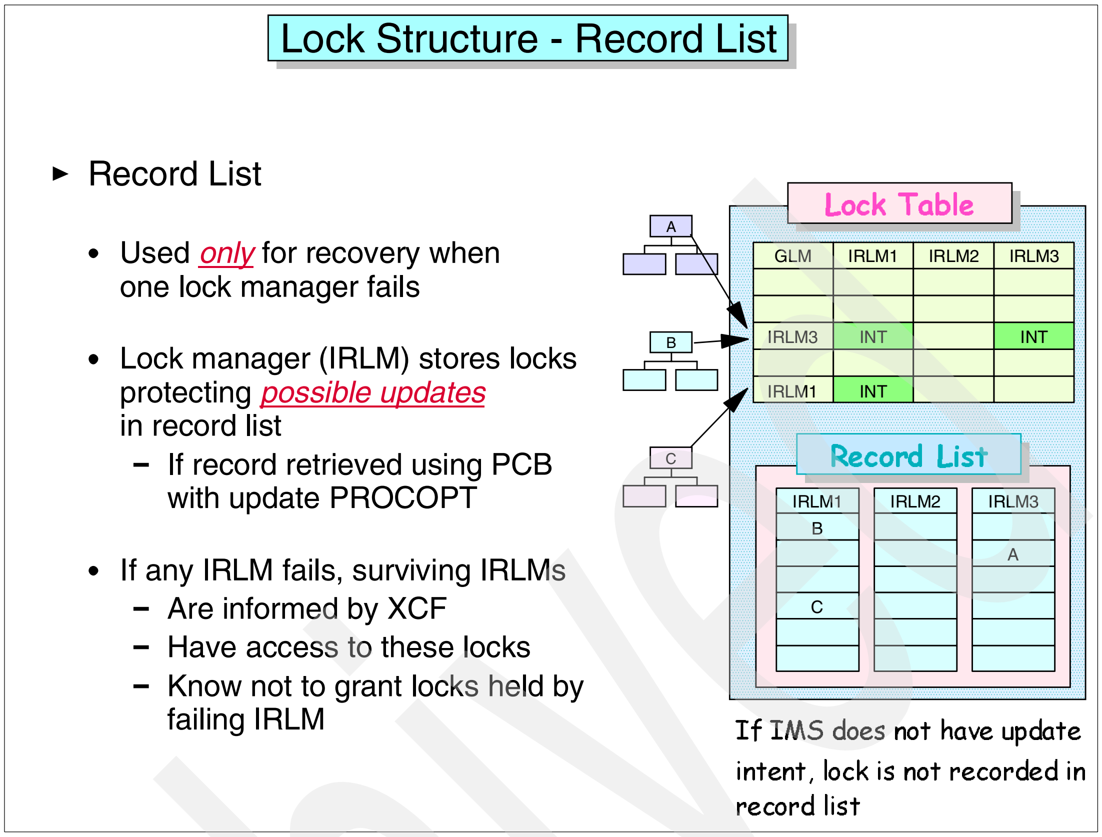
A lock structure can consist of two parts. The first part is a *lock table* — a series of lock entries which the system associates with resources. The second (optional) part is a set of *record data entries* containing a connected user's ownership interest in a particular resource and can be used for recovery if the connector fails.

IMS uses the IRLM to manage locks in a block level data sharing environment. The IRLM uses XES lock services to manage the locks globally. It is the IRLM, not IMS, that connects to the lock structure. In the context of the definition of the components of a lock structure, the connected user is an IRLM and the resource is an IMS database resource (for example, a record or a block).

The lock table consists of an array of lock table entries. The number and size of the lock table entries is determined by the first connector at connection time. Each lock table entry identifies connectors (IRLMs) with one or more clients (IMSs) that have an interest in a resource which has "hashed" to that entry in the lock table. The entry also identifies a *contention manager*, or *global lock manager*, for resources hashing to that entry. The purpose of the lock table is to provide an efficient way to tell whether or not there is a *potential contention* for the same resource among the data sharing clients. If there is no potential contention, a lock request can be granted immediately without the need to query other lock managers for each individual lock request. If the lock table indicates that there is potential contention, then additional work is needed to determine whether there is a conflict between the two requestors. This is the responsibility of the global lock manager.

In the example shown in Figure 2-8, a lock request issued by an IMS system (let's say IMS3) to IRLM3 for RECORDA hashes to the third entry in the lock table (it has a hash value of 3). Interest is registered in that entry and IRLM3 is set as the *global lock manager* or *contention manager* for that entry. There has also been a lock request to IRLM1 for RECORDB. This is *potential lock contention* which must be resolved by the global lock manager. It is impossible to tell from the information in the lock table whether the two requests are compatible or not. It is the responsibility of the global lock manager to resolve this. Also in our example, an IRLM1 client has also requested a lock for RECORDC which hashed to entry 5 in the lock table. Since there is no other interest from any IRLM client for this entry, the lock can be granted immediately.

We will discuss how potential lock conflicts are resolved when we discuss XES lock services.



Lock Structure - Record List

- Record List
  - Used *only* for recovery when one lock manager fails
  - Lock manager (IRLM) stores locks protecting *possible updates* in record list
    - If record retrieved using PCB with update PROCOPT
  - If any IRLM fails, surviving IRLMs
    - Are informed by XCF
    - Have access to these locks
    - Know not to grant locks held by failing IRLM

If IMS does not have update intent, lock is not recorded in record list

Figure 2-8 Record list component of lock structure

Figure 2-8 shows the record list component, or the list of record data entries showing the level of interest for specific resources. For the IMS data sharing environment, these entries represent record and block locks which IMS has acquired with *update intent*. It does not mean only those resources which IMS has updated, but those resources which IMS *might have updated*. If an IMS database record was retrieved with an update processing option (PROCOPT), then the lock for that resource goes into this record list. These record data entries in the record list are used only for recovery when a connector (IRLM) fails. They are not used for granting or rejecting locks at any other time. They are not used, for example, to resolve the potential conflict indicated by the lock table.

We will discuss how these are used when we discuss XES lock services.

XES lock services

The IRLM uses XES services to manage the global locks in an IMS data sharing environment. We talk about only two of these services, because these are necessary to understand the concepts of global data sharing.

Requesting ownership of a resource (OBTAIN)

When the IRLM wants to *obtain* a resource lock on behalf of one of its (IMS) clients, it issues an OBTAIN request (IXLLOCK REQUEST=OBTAIN). There are many parameters on this request, but the few that are significant to us are the resource name (what resource is being locked), the hash value (index into the lock table), the requested state (shared or exclusive), and the information to be put into the record data entry in the record list.

When the request is made, the hash value is used as an index into the lock table. The lock table entry will be updated to show interest by the requestor in this entry. If there is no other interest by any other lock manager, the lock table entry is updated with the ID of the requesting IRLM as the global lock manager and the requestor is informed that the lock may be granted. If there is other interest already in the lock table entry, then there might be a conflict with a lock held by some other requestor's client (by another IMS). The *contention exit* of the global lock manager already assigned to this entry is driven to resolve the potential conflict. If there is none, then the lock is granted; if the request is for the same resource, then the requestor must wait until the holder releases the lock.

Releasing ownership of a resource (RELEASE)

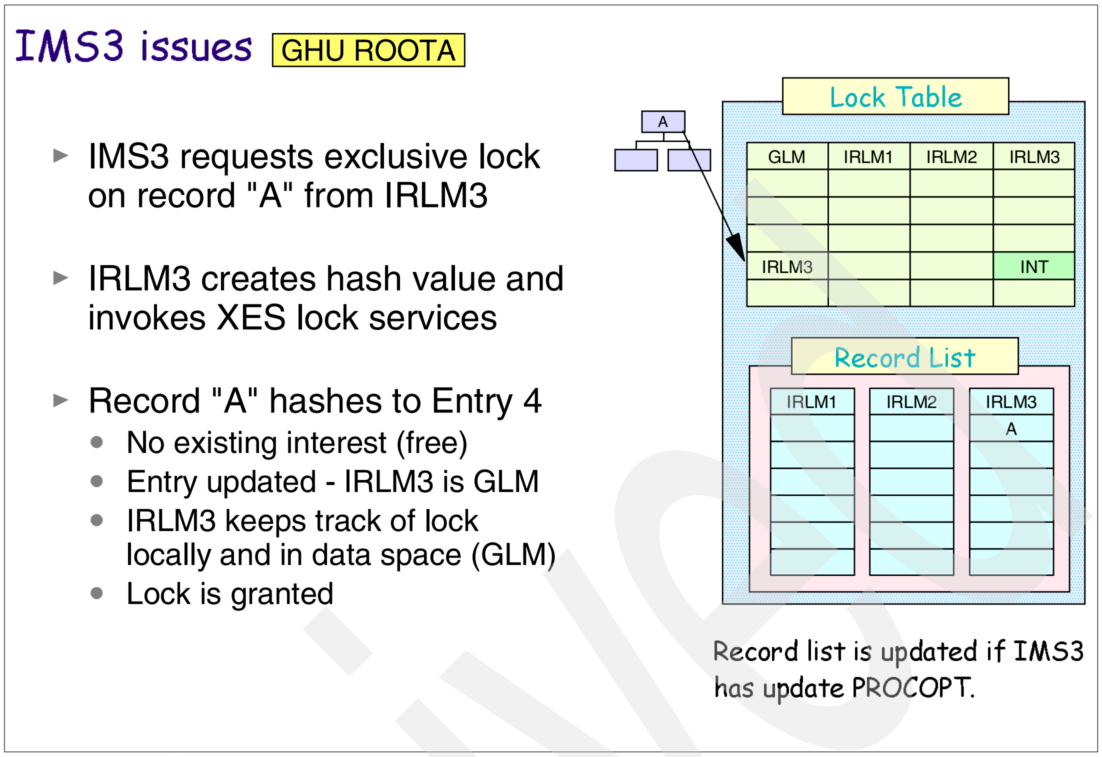
When the holder of a lock (IMS) is ready to release it, the IRLM issues a RELEASE request with the resource name, the hash value, and an indication to delete the record data. At this time, if there were waiters for the resource, the next one on the wait queue gets the lock and processing continues.

Global lock manager

One of the IRLMs is chosen as the global lock manager, or contention manager, for each lock table entry with some interest. It is usually the first one to request a lock that hashes to that entry, but in some cases it could be another IRLM. When a lock request is hashed to an entry with existing interest, and therefore with an existing global lock manager, that global lock manager's contention exit is driven to determine whether the lock request is for the same resource, and if so, are the locks compatible (for example, two share lock requests are compatible). The global lock manager will keep information about each resource (for example, the resource name and the held lock state) in a data space. When the contention exit is driven, that IRLM can just look in the data space to resolve the possible contention. It will add the new requestor's lock request information in the data space and, if a third requestor comes along, the global lock manager can make a decision without having to query any other lock manager.

Tracing a lock request

The next few figures illustrate what happens when first one IMS, and then a second IMS, issues lock requests for different resources that hash to the same value in the lock table.



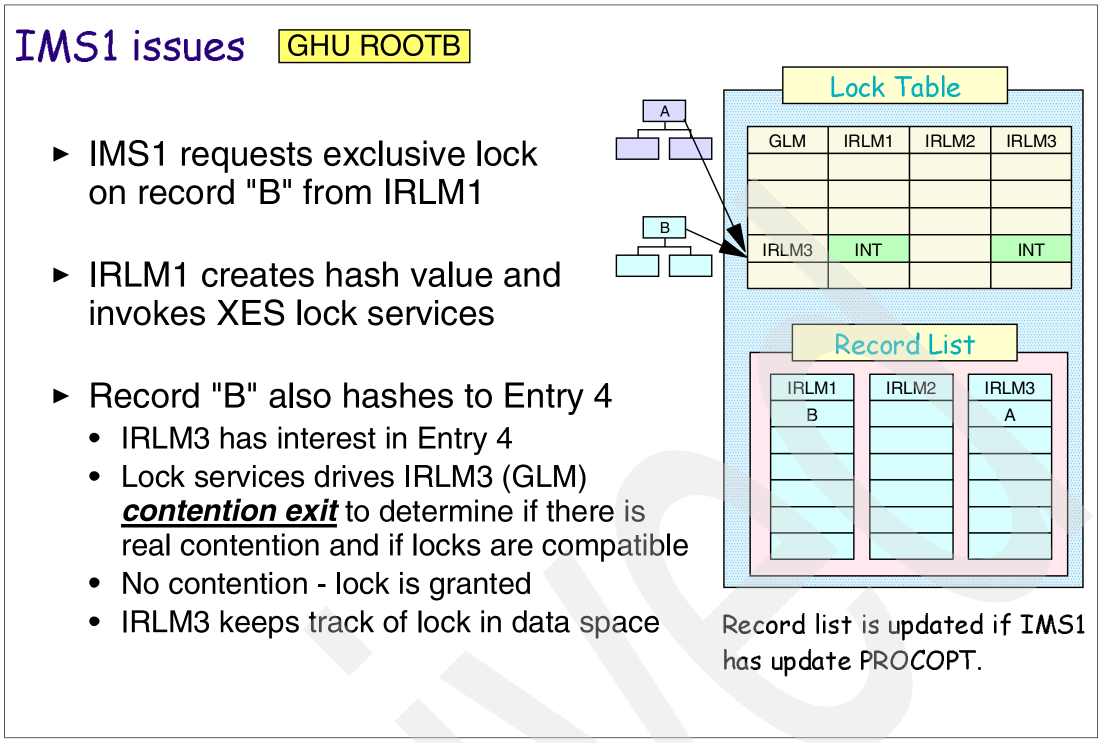
IMS3 issues GHU ROOTA

- IMS3 requests exclusive lock on record "A" from IRLM3
- IRLM3 creates hash value and invokes XES lock services
- Record "A" hashes to Entry 4
  - No existing interest (free)
  - Entry updated - IRLM3 is GLM
  - IRLM3 keeps track of lock locally and in data space (GLM)
  - Lock is granted

Record list is updated if IMS3 has update PROCOPT.

Figure 2-9 Requesting an exclusive lock - first requestor

In the example started in Figure 2-9, IMS3 has issued a lock request to IRLM3 for an exclusive lock on ROOTA. IRLM3 uses the resource name to create a hash value (HV=4) and then submits an XES lock services OBTAIN request. The hash table entry indexed by HV=4 shows no existing interest in any resource that would hash to this entry. The entry is updated to show interest by IRLM3 and sets IRLM3 as the global lock manager for this entry. Record data is added to the record list with information passed on the lock request, including the resource name and lock holder. The lock is granted.



IMS1 issues GHU ROOTB

- IMS1 requests exclusive lock on record "B" from IRLM1
- IRLM1 creates hash value and invokes XES lock services
- Record "B" also hashes to Entry 4
  - IRLM3 has interest in Entry 4
  - Lock services drives IRLM3 (GLM) ***contention exit*** to determine if there is real contention and if locks are compatible
  - No contention - lock is granted
  - IRLM3 keeps track of lock in data space

Record list is updated if IMS1 has update PROCOPT.

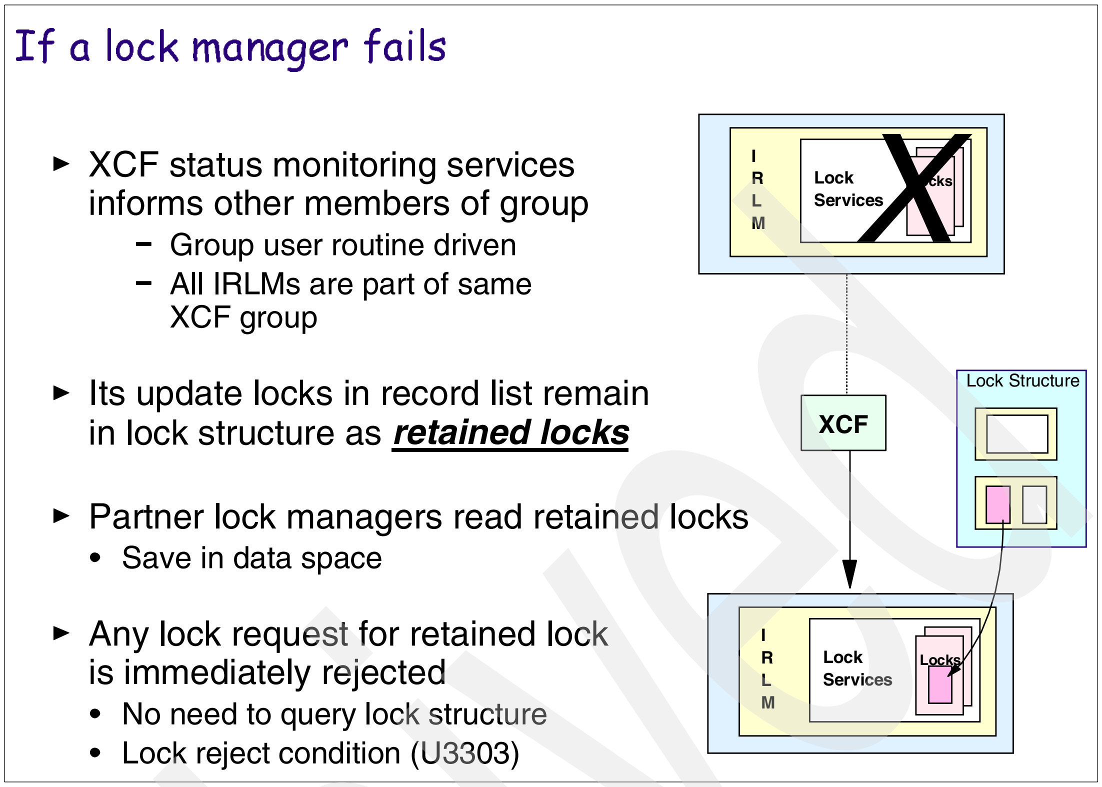
Figure 2-10 Requesting an exclusive lock - second requestor

Figure 2-10 continues the example. In this illustration, IMS1 has requested a lock on ROOTB. The resource name hashes to the same value as ROOTA (HV=4). When the lock request is issued to lock services for an exclusive lock on ROOTB, the lock table entry shows that there is already some interest in this entry, but we cannot tell from this whether it is for the same resource. IRLM3's contention exit is driven to resolve the potential conflict. IRLM3 has kept the information about the other lock in a data space and can determine immediately that the requests are for two different resources and the lock can be granted. IRLM1's interest is updated in the lock table entry, a record data entry is created, and IRLM3 keeps track of IRLM1's locked resource in a data space.

Handling an IRLM failure

If an IRLM fails, the locks that it held cannot be managed, and they will not be released until the IRLM is restarted. This may be immediately, or it may be a long time. When an IRLM fails, other IRLMs which are part of the same data sharing XCF group are informed by XCF that one IRLM has failed. Each surviving IRLM reads from the lock structure the record data entries of the failed IRLM into a data space. These locks are considered *retained locks* and, if the IRLM gets a request for any of these locks, instead of invoking XES services, the IRLM will immediately reject the lock and the application program will abend with a U3303.

Figure 2-11 illustrates what happens when an IRLM fails and surviving IRLMs are notified by XCF.



If a lock manager fails

- XCF status monitoring services informs other members of group
  - Group user routine driven
  - All IRLMs are part of same XCF group
- Its update locks in record list remain in lock structure as ***retained locks***
- Partner lock managers read retained locks
  - Save in data space
- Any lock request for retained lock is immediately rejected
  - No need to query lock structure
  - Lock reject condition (U3303)

Figure 2-11 Managing IRLM failures

Lock structure sizing considerations

Locking services will work no matter how big the lock table is, but if there are a lot of false contentions, then there will be a lot of additional overhead to resolve the potential conflict. Ideally we would like to have no potential contentions, but since this is not really possible, we would at least like it to be a very, very small probability. Therefore, the size of the structure and the size of the lock table should be big enough to keep this probability low. A lock table entry may require two, four, or eight bytes, depending on how many IRLMs may be connected to it. A two-byte entry can handle up to six IRLMs. Assuming there are no more than six IRLMs in your data sharing group, a 32 MB structure with a 16 MB lock table will have 8 million entries. If there are seven to 23 IRLMs, then the entry size is four bytes and the 16 MB lock table will contain only four million entries. And 24 to 32 IRLMs will require 8-byte entries, producing only 2 million entries. The first connector sets the size of the lock table and the maximum number of connectors. It is important that the IRLM sets this at the lowest possible value.

Help with sizing the IRLM lock structure can be found at:

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

2.6.5 List structures and list services

List services allow subsystems and authorized applications running in the same Parallel Sysplex to use a Coupling Facility to share data organized in a list structure. The list structure consists of a set of lists and an optional lock table. Information is stored on each list as a series of list entries. The list entries on one list may be kept in sorted order by key value. The lock table can be used to serialize on resources in the list structure, such as a particular list or list entry. A list structure with a lock table is called a serialized list structure.

While cache and lock services have very distinct purposes, the connector to a list structure may use it to keep *any kind of information* that needs to be shared. For example, in an IMSplex, list structures and list services are used for three functions:

- IMS uses the *resource list structure* for resource management. This structure is used to keep *status information* about the resources in the IMSplex, such as which nodes are logged on to which IMS, which users are in a conversation and what the conversation ID is, what the names are of the online change data sets, and other information related to the IMSplex.
- CQS uses up to four *shared message queue list structures* for shared message queues. These structures are used to keep *messages* (IMS full function and Fast Path EMH transactions and responses) that need to be shared among all the IMSs in a shared queues (XCF) group.
- CQS uses the System Logger to log activity to the shared queues structures. The System Logger uses one or two (full function and Fast Path) *logger list structures* to support a *shared log stream* among all the CQSs in the shared queues group.

Other users of list structures include XCF for signalling, JES2 checkpoint, VTAM Generic Resources and multinode persistent sessions, and just about anything that needs to be shared across the sysplex and which does not meet the specific requirements of cache or lock services.

Components of a list structure

The list structure has more components that either cache or lock structures. Figure 2-12 shows the major components of a list structure. Some of these components exist in every list structure, and some are optional. The first connector to a list structure determines its attributes and whether or not the structure will have these optional components.

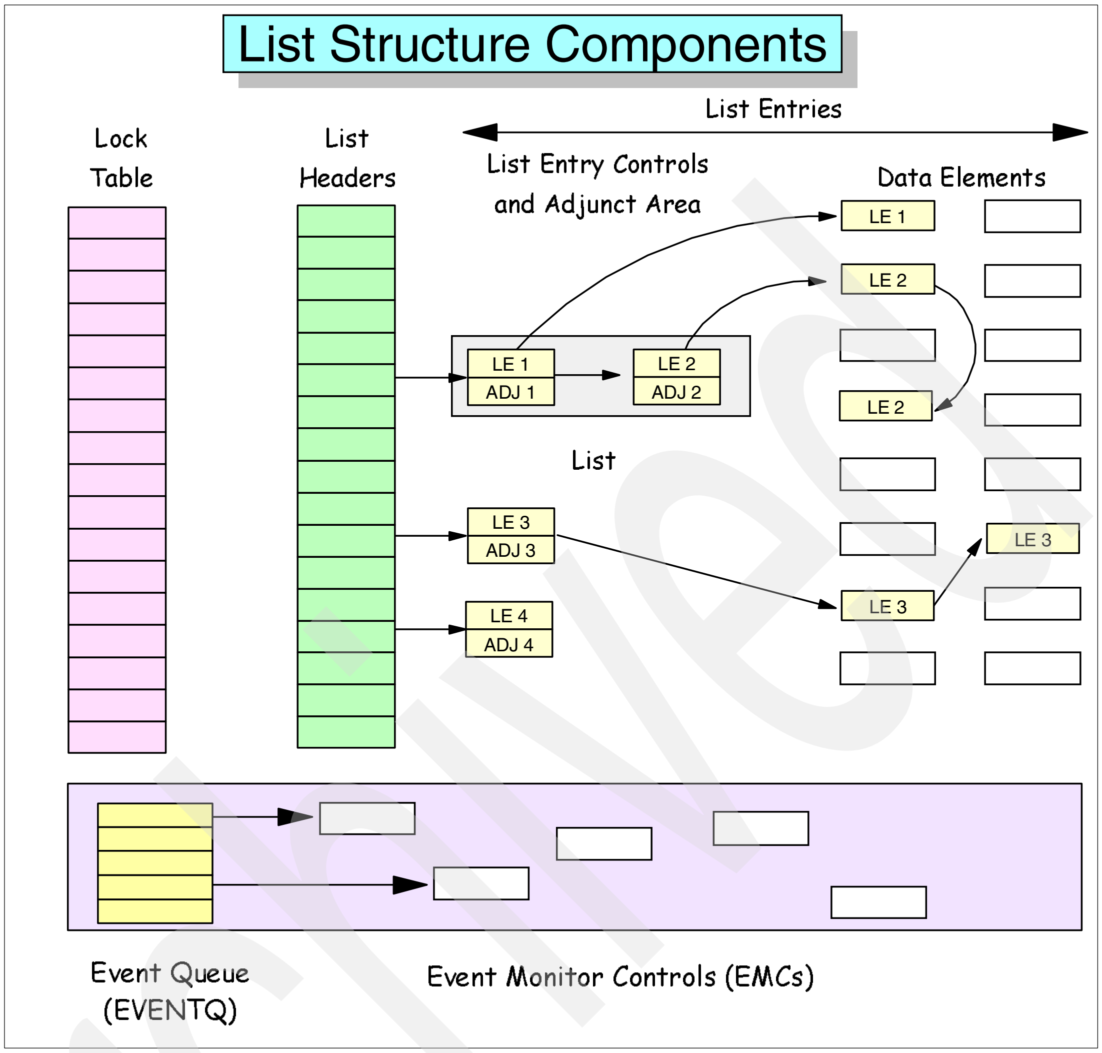


Figure 2-12 List structure components

The following describes these components and how they are used.

- List headers

  List headers are the anchor points for lists. The first connector to a list structure determines how many list headers the structure should have. Each list entry is chained off one list header. There is no preset meaning to a list header. The connector determines how the list headers are to be used. The Common Queue Server (CQS) for IMS shared queues allocates 192 list headers in a shared queues structure. It assigns some of them for transactions, some for messages to LTERMs, some for MSC links, and so on. Some list headers are used by CQS itself to help manage the shared queues.

- List entries

  This is where the information is kept. A list entry can consist of one, two, or three components — list entry controls, an adjunct area, and a data entry. All list entries on a list header form a *list*. All list entries on a list header that have the same entry key form a *sublist*.

  – List entry controls

    Each list entry must have a list entry control (LEC) component. The LEC contains the identification information for the list entry, such as the list header number, the list entry key (ENTRYKEY), the list entry id (LEID), the list entry size (how many data elements in the data entry, if any), and other attributes of the list entry. For CQS, the entry key includes the IMS destination, such as a transaction code or LTERM name. The data entry is the IMS message itself.

  – Adjunct area

    Each list entry can have an optional adjunct area. Whether or not there are adjunct areas is determined by the first connector. If they exist, then every list entry has one. Each adjunct area is 64 bytes and can contain whatever information the connector wants. CQS allocates the shared queues structures with adjunct areas. The adjunct area is used by CQS for internal controls.

  – Data entry and data elements

    The first connector determines whether the list structure is to be allocated with data elements, and the size of the data element. Each list entry may have a data entry associated with it. A data entry consists of one or more data elements, depending on how long the data is. For CQS, an IMS message is one data entry with one or more 512-byte data elements. There are also list entries with data entries for CQS control information.

- Lock table

  The first connector may optionally allocate a lock table. The lock table can be used to serialize a process, for example, to serialize access to a list. Access to an individual list entry is serialized by list services and the CFCC without having to serialize access to the list. CQS allocates a lock table and uses it to serialize access to its CONTROLQ (list header zero).

- Event monitor area

  Space within the structure can be allocated for event monitoring. The first connector can allocate a percentage of the list structure to be used for an event queue (EVENTQ) and event monitor controls (EMCs). These are used to monitor events such as the arrival of a message on a particular list, or a message with a particular entry key (sub-list). When a monitored event occurs, the connector is informed. The event is that an event queue or a list has *transitioned* from empty (no entries) to non-empty (one or more entries). CQS uses *event queue monitoring*.

  – Event queue (EVENTQ)

    When event queue monitoring is requested by the connector, an EVENTQ is created in the structure with *one entry for each connector*.

  – Event monitor controls (EMCs)

    A connector indicates what it wants to monitor by *registering interest* in a list (list number) or a sublist (the entry key). When interest is registered, an EMC is created for that connector and list or sublist. IMS tells CQS which sublists to monitor (for example, a particular transaction code) and CQS registers interest in that sublist using XES list services. IMS registers interest in message destinations, such as transaction codes, LTERM names, and MSC links (MSNAMEs).

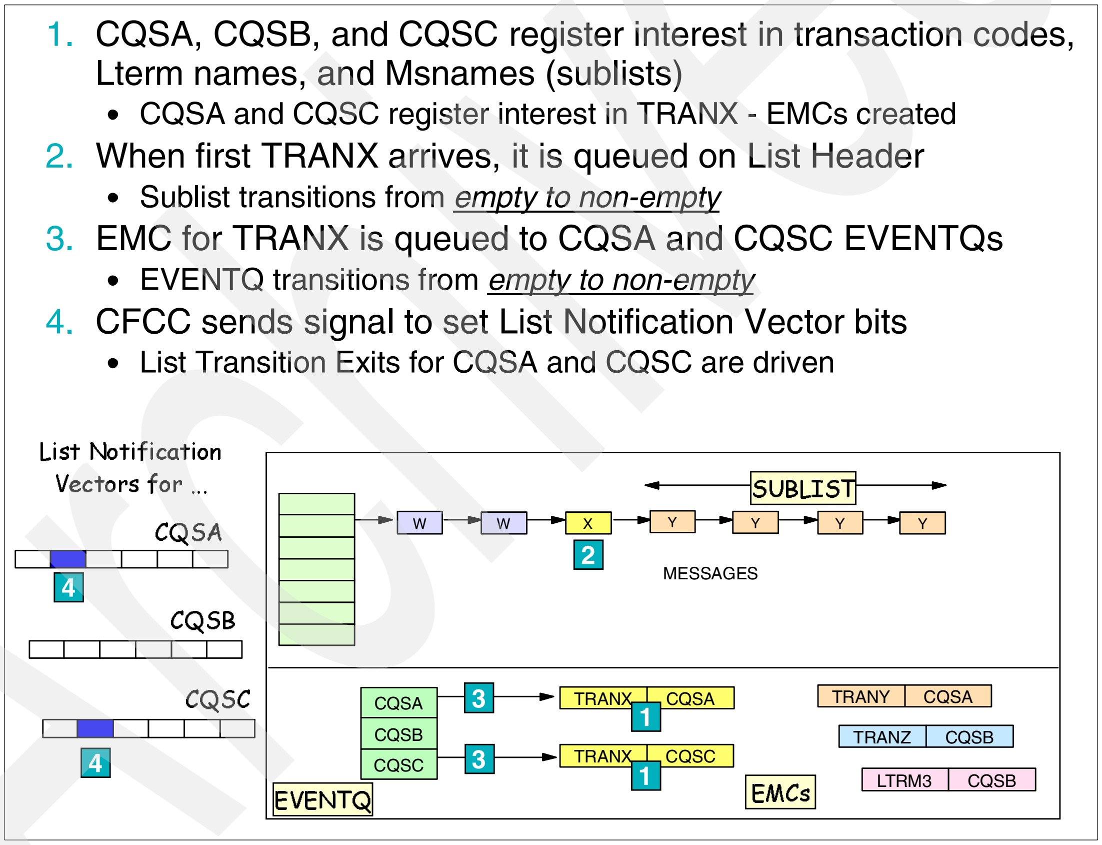
It is not a requirement for a list structure to have an event monitor area. The resource structure in IMS Version 8, for example, does not invoke list or event queue monitoring and so does not have this part of the structure. When monitoring is not requested by the connector, it is the responsibility of the connector to determine what is on the structure - there is no notification of changes.

Event queue monitoring

When EVENTQ monitoring is requested by the connector, a *list notification vector* is created in the hardware system area (HSA) of the connector's CPC. This vector is used similarly to the way the local cache vector is used for cache structures — to inform the connector that something has occurred. In this case, there is only one bit per connector (entry on the EVENTQ).

When an EMC is queued to an EVENTQ, the EVENTQ transitions from *empty to non-empty* and the list notification vector for that connector is set causing the *list transition exit* specified by the connector to be driven. It is the responsibility of the list notification exit to determine what the event was (that is, which EMC was queued to the EVENTQ). An XES list service request called DEQUEUE_EVENT will read the EMC on the event queue and dequeue it. The EVENTQ is now empty again and an EMC will not be queued again to the EVENTQ until the sublist becomes empty again and then non-empty.

Figure 2-13 and Figure 2-14 show an example of CQS monitoring of the shared queues structures.



1. CQSA, CQSB, and CQSC register interest in transaction codes, Lterm names, and Msnames (sublists)
   - CQSA and CQSC register interest in TRANX - EMCs created
2. When first TRANX arrives, it is queued on List Header
   - Sublist transitions from _empty to non-empty_
3. EMC for TRANX is queued to CQSA and CQSC EVENTQs
   - EVENTQ transitions from _empty to non-empty_
4. CFCC sends signal to set List Notification Vector bits
   - List Transition Exits for CQSA and CQSC are driven

Figure 2-13 Example of CQS event queue monitoring

Note that, in the above example, the EMC for TRANY is not queued to the EVENTQ. This is because it only occurs when the TRANY sublist transitions from empty to non-empty. When the first TRANY arrived, this would have happened, the same as it is doing for TRANX. It will not occur again for TRANY unless all the messages for TRANY are read and dequeued by IMS and the sublist becomes empty and then non-empty again.

Note also that, although there are TRANW messages on the queue, no CQS has registered interest so there is no EMC for TRANW. When (if) any CQS does register interest, an EMC will be created and immediately queued to the EVENTQ.

5. List Transition Exit notifies CQS that "something" has arrived
   - Don't know what yet
6. CQS reads EVENTQ and finds EMC for TRANX
   - Dequeues EMC; EMC won't be requeued until sublist becomes empty and then non-empty; LNV bit turned off
7. CQS notifies IMS that TRANX message has arrived
8. When dependent region is available, IMS requests CQS to read message from list structure
   - If only one message, only one IMS will be successful

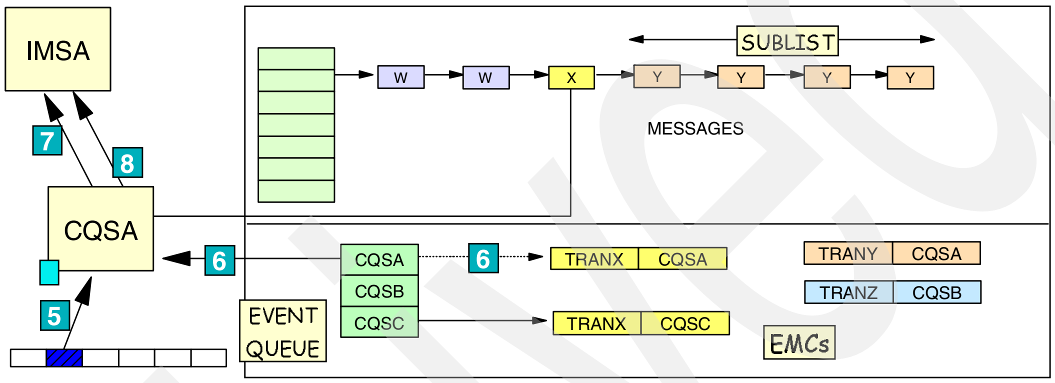


Figure 2-14 Example of CQS EVENTQ monitoring (continued)

The enqueuing of the EMC to CQSA's EVENTQ causes CQSA's list notification vector to be set, which in turn causes the list transition exit to be driven. All CQS knows at this time is that there is something on the queue, but does not know what. The exit will invoke list services to read the EMC and dequeue it from the EVENTQ. Now that CQS knows what was on the queue, it can notify IMSA. IMSA will, eventually, read the transaction from the shared queues, process it, and delete it. A similar sequence of events will occur for CQSC and IMSC, except that this time, when IMSC tries to read the message, it will already be gone (unless another arrived in the meantime). This is called a *false schedule* in IMS and incurs some wasted overhead. However, this technique is self-tuning as workload increases. The shared queues planning chapters in this book will discuss this and suggest ways to minimize the occurrence of false schedules.

Other list services

Besides the monitoring of lists, sublists, and event queues, other XES services allow the connector to:

- Read, write, move, or delete an entry in a list structure
- Combine operations:
  - Read an entry and delete it
  - Move an entry and update it
  - Move an entry and read it
- Read or delete multiple entries with a single request
- Serialize a list entry (using the lock table) and perform a list operation

CQS uses all of these services in support of IMS shared queues and the Resource Manager.

2.7 Other connection services

Besides connecting to and disconnecting from a structure, XES connection services also support several other request types.

2.7.1 Structure delete

When the structure is no longer required, it can be deleted. This may be done automatically if the structure is not persistent when there are no more active or failed persistent connections. If the structure is persistent, and therefore not deleted when all connections are gone, it may be deleted using an operator command. The command will only work if there are no connections. When this command is entered, XCF will invoke XES connection services to delete the structure:

```
SETXCF FORCE,STRUCTURE,STRNAME=structure-name
```

When there are one or more failed-persistent connections, which will occur if the structure (for example, an IMS shared queues structure) is defined with persistent connections and a connector (for example, CQS) fails. To delete the structure, you must first delete the failed-persistent connection. This can be done by entering the following commands:

```
SETXCF FORCE,CONNECTION,STRNAME=structure-name,CONNAME=connection-name or ALL
SETXCF FORCE,STRUCTURE,STRNAME=structure=name
```

In an IMS shared queues environment, the shared queues structures and their connections are persistent. Even though all of the IMSs and CQSs are down, the message queue structures will persist. If you want to get rid of them, the equivalent, for example, of an IMS cold start without shared queues, you must delete the structures.

2.7.2 Structure event notification

As mentioned earlier, all connectors to a structure belong to an XCF group. For example, all IRLMs connected to a lock structure belong to the same XCF group. When connection services are invoked to connect to the structure, an *event exit* is specified. Certain events related to the structure itself (as opposed to the data within the structure) cause this event exit to be driven. Examples of events that drive this exit are:

- New connection to a structure
- Existing connections
- Disconnection or failed connection
- Loss of connectivity by any connector
- Structure or Coupling Facility failure
- Several rebuild, alter, and duplexing events

2.7.3 Structure rebuild

Structure rebuild is the construction of a new instance, or a secondary instance, of a structure. When a connector connects to a structure, it may specify one or more of the following parameters:

- ALLOWREBLD=YES means that the connector will support *user-managed rebuild* of the structure.
- ALLOWDUPREBLD=YES means that the connector will support *user-managed duplexing* of the structure.
- ALLOWAUTO=YES means that the connector will support *system-managed processes*, which include system-managed rebuild and system-managed duplexing. These processes

also require the CFRM couple data set to be formatted to support system-managed processes (SMREBLD and SMDUPLEX) specified in the CDS format utility.

User-managed rebuild

There must be at least one active connector to the structure and all connectors must agree to coordinate the rebuild process. Briefly, this means that the connectors coordinate to quiesce activity, to connect to the new instance, to propagate data to the new structure instance, and then to signal completion, at which time the original instance is deallocated (deleted).

User-managed rebuild can be used to move a structure from one Coupling Facility to another, or to recover from a loss of connectivity. It may be initiated by one of the connectors, or by operator command:

```
SETXCF START,REBUILD,STRNAME=structure-name<,LOC=OTHER>
```

Specifying LOC=OTHER causes the structure to be rebuilt on another Coupling Facility and requires at least two Coupling Facilities to be specified in the PREFLIST in the CFRM policy.

System-managed rebuild

System-managed rebuild may be invoked when there are no active connectors to the structure. If there are active connectors, they only have to agree to quiesce activity until the system has rebuilt the structure.

System-managed rebuild will not rebuild a structure due to loss of connectivity, structure failure, or Coupling Facility failure. Such a rebuild requires an active connector and user-managed rebuild.

User-managed duplexing rebuild

Duplexing is a special form of rebuild where a secondary structure is allocated in another Coupling Facility than the primary structure, and all operations are directed to both instances of the structure. The purpose of duplexed structures is that, if one structure or Coupling Facility fails, structure operations can continue using the remaining structure. No IMS-related connectors support user-managed duplexing.

Note: User-managed duplexing should not be confused with Fast Path support for dual shared VSO structures.

System-managed duplexing rebuild

As its name implies, a structure can be duplexed without direct connector participation, other than to respond to the start-up event notification. A system-managed duplexing rebuild can be initiated even if there are no active connectors. Once the duplexing rebuild is complete, the connector issues requests only to the primary instance of the structure and the system propagates the operation to the secondary instance transparently to the connector. If one instance fails, for example because of a structure or Coupling Facility failure, connector operations continue with the remaining structure. When another suitable Coupling Facility becomes available, duplexing will be automatically reinstated. Except for the OSAM and VSAM cache structures, all IMS structures support duplexing.

Structure duplexing requires not only that the connector indicate support during connection, but it also requires the CFRM couple data set to be formatted with SMDUPLEX, and for the structure definition in the CFRM policy to specify DUPLEX(ALLOWED) or DUPLEX(ENABLED). When duplexing is enabled, the structure will be duplexed automatically by the system when the first connector allocates the structure. When duplexing is allowed, it must be initiated, or terminated, by operator command.

```
SETXCF START,REBUILD,DUPLEX,STRNAME=structure-name
SETXCF STOP,REBUILD,DUPLEX,STRNAME=structure-name
```

2.7.4 Structure alter

The structure alter function allows a Coupling Facility structure to be reconfigured with minimal disruption to connectors. Alter can change the size of a structure, the entry-to-element ratio, and the alteration of the percentage of structure storage set aside for event monitor controls (EMCs). Structure alter requires that the connector specify the following when connecting to the structure:

```
ALLOWALTER=YES
```

Structure size can be altered only within the maximum and minimum sizes specified on the structure definition statements in the CFRM policy (SIZE and MINSIZE). It may be initiated by a connector or by the alter command:

```
SETXCF START,ALTER,STRNAME=structure-name,SIZE=new-size
```

When the command is used to alter the size of a lock structure, the size of the lock table itself is not changed, only the amount of storage allocated for the record data entries.

Other structure characteristics, such as changing the entry-to-element ratio of a list structure, can only be initiated by a connector (or by the system if autoalter is enabled). For IMS structures, only CQS will initiate an alter, and then only to alter the size if it detects that the structure has reached some user-specified threshold of data elements in use.

Autoalter

When autoalter is enabled, the system may initiate a structure alter based on one of two factors:

- If the system detects that the storage in use in a structure has reached a structure full threshold specified in the policy, it may increase the size to relieve the constraint. At this time it may also alter the entry-to-element ratio if it determines that it does not represent the actual utilization of list entries and data elements.
- If the system detects that storage in the Coupling Facility is constrained, it may decrease the allocated storage for a structure that has a large amount of unused storage.

Autoalter is enabled only under the following conditions:

- ALLOWALTER=YES is specified during connect processing to enable the alter process.
- ALLOWAUTO=YES is specified during connect processing to enable system-managed processes.
- ALLOWAUTOALT(YES) is specified in the structure definition in the CFRM policy.

Also FULLTHRESHOLD(value) can be specified in the CFRM policy to override the default of 80% full.

2.8 Other sysplex services

In addition to the services described above, OS/390 and z/OS also offer you some additional services implemented through system programs taking advantage of some of these same XCF and XES services.

Workload Manager (WLM)

The Workload Manager is an OS/390 service that helps you better control the allocation of system resources to the jobs and transactions that are most important to your business. System resources include CPU, central and expanded storage, and DASD I/O paths. The WLM can operate in two modes:

- Compatibility mode: Allocation of system resources is managed by the system resource manager (SRM) according traditional techniques, including PARMLIB members IEAIPS, IEAOPT, and IEAICS.
- Goal mode: Allocation of system resources is managed according to performance goals set in the WLM policy. Performance goals can include:
  - Response time goals: The end-to-end response time of an IMS or CICS transaction
  - Velocity goals: The speed with which resource requirements are met
  - Discretional goals: Do the best you can after other goals have been met

As a user, you get to decide which mode using the z/OS modify command:

```
F WLM,MODE=GOAL - or -
F WLM,MODE=COMPAT
```

When running in *goal mode*, you must have defined your goals in one or more *WLM policies* in the WLM couple data set (WLM CDS) and then activate one of these policies according to your own objectives at the time they are activated:

```
VARY WLM,POLICY=xxxx <where xxxx = policy name>
```

You may have a different policy for an off-shift batch workload environment than for a prime-shift transaction oriented workload; or different policies for weekdays and weekends. Whatever your objectives, you must define in the policy a workload management construct consisting of:

Service policy	At least one is required; may have multiples; only one active at a time
Workloads	*Arbitrary* name used to group service classes together; for example, IMSP (IMS production), IMSD (development), IMST (test), TSOSP (systems programming), and so forth
Classification rules	Associates unit of work (transactions, batch jobs, etc.) with service class; for example, an IMS workload may be classified by its subsystem type (IMS), by its subsystem ID (IMSA), by a transaction code (TRANX), transaction class (3), userid (CIOBOB), or several other criteria.
Service classes	Associated with workload; contains service goals for that work; you may assign *importance* to favor workload when resources are limited
Performance goals	The desired level of service WLM uses to allocate a system resource to a unit of work; for transactions, goal may be percentage within response time (for example, 95% of TRANX should respond in less than 0.30 seconds)

Running WLM in goal mode may have several side effects in the IMSplex, other than the obvious effect of helping you allocate system resources to the important work in the system:

- JES may route batch work (such as BMPs) to an OS/390 image, which is best meeting its batch performance goals. For a BMP, this means that the IMSID= parameter in the BMP must match the IMSGROUP parameter in DFSPBxxx, enabling the BMP to connect to whichever IMS happens to be executing on the same image.
- VTAM Generic Resources may route a logon request to that IMS, which is best meeting its response time goals, overriding VTAMs attempt to distribute the logons evenly across the

members of the VTAM generic resource group (see “VTAM Generic Resources (VGR)” on page 46).

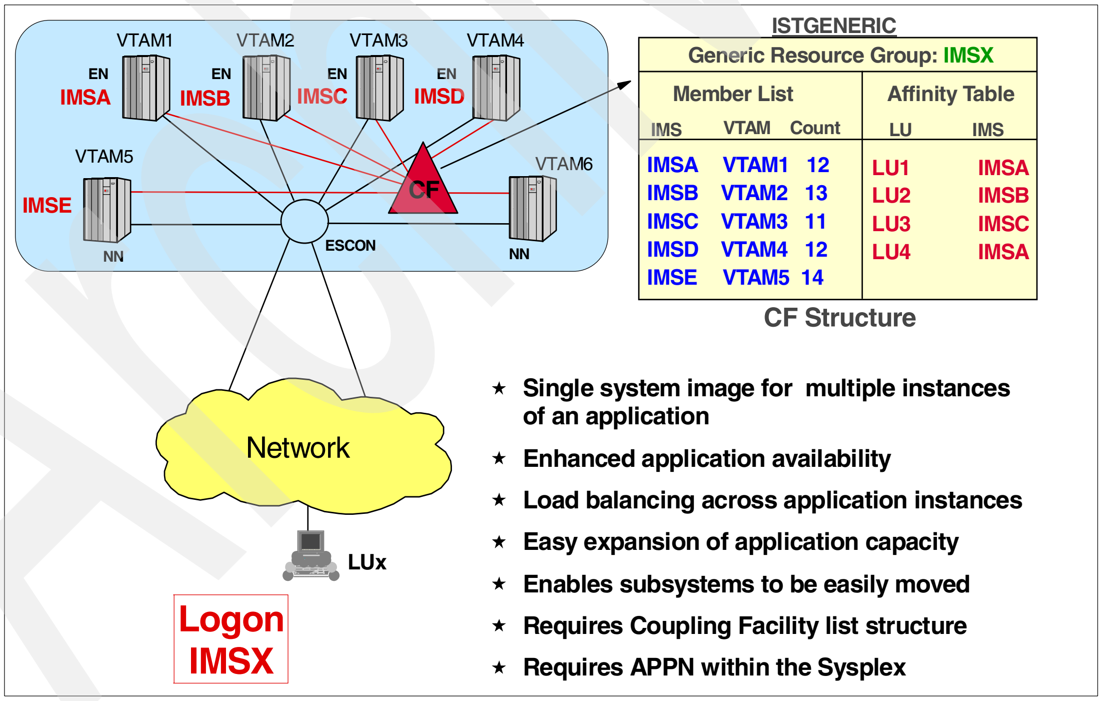
VTAM Generic Resources (VGR)

VTAM, a component of the OS/390 Communications Server, provides a service to VTAM users called *VTAM Generic Resources* support. With this support enabled, a remote end-user logical unit (LU) can log on to an application using a generic name instead of the VTAM APPLID. The application must first have joined a *VTAM generic resource group* (VGR group). The benefit is that you do not have to know which instance(s) of that application are active at the time you log on. A prime example is if you are an IMS user wanting to log on to any of several IMSs in a shared queues group. You see the entire group as a single image.

When IMS restarts, it may optionally join a VGR group using the GRSNAME specified in the DFSDCxxx PROCLIB member. When the first IMS joins this group, VTAM dynamically creates the group in a Coupling Facility list structure and adds this first IMS as a member in the *member list.* As other IMSs join the group, they are added to the member list.

When you log on using the generic resource name, VTAM makes a determination among the active members of the group as to which IMS your logon request will be routed. Your LU is then added to an *affinity table* in the list structure.

Figure 2-15 shows several IMSs running on several OS/390 images in the same Parallel Sysplex. All the IMSs have joined a VGR group called IMSX. VTAM maintains a member list in a CF list structure. As each user logs on, the LU name is added to an affinity table in the same structure.



Member List			Affinity Table	
IMS	VTAM	Count	LU	IMS
IMSA	VTAM1	12	LU1	IMSA
IMSB	VTAM2	13	LU2	IMSB
IMSC	VTAM3	11	LU3	IMSC
IMSD	VTAM4	12	LU4	IMSA
IMSE	VTAM5	14		

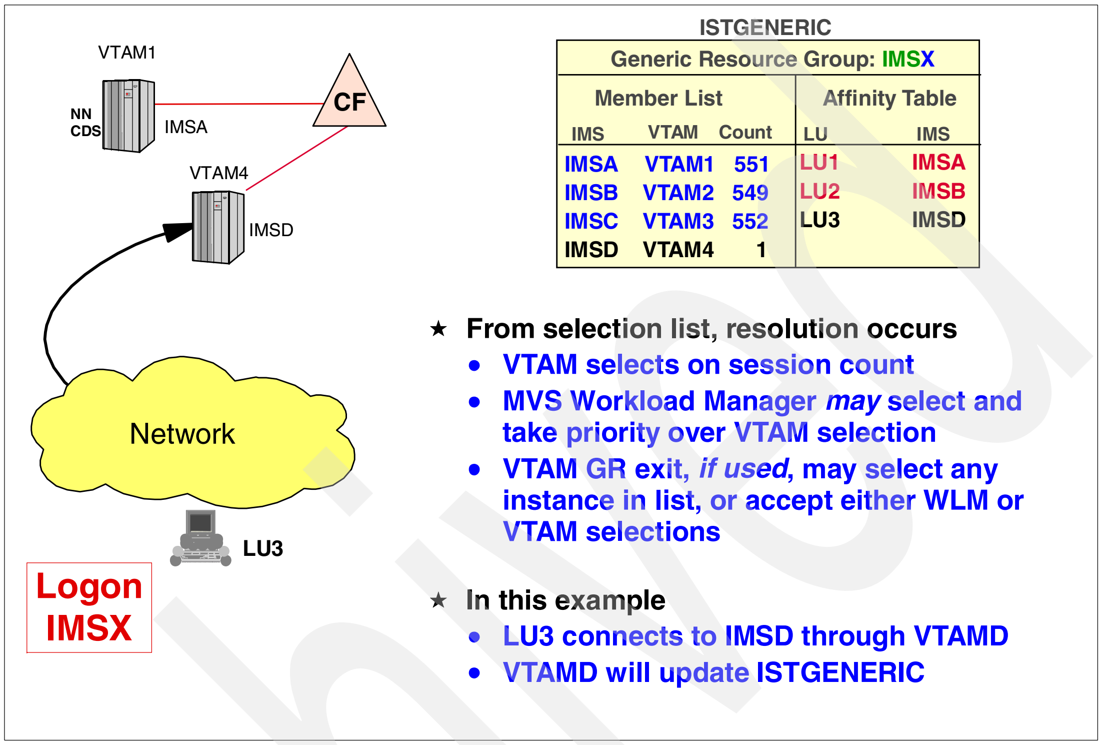
- ⋆ Single system image for multiple instances of an application
- ⋆ Enhanced application availability
- ⋆ Load balancing across application instances
- ⋆ Easy expansion of application capacity
- ⋆ Enables subsystems to be easily moved
- ⋆ Requires Coupling Facility list structure
- ⋆ Requires APPN within the Sysplex

Figure 2-15 VTAM Generic Resources

When you log on using the generic name, VTAM must decide to which IMS your logon request should be sent. If an affinity already exists, then the logon request will be sent to that application with which the LU already has the affinity. If no affinity exists, then there are three possibilities included in VTAM’s decision:

1. Send the logon request to the member with the fewest logged on users.

2. Send the logon request to the member chosen by the Workload Manager, running in goal mode, based on satisfaction of each member's goals (see "Workload Manager (WLM)" on page 45.
3. Send the logon request to a member chosen by the VTAM Generic Resource Resolution Exit (ISTEXCGR). This exit can override either of the above selections.



ISTGENERIC

Generic Resource Group: IMSX

Member List			Affinity Table	
IMS	VTAM	Count	LU	IMS
IMSA	VTAM1	551	LU1	IMSA
IMSB	VTAM2	549	LU2	IMSB
IMSC	VTAM3	552	LU3	IMSD
IMSD	VTAM4	1		

★ **From selection list, resolution occurs**
- **VTAM selects on session count**
- **MVS Workload Manager *may* select and take priority over VTAM selection**
- **VTAM GR exit, *if used*, may select any instance in list, or accept either WLM or VTAM selections**

★ **In this example**
- **LU3 connects to IMSD through VTAMD**
- **VTAMD will update ISTGENERIC**

Figure 2-16 VTAM Generic Resources selection algorithms

Without WLM in goal mode, and without the exit, VTAM will balance the logons across the generic resource group.

When a session is terminated, a decision is made by VTAM or by the application whether or not to delete the affinity. If the affinity is not deleted, the next time you log on using the generic resource name, VTAM will always route your request to the member with which you have the affinity. This is an important consideration in an IMSplex, since the existence of an affinity can be used to return you to an IMS with which you have significant status, for example, in a conversation. You can override this affinity only by logging on to a specific IMS APPLID.

System Logger (IXGLOGR)

The System Logger is a program provided with OS/390 and z/OS that provides an interface for other address spaces to log (write) and retrieve (browse) data. There are numerous users of the System Logger, including:

- OPERLOG and LOGREC
- CICS Transaction Server
- Transactional VSAM
- *IMS Common Queue Server (CQS)*

Users such as CQS which do not have their own internal logging mechanism, but still have a requirement to record data for later restart or recovery, or as an audit trail, may invoke the services of the System Logger (program IXGLOGR) to *connect to a log stream, write data to that log stream, and later to retrieve data from that log stream*. These log streams are physically managed by the System Logger and implemented in a Coupling Facility list structure (users without a CF have the option of writing a log stream to a data set instead of a structure). Figure 2-17 shows CQS writing shared queues updates to a log stream. Note that there are other CQSs using other System Loggers writing to the same log stream.

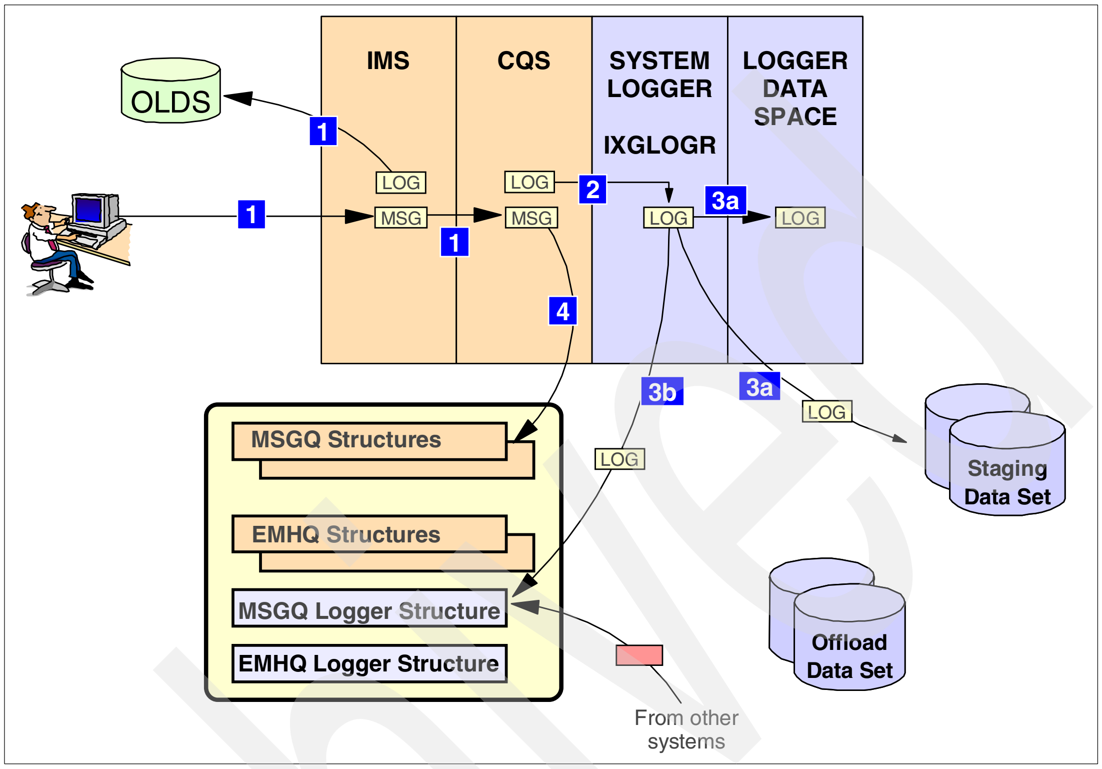


Figure 2-17 CQS writing to the log stream

The log stream can be shared by multiple connectors. For example, multiple CQS address spaces in a common shared queues environment can write to the same log stream. The System Logger will merge the log records from these CQSs into a single, sequential log stream. Figure 2-18 shows two CQSs writing log data to a single log stream with their log records merged by the System Logger.

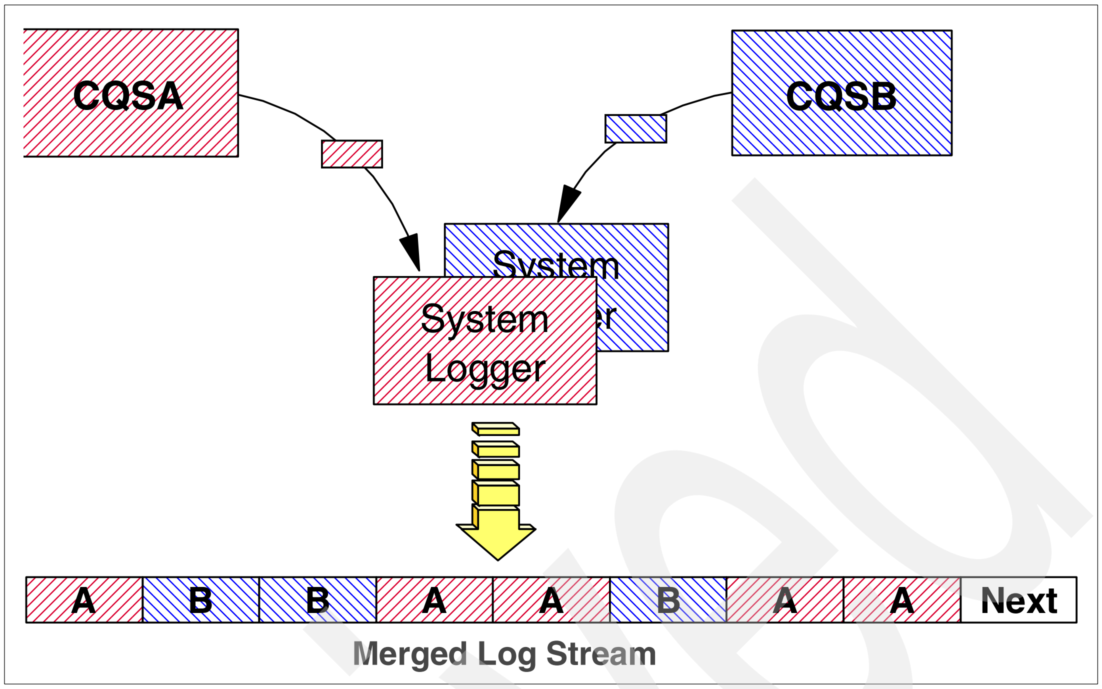


Figure 2-18 CQS merged log stream

If the log records are needed later for recovery or restart, they can be retrieved from the log stream as necessary. CQS uses the log stream to record changes to the shared queues structures. If a shared queues structure requires recovery, CQS will re-initialize the structure using an SRDS and then read the log stream and apply the changes. This is similar to the way IMS rebuild local message queues from a SNAPQ checkpoint and the OLDS/SLDS message queue log records. Figure 2-19 shows how a single CQS can read log records written to the merged log stream by all CQSs in the shared queues group in order to recover a lost or damaged shared queues structure.

If a structure needs to be recovered

- **Any CQS** has access to **ALL log records** by reading the merged logstream

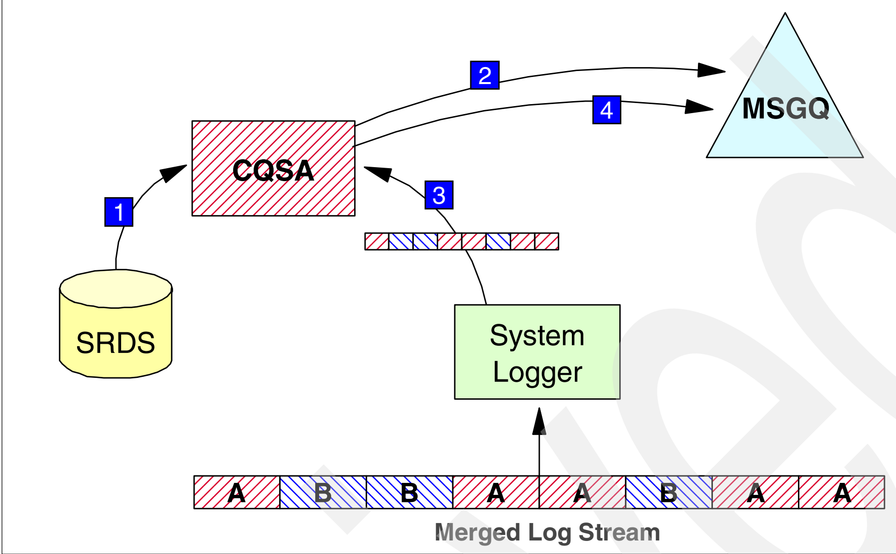
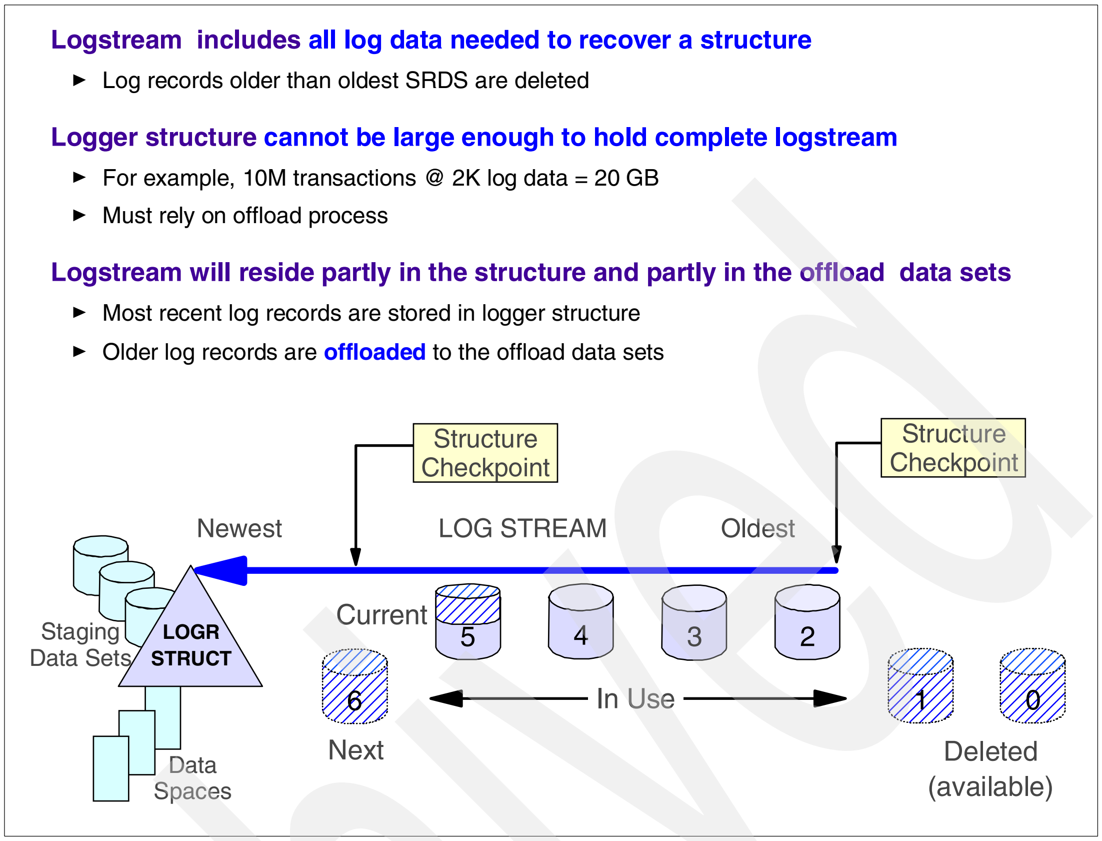


Figure 2-19 CQS reading the merged log stream

Because it is likely that a list structure cannot hold all of the data required in a log stream, as the structure reaches a user-specified full threshold, the System Logger writes the data to one or more offload data sets implemented as up to 168 VSAM linear data sets. When the log data is no longer needed, the connector (for example, CQS) can delete it from the log stream. CQS does not have to know the physical location of data in the log stream - in the structure or on an offload data set. Just issuing requests for the log data will retrieve it from wherever it is located.

Figure 2-20 shows graphically how the log stream exists partly in the logger structure (the more recent records) and partly on the offload data sets (the older log records). When log records are no longer needed for recovery, CQS will delete them from the log stream, releasing the storage.



Logstream includes all log data needed to recover a structure

- Log records older than oldest SRDS are deleted

Logger structure cannot be large enough to hold complete logstream

- For example, 10M transactions @ 2K log data = 20 GB
- Must rely on offload process

Logstream will reside partly in the structure and partly in the offload data sets

- Most recent log records are stored in logger structure
- Older log records are **offloaded** to the offload data sets

Figure 2-20 The CQS log stream

2.9 Objective of Parallel Sysplex development

The primary objective when developing the sysplex, and later the Parallel Sysplex, was to provide a price-competitive computing facility for large commercial processing systems. It is only when you consider the parameters within which this objective was reached that you can appreciate the significance of the achievement of the Parallel Sysplex. These parameters were:

- Preserve a single system image
- Provide customer investment protection (hardware, software, and skills)
- Ensure that current applications benefit
- Provide considerable room for growth
- Provide continuous availability to corporate information resources
- Make it easy to use

2.9.1 Benefits of a Parallel Sysplex configuration

Since the base sysplex and Parallel Sysplex were announced, many benefits have been documented for its use. These include:

- Reduced costs

  Costs can be reduced, or the growth of costs limited, by continuing to use the hardware, software, operational, systems management, and application development technologies, which are compatible with what you have been using for years. Among the sources of cost reduction are the ability of applications to continue to run without change; application developers and system operators to not have to be retrained; and end-users to be able to continue using an interface already familiar to them. Because the total workload, or a single application's workload can be distributed across multiple CPCs, it may not be necessary to have a single CPC capable of handling one application's peak workload and then be under utilized during less busy periods.

- Greater capacity

  Certainly when the sysplex was first introduced, one of the chief benefits was the increased throughput that could be attained by applying the power of two, three, or even thirty-two CPCs to a logically single workload. This is, of course, still true today. Even though the power of a single CPC has increased dramatically since 1990, so has the application workload. In fact, this increased power has led many businesses to develop new applications that were not possible with the smaller, slower, and multi-image processing capabilities without the sysplex.

- Better availability

  Because the application systems taking advantage of Parallel Sysplex capabilities can more closely function as a single system, and any work can process on any OS/390 image, the failure of a single instance of a sysplex member or application subsystem does not completely take down an entire application. Hardware and software systems can be rotated offline incrementally for maintenance while the workload processes on the remaining systems.

- Greater flexibility

  The Parallel Sysplex offers great flexibility in meeting your I/T requirements. The sysplex can be a mixture of older and newer technologies, additional systems can be dynamically and incrementally added (or removed) from the sysplex as business needs dictate, or a sysplex can be reconfigured to meet immediate and sometimes unforeseen requirements.

It is not our intention to present all of the benefits of the Parallel Sysplex. These have been documented in great detail in IBM documentation, trade magazines, trade shows, technical conferences, and so on. We did, however, want to set the stage for why the Parallel Sysplex environment is good for IMS.

2.10 IMS in the Parallel Sysplex

IMS does not have to participate in any Parallel Sysplex functions in order to execute in a Parallel Sysplex environment, however, when it does, we refer to those IMSs which are exploiting the Parallel Sysplex to share data, message queues, and/or status as an *IMSplex*. The IMSplex may consist of IMS control regions (DB/DC, DBCTL, DCCTL), IMS batch regions, and any batch or utility address spaces using DBRC services. Some address spaces with IMS automated operations functions, such as the IMS Version 8 TSO SPOC or IMS Connect, may also be a part of the IMSplex. The IMSplex may be participating with other subsystems such as DB2, CICS, IMS Connect, and WebSphere MQ, even though they are not formally recognized as part of the IMSplex.

IMS exploits many Parallel Sysplex functions, and each new release of IMS adds to that growing list. Each of these functions requires the user to execute a plan for implementing the IMSplex and operating in the IMSplex environment. The list, as of IMS Version 8, includes:

- Block level data sharing
- Shared queues
- VTAM Generic Resources
- VTAM multi-node persistent sessions
- Automatic restart management

- Sysplex communications
- Operations Manager and single point of control
- Resource manager and sysplex terminal management
- Resource manager and coordinated global online change
- Automatic RECON loss notification

The value of a new product or feature is often difficult to quantify, especially because IMS environments vary so widely. However, we believe that Parallel Sysplex technology is the vehicle to carry large and not so large computer applications into the future. A look at the history of IMS since Version 5 will show that each release adds more IMS features that are based on Parallel Sysplex technology. There is little reason to believe that future releases of IMS will not continue this progression into the Parallel Sysplex world — not only for capacity and availability, but for function as well.

This volume addresses those planning considerations, which must be a part of any implementation and operational plan.

3

Planning for block level data sharing

IMS block level data sharing was introduced in IMS/VS Version 1, Release 2. In IMS/ESA Version 5, it was enhanced significantly to exploit the benefits of the Parallel Sysplex and especially *Sysplex Services for Data Sharing*, which provide up to 255 IMSs and 32 IRLMs with the infrastructure and services necessary to share IMS data efficiently and with complete integrity.

IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology, SG24-6908, contains a detailed review of block level data sharing. This volume and this chapter identify some of the planning tasks necessary, either formally or informally, to implement block level data sharing in your IMSplex environment.

This chapter addresses the first two activities in the planning process for data sharing:

- Objectives and expectations
- Functional planning

The next two activities are addressed in Chapter 7, “Putting it all together” on page 159:

- Configuration planning
- Security planning

Data sharing implementation and operations are addressed briefly in this chapter, but are covered more fully in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929.

Like all of the other planning chapters, the objective here is to ask questions which you must answer when developing your plans. Each question is followed by some discussion of what you should consider when answering the question.

3.1 Block level data sharing

The concept of data sharing is not new to IMS. It was first implemented in IMS/VS Version 1.2 to share data between IMS subsystems and/or IMS batch jobs running in one or two MVS images in order to overcome the capacity constraints of the single-processor and tightly-coupled multi-processor environments of the day. Data sharing also provided a vehicle to create high availability environments where the impact to the end-user of outages caused by component failures and planned maintenance were minimized.

Supported by the Parallel Sysplex, IMS Version 5 was the first IMS version to expand the two-way block level data sharing limit of previous IMS releases to N-way data sharing. A maximum of 255 IMS data sharing subsystems (and/or IMS batch jobs) can simultaneously access and update a single set of IMS databases with full data integrity and good performance. These IMS subsystems, with their associated IRLMs, comprise a *data sharing group*. A maximum of 255 connectors to a cache structure are allowed, hence a maximum of 255 IMSs may be a part of the data sharing group. However, since each IRLM connects to a lock structure, and lock structures allow only 32 connectors, you can have "only" 32 IRLMs in a sysplex. Each IRLM can support multiple IMSs, batch and online, on the same LPAR.

3.1.1 IMSplex data sharing components

In order to understand what your data sharing migration plan should contain, it is first necessary to understand the components of an IMSplex data sharing environment.

IMS data sharing exploits the Parallel Sysplex architecture and services to improve greatly on the performance and capabilities of the prior implementation of block level data sharing. The components of the data sharing environment utilize the architecture and services of the Parallel Sysplex, but must *explicitly* take advantage of that architecture and its services. For example, IMS uses cache structures and XES cache services to register interest in data items, to invoke buffer invalidation, and in some cases to cache data. The IRLM uses a lock structure and XES lock services to manage lock requests from its IMS clients. XCF group, signalling, and monitoring services are used to enable the IMSplex components to better communicate and to know what is going on in the IMSplex.

Figure 3-1 shows the major components of a block level data sharing environment - an integration of IMS with OS/390 software services and S/390 hardware functionality, including Parallel Sysplex software services and hardware.

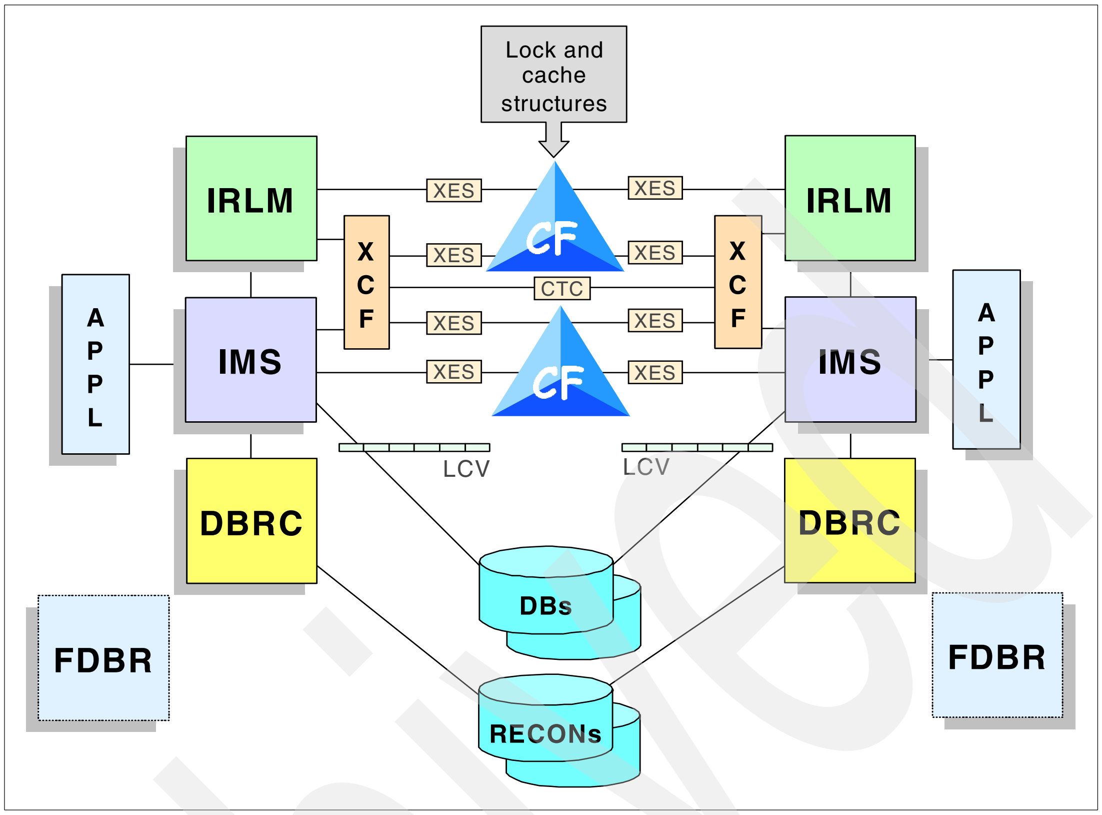


Figure 3-1 Components of data sharing environments

The IMS-related components of an IMSplex sharing data in a Parallel Sysplex are:

- Applications

  The end-user of the data sharing capability provided in the IMSplex is the application program. IMS online applications have always been able to share data with other applications within the same IMS environment. In the IMSplex, these applications can share data with applications running in up to 254 other IMS environments. Although the figure shows online applications, IMS batch applications can also share data with each other and the online systems.

- IMS subsystems and/or IMS batch jobs

  There can be a maximum of 255 IMSs (batch plus online) in the data sharing group. Each IMS must connect to an IRLM for lock services, and before accessing a shared database, must first get authorization from DBRC. Each IMS, batch and online, has its own set of buffers in one or more buffer pools into which IMS reads data or from which IMS writes updated data.

- DBRC and the RECON data sets

  IMS online subsystems and IMS batch jobs use DBRC for authorization processing to control access to the shared databases. Each registered database has multiple records in the RECONs keeping track of database and data set recovery information. The RECONs are also used to keep track of the sharing status of a database, including its *share level* and who (what IMSs) are currently *authorized* to use that database. Since all IMSs in the IMSplex use the same set of RECONs, access to the databases can be controlled by DBRC, the RECONs, and the share level and status of the databases. To be shared in a multi-system block level data sharing environment, the databases must be registered at share level three [SHARELVL(3)].

- Database buffer pools

  Each IMS has one or more buffer pools containing database blocks or CIs. The buffer header for each buffer contains a vector index into a *local cache vector* in the processor's hardware system area (HSA) which indicates the *validity* of the contents of the buffer. IMS will check this vector before referencing data in a buffer to determine whether it is still valid.

- Coupling Facilities

  The Coupling Facility, not to be confused with XCF (cross-system coupling facility software services), is hardware connected through Coupling Facility links to each processor in the Parallel Sysplex. Users, such as IMS and the IRLM, can use XES services to store data in the Coupling Facility in blocks of CF memory called structures. An IMS data sharing environment uses cache structures (IMS) and lock structures (IRLM). These structures can be accessed by other IMSs or IRLMs, allowing them to share information.

- Cache structures

  IMS online subsystems and batch jobs sharing databases connect to the same cache structures. Directory entries in the cache structures identify which IMSs have which data in which buffers. There is one directory entry for each block of data in any IMS buffer. These are used by cache services for buffer cross-invalidation. When one IMS updates a block or CI, it invokes XES cache services to cross-invalidate the buffer in any other IMS that has that data (block or CI) in its buffer pool, as determined by the directory entry. The total number of buffers in all IMSs should be used to determine the number of directory entries needed and the size of the cache structure.

  An IMSplex has:

  – One directory-only cache structure for VSAM; required even if not sharing VSAM databases

  – One directory-only or store-through cache structure for OSAM; required even if not sharing OSAM databases

  – One store-in cache structure for each shared DEDB VSO area; not required if there are no shared DEDB VSO areas. You may optionally choose to have two shared VSO structures per area with Fast Path maintaining duplexed copies. This is not the same as system-managed duplexing.

  The store-through and store-in cache structures also contain data entries containing blocks or CIs for fast access by connected IMSs. IMS can read data from a cache structure if it exists rather than having to read it from DASD.

- Local cache vectors (LCV)

  Each buffer in each IMS in each OS/390 image is associated with a bit in the local cache vector in the HSA. This bit is allocated during IMS initialization and is used to indicate whether the contents of that buffer are valid. If the block or CI has been updated by some other IMS since it was last read, XES buffer cross-invalidation services turns this bit on indicating that the contents of the buffer are invalid. IMS tests the LCV bit before each access to a buffer to make sure it is still valid.

- IRLM

  There can be a maximum of 32 IRLMs in the data sharing group. Only one per OS/390 image is required. Multiple IMSs can connect to a single IRLM on the same OS/390 image. The IRLMs accept lock requests from their client IMSs and invoke XES lock services to *grant* the lock, make the user *wait* for the lock, or *reject* the lock. Information about locks is kept in the IRLM lock structure. There should be one IRLM per OS/390 image where a data sharing IMS is running.

- Lock structure

Each IRLM in the data sharing group must connect to the same lock structure. The lock structure contains information about what locks have been requested by each IRLM, and which locks are protecting potential updates. There can be a maximum of 32 connectors to a lock structure.

- Fast Database Recovery (FDBR)

  Shown on the diagram as a dotted line box are *optional* address spaces called Fast Database Recovery regions. Each FDBR address space is associated with one IMS control region and monitors that IMS for failure - either an IMS abend or a system failure. If there is a failure, then FDBR performs some of the functions normally performed by emergency restart. It backs out in-flight (uncommitted) database updates and releases the locks for these updates held by the IRLM. For DEDBs, FDBR performs redo processing. This allows other IMSs to access the records being protected by the locks held by the failing IMS. It is still necessary to emergency restart IMS to recover message queues and to resolve indoubt units of work with DB2, CICS, WebSphere MQ, or any programs accessing IMS data using ODBA, but the dynamic backout has already been done by FDBR.

- XCF and XES services

  IMS data sharing is tightly integrated into the sysplex services which we have described in some detail in 2.3, “Sysplex services” on page 17. Since many of the above components are described there, we recommend that you review it closely before continuing, if you have not already done so. These services provide functionality, communications, and integrity, but may also impact performance.

Planning considerations: Be sure that you understand the components of an IMS data sharing environment. If necessary, arrange for education for all involved in the planning, implementation, and operation of the IMSplex.

3.2 Objectives and expectations

Sometimes, when implementing a new function or feature, or a new technology, you can lose sight of the reasons why you are doing it in the first place and get too caught up in the technology. The first step in the planning process should be to document your reasons for implementing the feature - block level data sharing (BLDS) in this case. What problems does it solve, or how does it make your application environment perform better or become more available. Then, throughout the planning process, *continually review your objectives* for implementing BLDS and what your expectations are. When the migration process is complete, come back one more time and see if it meets your expectations.

3.2.1 Objectives

Objectives for implementing any new technology generally fall into one or more of the following categories:

- Capacity
- Availability
- Performance
- Recoverability
- Operability
- Functionality

You should define, in your migration plan, what your objectives are - why you are implementing data sharing for your IMS databases.

- What are your *objectives* for implementing block level data sharing? Why are you implementing block level data sharing? For capacity? For availability?

  Availability and capacity are the two primary objectives users have for implementing data sharing. Block level data sharing should not be considered a performance objective *except in the case where performance suffers because of a lack of capacity.* It does it provide increased manageability, recoverability, or function relative to a non-data sharing environment.

  Planning considerations: Be sure you understand, and document, your objectives for implementing data sharing. Why you are doing it and how you will know when you are successful?

3.2.2 Expectations

Expectations are quite different from objectives. You may be implementing data sharing for availability, or capacity, but you should document your expectations for how (much) you expect availability or capacity to improve. You should also document your expectations relative to the other characteristics, which are not part of your objectives.

- How do you expect your IMS system *capacity* to change? How much do you need?

  This is one of the two major reasons for data sharing. When the workload becomes too high for a single processor to handle efficiently, then a second (or third or fourth) LPAR is added to the environment in order to distribute the workload. With more and more users opening their applications (and databases) to the general public through the internet, the potential is there for vastly increased transaction volumes.

- Are you expecting your data to be more *available* than currently? Why?

  While capacity may be a reason for data sharing, with the power and speed of today's processors, a more likely reason is the increased availability an application system gets by virtue of executing on multiple OS/390 LPAR images. If one fails, near full availability can be restored quickly by moving the workload from one system to another. Part of the planning process will be to determine how to distribute the workload across the data sharing IMSplex, and what to do in the event that something fails.

- Do you expect *performance* to decline when data sharing is implemented? Online? Batch? By how much? Is this acceptable? What if it is more?

  You should expect that, according to some measurements, applications which share data will not perform as well as those which do not share data. These measurements are typically internal. For example, you may measure processor consumption after data sharing and realize that more CPU is required in the data sharing environment. Or you may measure I/O response time and find that with the busier activity, I/O response time is slightly higher. You may even measure end-user response time and find that it went up several percent. However, most well designed application systems (databases included) continue to perform as well with data sharing as without when viewed from the perspective of the end-user. For example, a transaction whose execution time was 100 milliseconds without data sharing may require 110 milliseconds with data sharing. A 10% increase, but not noticeable by the end-user. If your reasons for going to data sharing is to relieve a capacity constraint, then that end-user may actually see improved response times due to spending less time in the input queue.

  Batch or BMP jobs, however, which run for long periods of time may display a measurable increase in elapsed time. This is especially true of a DLIBATCH job which has had exclusive use of its databases and is now forced to compete with the online for that data, or has been converted from DLIBATCH to BMP (with data sharing).

There is a general rule, however, that applies in most cases. Programs which run well in a non-data sharing environment will probably run well when sharing data. Those that do not run well will almost certainly not run well when sharing data. Part of the planning process will be to try to identify any characteristics of the applications or databases which would make them poor performers when migrated to a data sharing environment.

When you go from non-data sharing to data sharing, you will incur a measurable increase in overhead for the change from local to global locking. However, once you are data sharing and already paying for this global locking overhead, adding additional data sharing systems do not increase the overhead significantly - certainly not measurably.

- Do you expect database *recovery* to be easier, more difficult, or about the same?

  Recoverability is not an objective of data sharing, but you should have some expectations that your data is fully recoverable when there is a system or database failure. In fact, it is, but the recovery process for shared data differs from that of non-shared data, and the recovery process may (although not necessarily) take longer. In your data sharing plan, you should identify a task to document procedures for recovery from any failure.

  Although the workload (for example, the message queues) is important, and should be recovered if at all possible, generally the data is the most critical, so you will need to plan at least for recovering the databases if a disaster befalls the data center.

- Are you expecting your data sharing system to be more difficult to *manage or operate* than a non-data sharing environment? In what way?

  Again, systems manageability is not an objective of data sharing, but you should have some expectations regarding the ease or difficulty of managing and operating a data sharing environment. One of the most common issues with managing or operating a data sharing environment is the need to take databases offline for batch processing, or for periodic backup and reorganization. This will necessitate some changes in your operational procedures.

- Will data sharing provide any changes in *functionality*? Is there anything that you cannot do with data sharing that you could without it? Does it provide any new function or capabilities?

  The only functional benefit of data sharing is data sharing itself - the ability to share the data across multiple IMSs running on multiple LPARs. You should also expect, however, that there is no loss of functionality when databases are shared. Prior to IMS Version 6, for example, certain types of Fast Path databases could not be shared. This is no longer true except for main storage database (MSDB) which has been replaced in many shops with data entry databases (DEDBs) using the Virtual Storage Option (VSO), which can be shared.

  You do not have a lot of choices to make here, other than which databases are to be shared and for what level of sharing will they be registered. Not all databases in an IMSplex may need to be shared. It is perfectly valid to share some databases and not others. What you must do in this case is to make sure that the workload is delivered to the right IMS.

Planning considerations: Document your expectations in each of the areas identified above. Make sure that your sponsors have the same expectations. Verify that your objectives and expectations are consistent.

3.3 Functional planning

Data sharing is not a difficult function to implement. The implementation tasks are fairly straightforward with only a few decisions required, such as what databases do you want to share, at what level, and how many IMSs will be sharing the databases. *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929, describe these tasks. This section will address some of the planning issues, and offer some suggestions for reducing any negative impact, or identify where procedural changes must be made.

3.3.1 Data sharing function

As mentioned above, planning for what data sharing functions you will implement is not a difficult task. It will depend on what your objectives are, as all functional planning does. Since we have determined that *capacity and availability are the two primary objectives* of any data sharing environment, let's discuss these first.

Capacity

Implementing data sharing for capacity reasons is not as common as it once was. With the emergence of larger processors (the z900 2064-216 turbo 16-way can deliver over 3000 MIPS of processing power and have up to 64 gigabytes of real memory) it is usually *possible* for even the largest of applications to run on a single OS/390 or z/OS image. However, there are frequently other reasons why the full power of a single image is not available to the IMS application. The processor itself may not be configured as a single LPAR, or other work on that LPAR prevents IMS from having as much processing power as it needs. Perhaps there is sufficient processing power for most processing, but there are daily, weekly, monthly, or yearly peaks during which you need additional capacity. In this case, data sharing offers a way to utilize multiple OS/390 images on multiple LPARs to get the processing power needed. Once the initial conversion to data sharing is accomplished, it is relatively easy to add additional IMSs as needed for these periods of high demand, and then to remove them when the demand slackens.

From a capacity planning perspective, you need to plan on how many IMS images you will have and on how many LPARs they will execute. This requires knowing approximately what the processing requirements of your single image IMS currently requires. This should be the requirement at its peak, not its average, since the final configuration must be able to handle the peak load. You then need to add to that for growth (new applications, new workload, opening it to the internet, and so forth) and then add something for data sharing. How much to add for data sharing is really dependent on the applications themselves, but it is typically in the range of 10-15 percent.

Planning considerations: Determine the number of IMS images needed to provide the capacity required for your final configuration. Plan for multiple workloads - low, average, peak. You may want more than you need for availability reasons, but you should understand how many you *must* have.

Availability

Improved availability is often the overriding objective for implementing data sharing. Businesses today are more dependent on computers for their day-to-day business operations than ever before. It is rare that people can take over with a pencil and pad when the system is down. By having your IMS applications running on independent LPARs, a single failure would only affect a portion of the applications' availability.

Consider, however, that *unless there is an outage, a data sharing environment is no more available than non-data sharing*. Therefore, in order to benefit from the improved availability of data sharing, you must have solid procedures for what you will do when there is a failure. 1.2.3, "Define degraded mode environment" on page 8, identified part of the planning phase as deciding what you will do when something fails, and documenting a plan and procedures to handle each failure scenario. Without such a plan, you may still get some availability improvements by just continuing to run on whatever is left, but for maximum availability for critical applications, you should *be ready to move workload* to surviving IMSs when necessary, and perhaps even to discontinue some processing (IMS as well as other processing) to allow the critical work to complete while waiting for the failed system to become available again.

Planning considerations: Determine and document how you expect to achieve improvements in availability. What are you current availability problems and how will data sharing solve them. Identify and prioritize your critical business applications and define your degraded mode environment(s). Consider your capacity estimates when doing this.

What to share?

The above paragraphs discuss your objectives and what you should plan for - availability and capacity. However, you also need to plan for how you will configure your data sharing environment. While it is ideal to have each IMS a clone of the other(s), enabling any work to execute on any IMS image, technical or business requirements might dictate that this cannot be done. For example, if some applications require resources which cannot be shared, then those applications will have to be relegated to a single IMS. There is really only one IMS database resource that cannot be shared - the Fast Path main storage database (MSDB). However, IMS Version 6 provided an alternative to the MSDB that provides equivalent function with near equivalent performance - the data entry database (DEDB) with the virtual storage option (VSO). There may be other databases which you do not want to share, perhaps because of performance problems related to access patterns what would be particularly bad in a data sharing environment.

If there are some resources required by applications that cannot be shared, or that you do not want to share, you may find yourself with a *partitioned* application and data sharing environment rather than a *cloned* environment, or more likely, a *hybrid* environment where some applications can run on any IMS (cloned) and others must run on just one (partitioned).

Applications which can run on only one (or a subset of the total) IMS are said to have an *affinity* for that IMS. For example, you may have a few applications which require a database which cannot be shared. Those applications are said to have an affinity for the system where the database is available.

Planning considerations: Identify any database resources that cannot, or which will not be shared. Document how you intend to manage these unshared resources, including how you plan to route the work to the data.

3.3.2 Special application considerations

We have said that an application or database which runs well in a non-sharing environment usually runs well in data sharing, however, there may still be some characteristics which should be identified.

- PROCOPT=E

  Database PCBs with an exclusive processing option provide exclusive use of the database to the scheduled PSB instance when in a non-sharing environment. Since the database is used exclusively by a single program, locking is not required. PROCOPT=E may be used because the nature of the database and application requires that only one program should access the database at a time. Or, it may be done because only one program at a time needs the database and this would eliminate the locking overhead.

In a data sharing environment, PROCOPT=E *does not provide cross-system exclusive access.* It would be possible for two PSBs on two different IMSs, one of which used PROCOPT=E for the same database, to be scheduled concurrently. Therefore, if a database is being shared, even with PROCOPT=E, IMS will issue lock requests when accessing that database. This could cause several problems.

– If PROCOPT=E was being used to provide single program access, then this will not work in a data sharing environment. Other techniques must be used to guarantee exclusivity. The easiest way would be to limit the PSB(s) with PROCOPT=E for a database to a single IMS.

– Secondly, when IMS begins locking database resources, it may introduce locking problems that did not exist when locking was not done. For example, if the program were a BMP, and the BMP updated the exclusive database (no locks) but did not update other databases (short duration locks), then it might not take frequent (or any) checkpoints. In a locking environment, even though there should not be any lock contention (assuming you have limited the PSB to a single system), huge numbers of locks could build up and cause IRLM and lock structure storage problems. The best way to address this problem is to add checkpoints to the BMP. Another way would be to DBR the database on all IMS systems and then change the database ACCESS to exclusive for that IMS where the BMP is running by starting it with the following type of command:

```
/START DATABASE xxxx ACCESS=EX
```

This puts the database into non-sharing mode and locking would not be done. Our recommendation is to put checkpoints into the BMP.

Planning considerations: Identify all PSBs with PROCOPT=E PCBs and be sure you understand the reasons. If it really requires exclusive scheduling, then implement one of the techniques described.

- SERIAL transactions

  This problem occurs for the same reason the PROCOPT=E problem occurs - one IMS does not know what the other is doing. This is addressed again in the shared queues chapter, but even without shared queues, if a transaction must truly be serially scheduled within the IMSplex, then it must be limited to execution on a single IMS. There is no cross-system serial scheduling.

  One way to do this without shared queues would be to connect the data sharing systems with MSC and define the transaction as local in only one IMS and as remote in all others. This, of course, introduces an availability exposure, but no more than you had with just a single IMS before data sharing. However, you should make sure that these transaction really do need to be scheduled serially.

Planning considerations: Evaluate each transaction defined with SERIAL=YES to determine whether it really needs to be serially scheduled. If so, include in your implementation plans a technique to guarantee that these transactions can only be scheduled on a single IMS.

3.3.3 Dealing with the side effects

The more difficult part of planning for data sharing is identifying the *side effects* and how to manage or mitigate them. Examples of these side effects are:

- Performance degradation
- Database backup and recovery procedural differences

- IMSplex component failure recovery

Performance

Generally speaking, applications which perform well in a non-data sharing environment will perform well when moved to a data sharing environment. On the other hand, applications which do not perform well in a non-sharing environment will not improve with data sharing unless the reason for poor performance is capacity.

There are two *primary causes* of poor data sharing performance. They almost always go together. That is, if you have a problem with one of them you will probably have a problem with the other. Remember that we are talking about *differences* between the sharing and non-sharing environments. Poor non-sharing performers will have the same bad characteristics in data sharing.

- Block lock contention

  The main difference in locking between data sharing and non-data sharing environments is when a database record is updated. In a *non-sharing environment*, a record lock is acquired on the database record being updated, and this provides all the integrity needed. No other program can access that same record, but other programs can access other records in the same block. In the non-sharing case, there is only one copy of the block in one buffer and only in that one IMS (database authorization prevents other IMSs from accessing the same database). It is perfectly OK for multiple application programs on that IMS to share the same buffer.

  In a *data sharing environment,* IMS still gets a record lock on a segment which may be updated. Other programs on other IMSs can still access other records even in the same block. However, those other programs are not sharing the same buffer. When that other program is running in another IMS, there is another copy of the same block in that other IMS's buffers. If each IMS were to update its record, and write its version of the block back to DASD, then the update made by the last IMS to write the block would overlay the update made by the first. To prevent this from happening, before any IMS can update a segment, it must first acquire a block lock. This is a lock on the block or CI itself - not just on the database record. This block lock may result in another IMS trying to *update a different record in the same block* from getting the lock. That other IMS would have to wait for the update to be committed, the updated block or CI to be written to DASD, and the block lock to be released before it could continue.

- Buffer invalidation

  As mentioned above, these two side effects generally go hand-in-hand. In a non-sharing environment, the single IMS manages its own buffer pools and every write of blocks or CIs in those buffer pools to DASD is a result of an update made by that IMS.

  However, in a data sharing environment, there are multiple buffer pools in multiple IMSs and a single block or CI could exist simultaneously in each IMS. Block locking prevents multiple IMSs from updating different records in the same block so long as the block lock is held. However, when the block lock is released (see description of block lock contention above), the waiting IMS can continue. If it were to update its segment and write it out to DASD, then that updated block would overlay the first update. This problem is resolved by an XES service called *buffer invalidation*. After the write operation, but before releasing the block lock, the first IMS will issue an XES cross-invalidate call. If the block that it has updated is currently registered (according to the directory entry for that block in the cache structure) to any other IMS, that other IMS's buffer will be *invalidated* by setting its assigned bit in the local cache vector (LCV) to invalid.

  When that second IMS finally gets its block lock, before accessing the buffer to update it, it first checks this LCV to see if the buffer is still valid. When it finds an invalidated buffer, it must *reread the block or CI* from DASD (or in some cases, from a Coupling Facility cache structure) to get the updated version. This invalidation of buffers is never needed in a non-sharing environment. See 2.6.3, “Cache structures and cache services” on page 23 for a more detailed description of how buffer invalidation works.

For OSAM databases, you have the option of caching locks in OSAM buffer pools in a store-through cache structure instead of maintaining a directory-only cache structure as for VSAM. When you do this, IMS will write updated blocks to the structure after writing them to DASD. If that block had been in another IMS’s buffer which was invalidated, the other IMS could reread the block from the cache structure at Coupling Facility speeds instead of DASD speeds. Since this is available only for OSAM, you may want to consider changing any VSAM ESDS database data sets to OSAM [ACCESS=(HDAM,OSAM) on the DBD].

Some examples of where block lock contention and buffer invalidation can cause problems in data sharing are:

- Small heavily updated databases

  Small heavily updated databases tend to be a problem even in non-sharing environments. It just gets worse when that database is shared. The probability that two (or more) IMSs will try to update different records in the same block is higher than if the database were large.

  This, and the next few problem areas, are often difficult problems to address, especially for existing applications and databases. The best approach, short of redesigning the database and rewriting the application, is to make sure that each record is in its own block. This may mean making the database a lot bigger than it needs to be just to hold the data, but it will tend to *spread the data out* and reduce the chances of two records being in the same block. You may also consider making the *blocks or CIs smaller.* This naturally tends to put fewer records in the same block.

- Database hot spots

  Even though a database may be large, if update activity is concentrated in a small part of that database, block locks could cause contention.

  This is even more difficult to address. It may not be practical to just make the database a lot bigger if it is already large. It may be practical to *reduce the size of the block or CI*.

- HIDAM or PHIDAM databases with ascending keys and applications which insert at the end of the database

  Again, even if the database is large, inserts to the end of the database become a *very hot* spot. An example of this is a HIDAM database where the key includes date and time in the high order position. Every new record would go at the end of the data set - that is, in the same block until it is full then the next (new) block will be allocated and used. This process would be serialized across the IMSs.

  This is an *application and database design issue,* and would be difficult to address with the simple techniques described above. But this database probably already has performance problems.

- HIDAM or PHIDAM databases with no free space

  This is similar to the above problem, except that ANY insert, when there is little or no free space, will be put in the last block or CI of the data set. Since this would be the same block for all sharing IMSs, there will be block lock contention for that overflow block.

  The solution to this problem is to make sure that your database has plenty of free space for inserts.

- HDAM or PHDAM databases without enough free space

This is similar to the above problem, except with (P)HDAM, when data will not fit in the most desirable block, or another block in the root addressable area, the space search algorithm will send it to the current overflow block (at the end of overflow). Since this would be the same block for all sharing IMSs, there will be block lock contention for that overflow block.

This one is easy to fix. Make the database root addressable area larger. Avoid overflow.

- Databases with secondary data set groups

  Databases defined with secondary data set groups can be a problem when there is insert activity to that secondary data set group. Even when the database is HDAM, space is allocated in the secondary data set group the same way it is allocated in HDAM overflow. The first block or CI is filled, then the second, and so on. This, of course, leads to the same block lock contention and buffer invalidation problem.

  The solution to this is to eliminate secondary data set groups which have high insert activity for both (P)HDAM and (P)HIDAM.

- Frequent updates of the HD bit maps

  The bit maps in a (P)HD database data set identify blocks or CIs where there is enough free space to hold the largest segment. When a database has too little free space, it is more likely that the free space becomes exhausted requiring the bit map to be updated. Does this have a familiar ring to it? Same problem with lock contention and buffer invalidation.

  The solution, of course, is to have plenty of free space in your databases.

The common thread in each of the above is that the same block is being frequently updated by multiple IMSs. This causes no problems in non-sharing environments (it might even be a benefit) but it can be a severe problem when data sharing. For some of the above, if the suggested techniques do not work, you may have to declare the database unshareable and route all applications needing that database to the one IMS system where it is available. That is, *partition* the applications which need these databases. This reduces some of the availability benefits of data sharing, and it may not be practical if the database is used by a large number of application programs.

For the free space issue, if your databases are already quite large, and adding more free space will give you data sets larger than allowed, or larger than you want to support, you might consider converting HDAM and HIDAM databases to the HALDB PHDAM and PHIDAM databases (only available beginning with IMS Version 7). HALDB is a partitioned database allowing you to define up to 1001 partitions for your data, allowing each data set to be smaller, *with plenty of free space.*

Planning considerations: Analyze your current applications and databases to identify which have processing characteristics that would result in particularly poor performance. In particular, look for the common causes described above. If you do find some potential problems areas, try to get these fixed before you go to data sharing - it might take a while so get started early.

Processing overhead

In addition to the potential problems caused by database design and application program access patterns, there is also some *additional overhead* just in processing shared databases. This overhead is the result of several requirements for data sharing that do not exist when data is not shared.

- IRLM versus program isolation

In a non-data sharing environment, you have a choice between using IMS's program isolation (PI) locking or the IRLM. IRLM locking has longer path length and uses more storage than PI locking, so most users tend to use program isolation when not sharing data. You do not have the choice with data sharing - you must use the IRLM.

- Lock requests requiring lock structure access

  Many IMS calls require a lock to be acquired before accessing the record or updating a block or CI. In a global data sharing environment (IRLM SCOPE=GLOBAL), each lock request requires one or more accesses to the lock structure in the Coupling Facility. If the lock structure is too small, then you may encounter false contentions which require inter-IRLM communications to resolve a potential lock conflict, even if no real conflict exists. Additionally, there are more lock requests. Not only are there record lock requests, which we have even in non-sharing environment, but there is also the block lock request for updates.

  False contentions can be identified from RMF reports on Coupling Facility structure activity, and can be resolved by simply making the lock structure larger.

- Registration in cache structure

  Before IMS reads any block of data into a buffer, it first registers that block in the Coupling Facility cache structure using XES cache services. This either adds a directory entry to the structure, or adds IMS as a registered user of that block in an existing directory entry.

- Cross-invalidation of buffers

  When IMS updates a block, after writing it to DASD, it issues a cross-invalidate call using XES cache services to invalidate that block in other IMSs' buffers. Even if the block does not exist in other IMSs, IMS does not know this and still issues the call.

How much overhead each of the above causes depends on the application and database design. Not all calls require locks, not all calls require I/O, and not all calls result in cross-invalidation. An application which issues lots of GNP calls to databases which have lots of dependent segments do not require as many locks as applications which issue lots GU calls to root segments. Applications which do not do a lot of read I/O do not have to register as often with the cache structure. Applications which do not update much do not do as many cross-invalidate calls. All of this activity is measured in terms of additional CPU utilization. Even the requests to the Coupling Facility are synchronous requests, meaning that the TCB (one processor engine) is in a busy state.

But it is reasonable to assume that the sum of all this overhead would require an additional 10-15% CPU than an equivalent workload running on a single IMS without any data sharing overhead. This type of overhead seldom leads to any perceived increase in response time or elapsed time since it does not involve I/O - the major component of all transactions and batch or BMP applications.

Planning considerations: This is really an *expectations* issue. Develop a good sense of your current environment and how much resource is required at least for your critical or high volume applications. Then establish procedures to monitor their resource utilization after data sharing implementation. Document how to recognize and report unusual or unexpected performance degradation.

Database backup and recovery

In a non-shared environment, databases are allocated to, and updated by, only one IMS at a time. There is a single log stream (although perhaps created by different IMSs at different times) which the recovery procedures use to recover the databases. Database recovery requires image copies from which to begin the recovery, and a log stream which has any updates since the image copy was taken. This log stream can be either the logs themselves, a change accumulation of the logs, or a combination of logs and change accumulations.

Because shared databases are *allocated to and updated concurrently by multiple IMSs,* database backup and recovery procedures will be different. There are several considerations.

Image copies

There are two types of image copies you can take with IMS databases (three if you count “user image copies” which are not recorded in the RECONs) which can be used, with recovery utilities or IMS Online Recovery Service for z/OS (ORS) product and log data, to recover databases.

- Clean (non-concurrent) image copies

  A clean image copy requires the database to *not be allocated for update to any IMS* while the image copy is being taken. For batch IMS, you simply have to wait until the batch job terminates. For online systems, this generally means issuing the /DBR command. In a data sharing environment, the /DBR command must be entered from every online IMS where the database is allocated.

  Alternatively, a /DBR DB xxxx GLOBAL command can be issued from one IMS, which notifies other IMSs in the data sharing group to DBR the database locally. The GLOBAL option will turn on the “prohibit further authorization” flag in the RECONs for that database, which would prevent even image copy jobs from getting authorization. When issuing this command for purposes of offline image copy, you should include the NOPFA parameter on the command as follows:

  ```
  /DBR DB xxxxx GLOBAL NOPFA
  ```

  This command does NOT turn on this flag and allows any requestor, including image copy and other IMSs, to get authorization.

  This command (/DBR) establishes a *recovery point* from which, or to which, a database can be recovered using standard IMS utilities or tools. The recovery point is a point of consistency at which *DBRC knows* there are no uncommitted updates on the database, and therefore, none in the image copy which might need to be backed out. Most users take clean image copies on a regular basis either to speed up any required recovery process, or to establish a recovery point at some specific time to which the database can be recovered.

- Fuzzy (concurrent) image copies

  Fuzzy image copies, also called concurrent image copies, do not require a recovery point. The database can be allocated for update to multiple IMSs during the image copy. DBRC also tracks these image copies and knows what logs are required to perform a recovery from the time of the fuzzy image copy. The obvious benefit is that the database does not have to be taken offline. The disadvantage is that it does not establish a recovery point to which the database can be recovered using standard IMS utilities. It also does not allow the use, by the recovery utility, of change accumulation data sets. Many users also take fuzzy image copies between clean image copies to speed up recovery without having to take the database offline to establish a recovery point.

Planning considerations: Review your current image copy procedures. You may discover that fuzzy image copies can be substituted for clean image copies (at least partially) and improve database availability. By using a tool such as IMS Online Recovery Service for z/OS (ORS), you no longer have to DBR the database for a clean image copy to establish a recovery point. When you do need a clean image copy, the IMS Image Copy 2 utility, along with supporting hardware, can reduce the duration of an outage.

Change accumulation

The IMS Change Accumulation utility uses prior change accumulation data sets and subsequent logs from any number of IMSs to create a data set with only the most current updates. For example, if a segment was changed 50 times since the last image copy, only the final version of the segment would be recorded in the change accumulation data set. The utility uses timestamps in the log records to accomplish this, but does not consider whether a logged updated is committed. If that 50th update was uncommitted, it would have to be backed out using the OLDS (if still available) or the SLDS after recovery completes.

In a data sharing environment, because the recovery utilities cannot process multiple log streams as input, a change accumulation of all the logs from all the sharing IMSs is required for recovery. If a database problem occurs requiring full recovery, the database would have to be DBRed from each IMS, and each batch job terminated, and change accumulation run one final time to capture the latest updates up to the time the database was deallocated by each IMS.

Planning considerations: Since database recovery cannot be done in a data sharing environment without first running change accumulation with all the logs, you should examine your current policies for change accumulation and consider altering them to be more prepared for a database recovery by running change accumulation more often. Note that with ORS, a change accumulation is not required for recovery, but it will improve its performance.

Database recovery

Each IMS (batch and online) which updates a shared database will generate its own set of logs recording those updates. These logs will probably overlap. When this is the case, the recovery process (for example, the database recovery utility) cannot use the logs directly. They must first go through the change accumulation process which merges all the updates and produces an output file that has only the latest updates from all the sharing IMSs. In this case, uncommitted updates may also be “recovered” since they were on the change accumulation data set. They would have to be backed out either by online dynamic backout, and/or by running the Batch Backout Utility against each active PSB (online or batch).

It is usually faster to recover databases, even in non-sharing environments, if you first run change accumulation. In data sharing, it is a requirement and may require a change in your recovery procedures.

Planning considerations: Review your database recovery procedures and make changes are required.

Database recovery with ORS

IMS Online Recovery Service for z/OS (ORS) product allows you to do database recoveries quickly, without the need for change accumulation data sets, even in a data sharing environment. The tool reads multiple log streams concurrently and applies updates to an image copy starting point in commit time sequence. If multiple database data sets are being recovered, ORS will recover them in parallel, including reading the image copies, any change accumulation files you have, and the logs. Information about ORS can be found on the web at IBM DB2 and IMS Tools site:

`http://www-3.ibm.com/software/data/db2imstools/imstools/`

Planning considerations: If you do not already have ORS, visit the Web site to determine whether it can help in your environment.

Data sharing component failure

First go back and take another look at Figure 3-1 on page 57, which identifies the major components of a data sharing environment. This section addresses the failure of those components.

There are two issues dealing with the failure of a data sharing IMS. One is the *recovery (or restart) of the IMS* itself. The other is the *redirection of the workload* to an IMSplex member that is still available. The following identifies what the impact of that failure is on the end-user, how one might recover from that failure, and if the work can be redirected, how to do it.

Recovery of a failing IMS

When IMS fails, it is usually best to restart it as quickly as possible. There are several very important reasons.

- The obvious reason for restarting IMS is to *restore full service* to the end-users. Depending on your end-user procedures, some users may be able to log on to another IMS and continue doing some work, however, in general, any surviving IMSs will not be able to handle the full workload.
- Another is if the failed IMS is in session with CICS, DB2, or WebSphere MQ, an emergency restart is necessary to *resolve any indoubt units of work* between the subsystems.

But in a data sharing environment, there are others reasons to restart IMS as quickly as possible.

- One reason is to *release any locks* that are held by the IRLM on behalf of the failed IMS. Any user on another still active IMS who requests this lock will get a lock reject condition and abend with a U3303.
- A second is to *release the database authorizations* still held by the failed IMS. No batch job (or utility) can get authorization to a database if it is authorized to a failed subsystem, even with data sharing enabled.

Planning considerations: Operational procedures should include rapid restart of a failed IMS subsystem.

Fast Database Recovery (FDBR)

Fast Database Recovery (FDBR) addresses only one of these data sharing side effects - releasing retained locks. FDBR will be notified by XCF that the IMS it is tracking has failed and will begin a recovery process which dynamically backs out inflight units of work and releases the locks protecting these updates. But FDBR then terminates, leaving the other three to be resolved by emergency restart. The databases remained authorized to the failed IMS subsystem.

Planning considerations: Identify requirements for FDBR regions to track each active IMS. Determine which LPAR each FDBR will run on (it should be on a different LPAR than the IMS which it is tracking). It is OK to have both FDBR and ARM since FDBR will complete its work before ARM can restart IMS.

Automatic restart management

The IMS control region, the IRLM, and many of the other IMSplex address spaces, support Automatic Restart Manager (ARM) unless you specifically disable it by starting with ARM=N in DFSPBxxx (for IMS) or in the JCL or other PROCLIB member for other address spaces. When ARM is enabled, if the address space abends, ARM will automatically restart it on the same OS/390 image. For IMS, an ERE command is issued internally, meaning that not only is

IMS reinitialized, but emergency restart is performed. If the LPAR on which IMS is running has failed, ARM services will automatically restart on another eligible LPAR in the Parallel Sysplex as determined by the ARM policy. This is called a cross-system restart. When this happens, an open SUBSYS record will still exist in the RECONs. In this case, ARM restart will also issue the OVERRIDE parameter on the ERE which lets IMS do emergency restart even though the RECONs did not recognize it as having abended.

Planning considerations: ARM should be active for all IMSplex address spaces which support it. If you do not currently use it, you should learn about it and seriously consider activating it. Review, or create, an ARM policy for all of the IMSplex components which can register with ARM. Identify, and define, ARM restart groups as necessary to keep IMS and CICS or DB2 systems together.

IMS batch jobs

An IMS batch job can participate in data sharing with other batch jobs, with online systems, or with both. Each batch job counts as one of the total of 255 sharing systems. If running on the same LPAR as an online system, it should use the same IRLM. It will have DBRC imbedded within the address space and, when it registers with DBRC, a SUBSYS record will be created in the RECONs. DLIBATCH allows you to request DASD logging. This should always be used for data sharing batch. If the job abends with an IMS pseudo-abend, IMS can use the log records on DASD to dynamically back out any uncommitted database changes, release IRLM locks, release database authorizations, and generally clean up the RECONs before terminating.

Without DASD logging, or in the case of a system abend or non-pseudo abend, none of this will happen. The databases will not be backed out, the locks will be retained, and the databases will remain authorized to the batch job (in the SUBSYS record). Batch backout will be required to do these functions which will be delayed much longer than any dynamic backout would require.

Planning considerations: If you are sharing data with your batch applications, you should consider using DASD logging to enable dynamic backout. This will require JCL changes and procedures for archiving the batch SLDSs to tape.

IRLM

When an IRLM fails, its data sharing clients can no longer request (or release) locks until that IRLM is restarted. Any locks that were protecting updates, or potential updates, will have been recorded in a record list entry in the lock structure. Other IRLMs in the data sharing group will be notified about the failed IRLM by XCF and will read the failed IRLM's record list from the lock structure into a data space. These are referred to as retained locks. Any *lock request for a retained lock will be rejected* and the requesting application program will abend with a U3303, including BMPs and batch data sharing jobs.

Currently executing IMS online transactions and BMPs, as well as batch jobs with DASD logging, will abend (U3303) and be dynamically backed out (we do not need the IRLM to do this). Online transactions will be requeued (suspend queue). BMPs and batch jobs will have to be restarted from the most recent extended checkpoint. If a batch job is not backed out dynamically, then batch backout will have to be run.

The online IMS will issue a message indicating that it is quiescing and stop all activity. Note that this includes not only shared databases, but unshared databases as well since the IRLM is managing those looks too.

```
DFS2011I IRLM FAILURE - IMS QUIESCING
```

When this happens, all full function databases and Fast Path areas are closed, deallocated, and unauthorized. Fast Path DEDB areas are stopped. When IMS has completed its quiesce processing, it will attempt to reconnect to the IRLM. If it has been successfully restarted before IMS attempts to reconnect, then the reconnect will be successful. This will usually be the case when ARM restarts the IRLM. If the automatic reconnect attempt by IMS fails because the IRLM is not yet available, then when it becomes available, IMS reconnect must be initiated using the following command:

```
F imsprocname,RECONNECT
DFS626I - IRLM RECONNECT COMMAND SUCCESSFUL
```

For full function databases, after IMS reconnects to the restarted IRLM, processing can begin again. Full function databases will be available since they were not stopped, but this is not true for the stopped DEDB areas which must be restarted by issuing the /START AREA command for each area.

Planning considerations: There are at least three considerations here.

- IRLM restart (should be done by ARM)
- IMS reconnect to the IRLM (if not done automatically)
- Restart of the stopped DEDB areas

Lock structure

When a lock structure fails, the IRLMs are notified by XCF and immediately begin the rebuild process. Each IRLM knows all of the locks held by its client IMSs and can use this information to completely rebuild the structure on another Coupling Facility. This, of course, requires that the lock structure be defined in the CFRM policy with multiple CFs in the preference list (PREFLIST). All updating IMS batch data sharing jobs will abend. They may be dynamically backed out if DASD logging was in use, or by using the IMS Batch Backout Utility. Online systems continue processing. Applications requesting locks will just wait for the lock structure to be rebuilt and for lock services to resume.

When connectivity from an IRLM to a lock structure fails, assuming that the REBUILDPERCENT in the CFRM policy allows it, and that you have identified more than one CF in the PREFLIST, the structure will be rebuilt on another candidate Coupling Facility after which “disconnected” IRLMs continue processing. When this happens. there are no abends, even for batch jobs. Their lock requests simply wait for the rebuild to complete.

Planning considerations: The IRLMs can rebuild a lock structure only if there is an available CF to rebuild it on. Be sure that when you define your lock structure in the CFRM policy, you define at least two CFs in the PREFLIST.

OSAM and VSAM cache structures

When an OSAM or VSAM cache structure fails, IMS temporarily stops data sharing while it rebuilds the structures. There is no loss of data since only OSAM can have data in a cache structure, and the OSAM data has already be written to DASD. What gets lost is the registration information about which blocks or CIs were in which IMS buffer pools. Since there is really no efficient way to recover this information, every unmodified buffer in every IMS is invalidated. If anyone needs the contents of the buffer, then IMS will reread it from DASD. But we cannot just discard modified buffers. However, because of the block lock, a modified block or CI can only exist in one IMS at a time. Therefore, for modified buffers, after the cache structures have been rebuilt, IMS will register interest again.

The point here is that there are no user requirements for addressing the loss of an OSAM or VSAM cache structure. You will get messages indicating their loss, and that data sharing has stopped, but it will eventually, and quickly, fix itself as long as there is a good Coupling Facility available to rebuild the structures. Just be sure that you include at least two CFs in the PREFLIST of the CFRM policy defining these structures.

Planning considerations: Like the lock structure, IMS can only rebuild these structures if there is an available CF on which to rebuild them. Make sure your OSAM and VSAM cache structure definitions have multiple CFs in their CFRM policy definitions.

Shared VSO cache structures

If a shared VSO structure fails, and if you have only defined one, then you have lost committed data that must be recovered. The only way to recover it is to run database recovery, which can be a long, tedious, and not very "availability friendly" process.

Planning considerations: Although CF and structure failures are rare, the impact of a failed shared VSO structure is high, requiring a full database recovery. Your plans should include either Fast Path managed duplexing of the VSO structures, or, when available, system managed duplexing of these structures.

Redirection of the workload

Although recovery of the failed IMS is important, the fastest way to restore availability may be to redirect the workload to an IMS which has not failed. This is especially true if it looks like recovery may take a long time. The workload may be online transactions and/or BMPs. In this case, it is important that you have predefined your *degraded mode environment* and identified the critical workload that must continue.

Online transactions

In a data sharing environment without shared queues, each online user is logged on to one specific IMS and that IMS must process all of that user's transactions. When that IMS fails, you are no longer able to continue working. You have two choices.

- Wait until IMS is restarted, log back on, and continue working. Without shared queues and the sysplex terminal management (STM) function of IMS Version 8 running with the Common Service Layer (CSL), this may be your only choice in order to keep some significant status, such as continuing an IMS conversation. If you were to log on to another IMS, even though the application and data is available on that other IMS, information necessary to continue the conversation would not be available. If you logged on to the original IMS later, that IMS would reestablish your conversation as it was at the time of failure (could be very confusing).
- Log on immediately to another IMS. This restores service to the user most quickly, but without shared queues and STM, any significant status that you had at the time of the failure would be lost. Note that if you are using the Rapid Network Reconnect (RNR) function made available with IMS Version 7, a user remains logged on to IMS even after it fails. Logging on to another IMS may not be an option. For this reason, RNR is not recommended for environments where CSL and STM are active.

From a planning perspective, your users must know how to react when IMS fails. One must know how long the failed IMS is likely to be down, and be able to evaluate the possible loss of significant status versus the need to get back into service quickly. Each user may make these decisions based on different criteria, and you may want some users to log on immediately to another IMS while others should wait. This goes back to the identification of workload priorities.

We have mentioned shared queues, sysplex terminal management, and Rapid Network Reconnect. Planning for these will be addressed more fully in “Shared queues planning considerations” on page 77, “Planning for the Common Service Layer” on page 105, and “Planning for IMSplex connectivity” on page 131.

Planning considerations: When an end-user is disconnected from one IMS, he must consider whether to wait for that IMS to be restarted or to log on to another IMS in the same data sharing group. This may depend on several factors, including whether all the data is available on all the IMSs, and whether the user has some significant status, such as a conversation, which would be lost if one logged on to another IMS. Whatever that user’s decision, you must develop policies and procedures for end-users to use when their sessions with IMS are terminated because of any IMS or session failure.

BMPs

When IMS fails, so does its BMPs. There are several issues to consider in your planning here:

- One is that any inflight work (uncommitted updates) of the BMPs must be backed out before the BMP can be restarted on ANY IMS. This can be done either by emergency restarting the failed IMS, or by having a Fast Database Recovery (FDBR) region tracking that IMS. When IMS fails, FDBR will dynamically back out all inflight units of work (UOWs) that were executing on the failed IMS and release the IRLM locks held on behalf of the failed IMS (this is always good, even if you do not intend to restart any work on another IMS).
- The second issue is that, to restart the BMP, the extended checkpoint log records must be available. If the BMP is restarted on another IMS, that IMS will not have the BMP checkpoint ID table (used for CHKPTID=LAST restart) nor the extended checkpoint log records (x’18’) anywhere on its OLDS or SLDS. In this case, if it is really important to restart that BMP before the failed IMS is restarted, you must provide the restart checkpoint ID on the BMP restart JCL, and provide the x’18’ log records through the //IMSLOGR data set. You must also be sure the BMP connects to the correct IMS. This can a be done either by specifying the IMSID on the restart JCL, or by using the IMSGROUP parameter in each online IMS, and have each BMP specify this IMSGROUP name as its IMSID.

  As with the transaction workload, it is important that for each BMP, you know how important it is to redirect the work to a surviving IMS.

  Planning considerations: Determine whether you will restart your BMPs on the same IMS after a failure, or on any available IMS. If any available IMS, then you need procedures for making the BMP’s extended checkpoint records available for restart.

3.4 Configuration, security, implementation, and operations

Because these planning activities are so often closely related to the same activities for other IMSplex planning activities, they are discussed in their own chapter, Chapter 7., “Putting it all together” on page 159.

3.5 Data sharing planning summary

This chapter addressed many of the issues that you will have to address when developing a migration plan for data sharing. To summarize:

- Understand the functionality, the components, the benefits, and the potential problems of a data sharing environment.

- Understand and document your objectives for data sharing. What problem(s) are you trying to solve?
- Document your expectations. How will it meet your objectives?
- Document what data sharing function you will use. How many images will you need to meet your capacity requirements? Your availability requirements? What databases will be shared?
- Understand the side effects of data sharing. Examine your current applications and databases to find any potentially poor performers. Identify procedures that must be changed. Any databases that cannot or should not be shared? Include in your plan, tasks to address these problems.
- Develop a target configuration that meets your objectives. Refine the target configuration throughout the planning phase. Include the platform requirements for OS/390, the Coupling Facility, and structures.
- Identify connectivity requirements. How will users get connected to the IMSplex? Where will BMPs be scheduled?
- Review and update your failure recovery procedures. Be ready to implement degraded mode processing. What will users do if their IMS fails or just isn't available? How and where will BMPs be restarted?
- Review your current database support processes (backup, recovery, reorganization) and update them as required.

4

Shared queues planning considerations

IMS shared queues was introduced in IMS Version 6 as a means of distributing the online transaction workload across multiple IMSs sharing data in an IMSplex. Like the data sharing enhancements in IMS/ESA Version 5, shared queues support relies heavily on the Parallel Sysplex services of XCF and XES to create a single set of IMS input and output message queues, including both Fast Path and full function messages, and then be able to access those queues from any IMS in the IMSplex capable of processing that message.

IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology, SG24-6908, contains a detailed review of IMS shared queues support. This volume and this chapter identify some of the planning tasks necessary, either formally or informally, to implement shared queues in your IMSplex environment.

This chapter addresses the first two activities in the planning process for shared queues:

- Objectives and expectations
- Functional planning

The next two activities are addressed in Chapter 7, “Putting it all together” on page 159:

- Configuration planning
- Security planning

Shared queues implementation and operations are addressed briefly in this chapter, but are covered more fully in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929.

Like all of the other planning chapters, the objective here is to ask questions or raise issues which you must address when developing your plans. Each question or issue is followed by some discussion of what you should consider when developing your migration plan.

4.1 IMS shared queues

Data sharing allowed application programs running on multiple IMSs to share the same data. There were several reasons for this, including providing the computing power of multiple processors to a workload, and providing improved availability by not putting all your eggs in one basket. For this to work, of course, the workload had to be distributable to IMSs across the IMSplex.

For BMPs, and batch IMS, "distributing" the workload was fairly straightforward - just let JES schedule it wherever it wanted, or direct it to a specific OS/390 (MVS) platform any number of ways. However, for transaction-driven applications, it was left to the user to determine how to distribute the transaction workload. IMS has its own set of *private message queues,* based on which IMS the user is logged on to. Once a message is queued to one IMS, it is no longer available to another, even if the first one fails, or if the first one is too busy to process it in a timely manner.

Several techniques were available, some of them not particularly user friendly:

- Tell each end-user which IMS to log on to. This requires the user to know the specific APPLIDs of the IMSs in the IMSplex and to know which ones are available in case their assigned IMS is not available. Once logged on to a particular IMS, all work must be done by that IMS.
- Use MSC to send some transactions to other "remote" IMSs within the same data sharing group based on the transaction code. This is just a form of partitioning the IMSplex and does not provide much benefit for availability. Each IMS can process only those transactions defined as local.
- Use VTAM Generic Resources (VGR) to allow the user to log on to a generic IMS APPLID and then let VGR distribute the workload. This shields the users from having to know the real IMS APPLID or which ones are currently available, but once a user is logged on to a particular IMS, all work must be done by that IMS.
- Use the IMS Workload Router product to use MSC to "spread the workload" around the data sharing group using MSC and a user-specified algorithm. This technique works fairly well at balancing the workload but it does, however, add the overhead of MSC to any "distributed" transactions.
- Use some external (to IMS) process, such as a session manager, to distribute the logons across multiple IMSs. Like the others, this process makes a decision about which IMS to establish a connection with and then sends the work to that IMS regardless of how busy that IMS may be. Some of these techniques may do some form of load balancing.

The list could go on with other non-IMS products which can be used to distribute the connections, but each one employs a "push-down" technique of workload distribution - send the work to one particular IMS regardless of how busy that IMS is. Chapter 6. "Planning for IMSplex connectivity" on page 131 addresses some of these techniques in more detail. One of the benefits of shared queues is that it uses a "pull-down" technique of workload distribution. The messages wait on a queue until an IMS is ready to schedule it. Whichever one is available first gets the work. If one IMS is too busy, including the one where the user was logged on and entered the transaction, then another can process the transaction.

Sysplex terminal management, a capability provided with the Common Service Layer in IMS Version 8, requires shared queues and uses them to be able to shift a user's status from one IMS to another when IMS fails. Sysplex terminal management is described in *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908, and in Chapter 5. "Planning for the Common Service Layer" on page 105 of this book.

4.1.1 Shared queues components

In order to plan for, and later to implement and operate, an IMS shared queues environment, it is first necessary to understand the components. The shared queues functionality takes advantage of many of the sysplex services offered in the Parallel Sysplex, including both XCF and XES services.

By now we know that IMS data sharing uses XES cache services with OSAM, VSAM, and shared DEDB VSO cache structures, and the IRLM uses XES lock services with a lock structure. Shared queues also uses sysplex services for the shared queues themselves, and for logging updates to the shared queues. Both of these use *XES list services with list structures.* Figure 4-1 shows the components of a shared queues environment with two IMSs sharing the same message queues. As a reminder, following this figure is a brief description of the components, but for a more detailed description, refer to *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908.

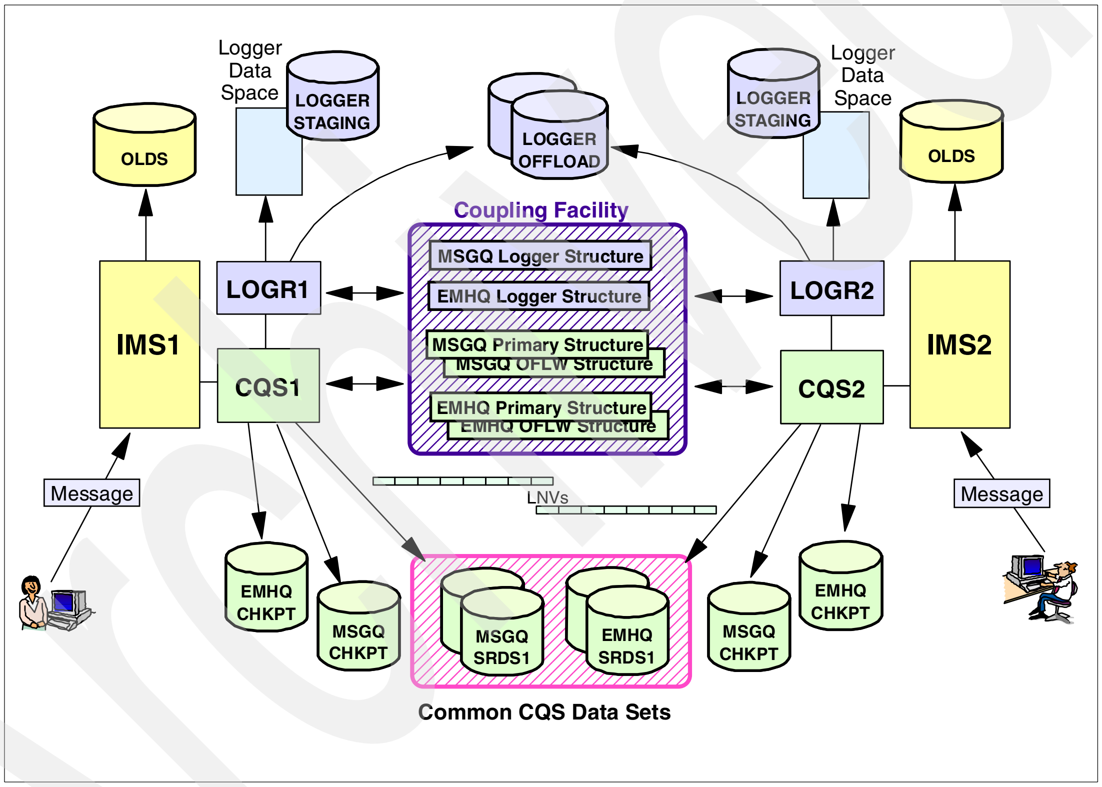


Figure 4-1 Components of an IMS shared queues environment

The components we need to understand are:

IMS control region

End-users still log on to a single IMS in a shared queues environment. Without shared queues, when a message arrives from an end-user, IMS determines the destination of that message and queues it for processing. Valid destination queues include transaction, logical terminal, and remote destination (MSC) queues. The messages are placed in the QPOOL and queued off a queue header representing that destination. For transaction destinations, the queue header is the scheduler message block (SMB) control block. For messages destined to a logical terminal, it is the communication name table (CNT). For messages destined for a remote IMS across an MSC link, it is the RCNT or RSMB.

In a shared queues environment, list entries in a list structure in the Coupling Facility replace the QPOOL for storing the message, and list headers in the list structure replace the IMS control blocks for queue headers. There are list headers for transaction codes, for LTERMs, and for remote (MSC) destinations. There are also special purpose list headers which we will not describe here. The process for shared queues is very similar to that of a non-shared queues IMS. The biggest differences are that the message queues are on a Coupling Facility where they are available to all IMSs in the shared queues group, and an IMS cold start does not delete the message queues.

IMS supports two types of messages - those we call full function messages, and those we call Fast Path expedited message handler (EMH) messages. Fast Path messages are stored in a different set of buffers, and queued off different queue header control blocks. In a shared queues environment, these message types are also treated separately with separate structures.

IMS messages

Messages can be received by IMS from multiple sources, including a VTAM network, a TCP/IP network, or from an application program running in IMS and inserting the message to an alternate TP-PCB. When IMS receives a message from the network, it performs a process called find destination (FINDDEST), which examines the beginning of the message for the destination name. FINDDEST is also invoked when an application program issues a CHNG or ISRT call to place a message on the message queue.

FINDDEST can produce several results:

- Command (first character is a slash '/')
- Statically defined transaction (TRANSACT macro in system definition)
- Statically defined logical terminal (NAME macro in system definition)
- Dynamically defined logical terminal (created by extended terminal option (ETO))
- Unknown or invalid destination (none of the above)

Handling of messages by IMS depends on the message destination type, and in some cases on the message source. Shared queues affects the way IMS handles these messages, including those with an unknown destination, in ways other than just putting them on the shared queues. We will discuss the handling of these messages in 4.3 “Functional planning” on page 86.

Common Queue Server (CQS)

IMS does not directly connect to the shared queues structures. It uses the services of the Common Queue Server (CQS) - a new address space required for shared queues. In addition to managing the messages on the shared queues, CQS also monitors the queues and notifies IMS when a message of interest arrives. For example, if IMS1 were interested in TRXN1 (meaning it was defined to IMS), it has to know when it, or another IMS, has placed a TRXN1 message on the shared queues. CQS, with the help of XES list services, performs this notification.

Coupling Facility

The Coupling Facility, not to be confused with XCF (cross-system coupling facility software services), is hardware connected through Coupling Facility links to each processor in the Parallel Sysplex. Users, such as the Common Queue Server, can use XES services to store data in the Coupling Facility in blocks of CF memory called structures. In the IMS shared queues environment, CQS uses list structures to store input and output messages. These structures can be accessed by other CQSs, allowing them to share the message queues.

CQS shared queues structures

All IMS messages are stored in one of several possible list structures in the Coupling Facility. There is one required and one optional structure for full function messages, and if Fast Path is defined in your system, one required and one optional structure for Fast Path messages.

- Full function primary structure

  This structure is *always required.* It holds all input and output message traffic for IMS other than those messages defined as Fast Path expedited message handler (EMH) messages.

- Full function overflow structure

  If this optional structure is defined, it is used by CQS to prevent the primary structure from filling up. When the primary structure reaches a user-specified full threshold (for example, 80% full), those queues using the most space in the primary structure are moved to the overflow structure, freeing up space in the primary structure.

- Fast Path EMH primary structure

  This structure is required only if the IMS system is defined in the IMS system generation with Fast Path enabled (FPCTRL macro). If it is, then this structure will serve the same purpose for EMH message traffic that the full function primary structure does for full function message traffic. Note that this structure is required even though you are using only Fast Path databases and no EMH transactions. It is not required if you have not included the FPCTRL macro in your system definition.

- Fast Path EMH overflow structure

  Like the full function overflow structure, this is used to prevent the primary structure from filling up.

CQS data sets

The CQS address space has several data sets used for restart purposes and for structure backup and recovery. Some of these data sets are unique to each CQS, and others are shared by all CQSs in the shared queues group.

- Full function checkpoint data set (unique)

  Each CQS has a full function checkpoint data set. CQS takes system checkpoints to record information about the current status of the full function shared queues. This information can be used to resynchronize CQS and IMS when they reconnect following a planned or unplanned outage.

- Fast Path EMH checkpoint data set (unique)

  If you have an EMH structure, then CQS must also have an EMH checkpoint data set. Its purpose is the same as that of the full function data set.

- Structure recovery data sets (2) (shared)

  There are also a *pair* of structure recovery data sets (SRDSs) which are *shared by all the CQSs in the shared queues group*. These data sets are the functional equivalent of an IMS SNAPQ (or DUMPQ), and when a structure checkpoint is taken, all of the list entries (messages) currently in the shared queues structures are read and written to one of these two data sets. The next time a structure checkpoint is taken, the other SRDS is used. They continue in this flip-flop fashion so that we are never writing over our latest good copy of the data.

  The entries in the SRDS, along with the logs (described later) are used to recover a failed shared message queue structure similar to the way a BUILDQ works for non-shared messages queues.

List notification vector

IMS registers interest with CQS in queues which it can process. For example, if a transaction (TRXN1) is defined to IMS1, then IMS1 will want to know when there is an instance of TRXN1 on the shared queues. If IMS2/CQS2 put it there, IMS1/CQS1 would have no way of knowing. This notification is done using a Parallel Sysplex service called *event queue monitoring.* When an event occurs which CQS has expressed interest in (on behalf of its client IMSs), CQS is notified. For example, if one of the CQSs puts a TRXN1 message on an empty queue (that is, there are no other TRXN1 messages already on the queue), the Coupling Facility Control Code (CFCC) will notify CQS by turning on a bit representing that CQS in a *list notification vector (LNV)* in the hardware system area (HSA) of the processor. This drives an exit in CQS which determines what message just arrived. CQS then notifies IMS that there is a transaction on the queue.

This process is somewhat similar to the buffer invalidation process where a bit in the local cache vector is turned on to indicate that a buffer is invalid.

System Logger (LOGR)

In order to be able to recover the shared message queues in the event of a failure, all message queue activity must be logged. CQS does not have its own logger. Instead, it invokes the services of the System Logger to log shared queues activity. There must be one System Logger address space in each OS/390 LPAR. All instances of the System Logger together are responsible for the integrity and recovery (if necessary) of the log stream. To help ensure recoverability, the System Logger will *duplex* all log streams, including CQS's. The duplexed copy is maintained either in a data space or in a data set. This is a user option and will be addressed later.

Message queue logger structures and log streams

Log records written to the System Logger by *all of the CQSs in the shared queues group* are merged together into a single *log stream* in a Coupling Facility list structure referred to as the logger (or LOGR) structure. Since all CQSs are updating the same set of shared queues structures, the log records representing those activities must be merged, There is one log stream (and one structure) for the full function queues and another for the EMH queues. These log records, along with the SRDSs, are used to recover a failed message queue structure.

LOGR data sets

There are two types of data sets associated with the System Logger.

- Offload data sets

  If CQS continuously wrote log records to the logger structure, it would eventually fill up. When the OLDS fills up, we archive it to an SLDS and make the OLDS available for reuse. When the SLDS is no longer needed for recovery, its entry in the DBRC PRILOG record is deleted. When the logger structure reaches a user-specified threshold (for example, 75% full), the System Logger will offload them to VSAM linear data sets called offload data sets. These are the functional equivalent of the IMS SLDSs. Eventually, the log records on those data sets are no longer needed for recovery and they are deleted. They are no longer needed for recovery when they are older than the oldest SRDS (see above).

- Staging data sets

  Each System Logger will always duplex its portion of the log stream. The user has the option to have the second (duplexed) copy either in a staging data set or a data space. This data set is used only to recovery the logger structure if it fails. As entries are moved to the offload data sets, they are deleted from the staging data set.

LOGR data space

Because the staging data sets have the potential to slow down the logging process (even the fastest DASD I/O is very slow compared to CF access), the option for staging data sets is generally not used, Instead, each logger may use a data space to keep a second copy of the list entries in the logger structure. If this is done, it is important that the logger structure not be on a Coupling Facility on the same hardware box as the data space. A failure of that box would lose both the logger structure and its backup copy.

XCF and XES services

Like IMS data sharing, shared queues support is tightly integrated into the sysplex services which we have described in some detail in "Sysplex services" on page 17. Since many of the above components are described there, we recommend that you review it closely before continuing, if you have not already done so. These services provide functionality, communications, and integrity, but may also impact performance.

4.2 Objectives and expectations

Sometimes, when implementing a new function or feature, or a new technology, you can lose sight of the reasons why you are doing it in the first place and get too caught up in the technology. The first step in the planning process should be to document your reasons for implementing the feature - shared queues in this case. What problems does it solve, or how does it make operation of the IMSplex easier. Then, throughout the planning process, continually review your objectives for implementing shared queues and what your expectations are for each component. When the migration process is complete, come back one more time and see if it meets your expectations.

4.2.1 Objectives

Objectives for implementing any new technology generally fall into one or more of the following categories:

- Capacity
- Performance
- Availability
- Recoverability
- Operability
- Functionality

You should define, in your migration plan, what your objectives are and why you are implementing shared queues.

Shared queues with data sharing

It is far more likely that you will be implementing shared queues in a data sharing environment - probably sometime after data sharing has been implemented. You should document your objectives for shared queues in this environment.

- Functionality

  A primary objective for shared queues is functionality. For shared queues, this means the ability to enter any transaction on any IMS in the IMSplex and have that transaction available to that or any other IMS in the shared queues group capable of processing it. Even in a partitioned environment where not every transaction can be executed on every IMS, every transaction can still be entered on every IMS where it can be queued and then scheduled by the IMS capable of processing it.

- Availability

  Shared queues also provides an availability enhancement for the message queues. Since these queues are no longer in one IMS's private queues, they are preserved across any type of IMS failure or restart, including a cold start.

  Additionally, without shared queues, a full message queue can bring down IMS with a U758 abend. With shared queues this is much less likely to happen (although not impossible). When the primary shared queues structure starts to become full, CQS moves some messages to an overflow structure. If the structure still becomes full, IMS stops accepting new input and logs (in the OLDS) output for later delivery.

- Capacity

  If data sharing is being implemented for capacity reasons, then shared queues can work to help distribute the transaction workload across all the data sharing IMSs. When the IMS to which a user is logged on (the "front-end IMS") is too busy to process the workload, another IMS (a "back-end" IMS") can process the message and relieve the capacity constraint.

  By sharing the message queues, online "front-end" systems with connectivity to the end-user can put messages on shared queues where "back-end" systems can be started and stopped at will as capacity needs change.

- Recoverability

  CQS can recover lost or damaged shared queues structures using the SRDS and the appropriate log stream. This is the functional equivalent of an IMS BUILDQ.

  When used in conjunction with the Common Service Layer (CSL) and sysplex terminal management (STM), available beginning with IMS Version 8, shared queues can provide the VTAM end-user with recovery of *significant status,* such as an IMS conversation. When the user logs on to another shared queues group member of the IMSplex following an IMS or LPAR failure, that new IMS can access the significant status from a resource structure. Note that shared message queue recovery is independent of IMS subsystem recovery.

Shared queues without data sharing

Shared queues by itself, without data sharing, has only limited functionality since transactions can only be executed on the IMS where the data resides. However, there are still a few reasons which we will briefly discuss as valid objectives:

- Availability

  In a non-shared queues environment, IMS input and output messages reside in the buffers and message queue data sets of each individual IMS. If that IMS fails, those messages are not available until that IMS is restarted. In a shared queues environment, messages on the shared queues can be recovered from another IMS.

- Recoverability

  In a non-shared queues environment, if IMS is cold started, the message queues are lost and can only be recovered with an external tool such as IMS Queue Control Facility (program number 5697-E99). When the messages are in Coupling Facility structures, IMS can be cold started without loss of the message queues.

- Functionality

  When end-users require access to transactions which are defined on an IMS system to which they are not connected, they must either log off their current IMS and log on to the other, or IMS must distribute that transaction to where it can be processed. This may be done, and is done in many data centers, using IMS multiple systems coupling (MSC) where transactions are defined as local (process here) or remote (process somewhere else).

Even without data sharing, shared queues could be used as a means of distributing the transactions to an IMS where they can be scheduled. A user could log on to any IMS in the IMSplex and that IMS would place the message on the shared queues from where it would be retrieved by an IMS capable of processing it. This might take the place of MSC links between IMSs within the same data center avoiding the overhead of MSC. Any transaction could be entered on any IMS and it would go to the IMS capable of processing it.

4.2.2 Expectations

Expectations are quite different from objectives. You may be implementing shared queues for its functionality, availability, capacity, or recoverability, but you should document your expectations for how (much) you expect these to improve, *or degrade*. You should also document your expectations relative to the other characteristics that are not part of your objectives.

- How do you expect your IMS system *capacity* to change?

  You should not expect the overall capacity of a data sharing environment with shared queues to change as a result of implementing shared queues. The primary reason for a capacity increase is because of the data sharing and the availability of multiple IMSs to process the total workload. Shared queues just makes it easier to realize that capacity improvement by distributing the workload to the first IMS that can process it. Shared queues can, however, provide capacity relief for a temporarily over-utilized IMS by allowing some of the workload to be processed by another less busy IMS.

- Are you expecting your IMSplex to be more *available* than currently? Why?

  Because messages on a shared queues are available to all IMSs from the shared queues structures, the IMSplex itself becomes more available. You should expect that the *end-users perception* of availability is greatly increased even though some individual IMSs may be unavailable due to planned or unplanned outages. Shared queues also enables sysplex terminal management, a component service of the IMS Version 8 Common Service Layer. See 5.3.3 “Sysplex terminal management” on page 116 for the planning considerations for sysplex terminal management.

- Do you expect *performance* to decline when shared queues are implemented? By how much? Is this acceptable? What if it is more?

  Like data sharing, you should expect that, according to some measurements, applications which share the message queues will not perform as well as those which do not share message queues. These measurements are typically internal. For example, you may measure processor consumption after shared queues implementation and realize that more CPU is required in the shared queues environment. You may even measure end-user response time and find that it went up a few percent. Like data sharing, there are a number of reasons for this, many of which can be minimized through proper planning. We will address these later in this chapter.

- Do you expect message queue *recovery* to be easier, more difficult, or about the same?

  Recoverability may be one of your objectives for shared queues. For example, messages on non-shared queues will be lost if IMS is cold started. Some user status will be lost if the user logs on to another IMS. Shared queues, especially when used in conjunction with sysplex terminal management, can provide improvements in this area, but it will not address every possible scenario. You should document what you expect in terms of message recovery when shared queues are implemented.

- Are you expecting your shared queues environment to be more difficult to *manage or operate* than a non-shared queues environment? In what way?

Shared queues certainly adds to the complexity of an IMSplex. There are new address spaces, new messages, new structures to manage, new opportunities for failure, and new procedures for failure recovery. There are also new considerations for performance and tuning. It is important to realize at this stage of planning that shared queues do not just take care of themselves - they require some care and feeding.

- Will shared queues provide any changes in *functionality*? Is there anything that you cannot do with shared queues that you could do without it? Does it provide any new function or capabilities?

  We said in the data sharing planning chapter that "*Part of the planning process will be to determine how to distribute the workload across the data sharing IMSplex, and what to do in the event that something fails.*" Shared queues is one of the techniques available for workload distribution. Other than workload distribution, and improved recoverability when used in conjunction with the Common Service Layer, there is no new functionality, or lost functionality, provided by shared queues.

4.3 Functional planning

Shared queues sounds straight-forward on the surface. Just put the messages out on the structure where everybody can get at them and it will work just fine. This was basically true for data sharing, which had only a few potential problem areas. With shared queues there are several things you must plan for in order to achieve the function and performance you expect. In this section, we will look at some of these *special considerations* and try to provide enough information for you to achieve your objectives and meet your expectations.

4.3.1 Planning considerations for shared queues functionality

There are many special considerations when planning for a shared queues environment - differences from non-shared queues that you must plan for to be successful and avoid unpleasant surprises. This is probably more true for shared queues than for any other of the IMSplex features. The following discussions address these special considerations and identify planning considerations for each.

Scheduling - full function transactions

In a non-shared queues environment, when an input transaction arrives either from the network or from an application program inserting to an alternate TP-PCB, IMS queues the message on a control block for that transaction (SMB - scheduler message block) and, when resources are available, will schedule that transaction in a local dependent region. There are many factors that determine how IMS schedules full function (non-EMH) transactions in a non-shared queues environment, including *system definition parameters, transaction status, and dependent region availability.*

System definition

The following system definition parameters are used by the IMS scheduling algorithms:

- Application definition parameters
  - SCHDTYP (serial or parallel) - determines if the PSB can be scheduled in multiple dependent regions at one time
- Transaction definition parameters
  - MSGTYPE
    - Message class - must match a dependent region message class
  - PRTY (scheduling priority)

        - Normal priority - highest priority transactions are scheduled first
        - Limit priority - allows increase in scheduling priority based on limit count
        - Limit count - determines when scheduling priority should be increased

    - PROCLIM (processing limit count) - determines how many transactions can be processed in a single PSB schedule
    - PARLIM (parallel limit count) - determines when a parallel scheduled transaction can be scheduled into another dependent region
    - MAXRGN (maximum regions) - limits the number of dependent regions that a transaction can be scheduled into
    - SERIAL (serial or parallel) - determines if a transaction can be scheduled in multiple dependent regions at one time

Planning considerations: Most of these have the same meanings with and without shared queues. The exception is the scheduling priority (PRTY) and the parallel limit count (PARLIM). These are discussed in "Global scheduling" on page 88. Some consideration needs to be given to transaction defined as SERIAL also. This is discussed in "Scheduling serial transactions" on page 90.

Transaction status

In addition to the scheduling-related definitions, the status of a transaction also determines whether it *can* be scheduled.

- Transaction and PSB status:
    - IMS will not schedule transactions whose status is stopped, pstopped, or locked.
    - IMS will not queue transactions whose status is stopped or locked.
    - IMS will not schedule transactions whose message class is stopped.
    - IMS will not schedule a transaction defined to a stopped program (PSB).

    In a shared queues environment, if a transaction is stopped or locked in the front-end IMS, that IMS will not queue it even if the transaction is not stopped on other IMSs. Pstopped transactions can still be queued but will not be scheduled. A stopped PSB or CLASS does not affect the queuing of transactions.

    Planning considerations: There is no global transaction status. To stop a transaction from queuing or scheduling, it must be stopped on every IMS. Operational procedures should be updated accordingly. The same is true for restarting queuing and scheduling.

- Transactions on the queue

    Scheduling also depends on whether there are any transactions to be scheduled, and if so, how many. In a non-shared queues environment, transactions are stored in the local message queue pool (QPOOL) and are queued locally off the SMB. IMS always knows how many transactions are available for scheduling.

    There are two issues in a shared queues environment which you must consider when developing your plans:

    - IMS does not know how many messages are on the shared queues in the Coupling Facility list structure.
    - Because of the above, some of the scheduling parameters are either ignored or are interpreted differently.

    See "Global scheduling" on page 88 for more discussion on this topic.

Dependent region availability

Dependent region execution parameters (CL1, CL2, CL3, and CL4) define the transaction message classes which they can process. In a shared queues environment, IMS may register interest in a transaction *even if there is no active dependent region for that class*. IMS will continue to be notified when a queue goes from empty to non-empty, but will never try to schedule such a transaction until there is a dependent region available with the correct class.

Global scheduling

Because messages are not queued locally, IMS does not know how many messages are queued on the transaction queue in the CF, or even if ANY transactions are queued. Since IMS does not connect directly to the shared queues structure itself, it relies on CQS to notify it when *something of interest* arrives on the queue. Something of interest is something which IMS is capable of processing. In this case, it is a transaction which is defined to IMS, and for which the transaction code, the message class, and the PSB are not stopped, pstopped, or locked. When these conditions exist, IMS *registers interest* in that transaction code (in that transaction queue). It does this even if it has no dependent regions with the associated message class (it will just never try to schedule it).

CQS must also be notified when something of interest is on the queue. It does this by invoking a sysplex service called event queue (EVENTQ) monitoring - a service of XES and the Coupling Facility Control Code (CFCC) requested when CQS connects to the structure. CQS registers interest in specific queues (called sublists) representing transaction queues in which its client IMS has registered interest.

When any CQS places a transaction on the shared message queue list structure, if that message caused that queue (that sublist) to go from *empty to non-empty,* then the CFCC notifies CQS that *something* has arrived. Notification is done by turning on a bit in the list notification vector (LNV) for that CQS. This drives a CQS exit which then invokes another XES service to determine what has arrived. CQS notifies its IMS client that a registered transaction queue now has work on it. IMS turns on a flag in the corresponding SMB that says there is work on the queue for that transaction and queues the SMB on the transaction class table (TCT) queue according to its *normal* scheduling priority. This flag stays on, and the SMB remains queued on the TCT, until IMS attempts to retrieve a message and is told by CQS that the queue is empty. At this time, the flag is turned off and the SMB is dequeued from the TCT. IMS will not try to schedule that transaction again until notified by CQS that there is work on the queue.

When a dependent region becomes available, IMS scheduling algorithms determine which transaction to schedule based on the scheduling priorities of SMBs on the TCT. IMS makes sure the PSB is loaded, allocates space in the PSB work pool, and performs other scheduling functions. It then issues a call to CQS to retrieve the transaction (equivalent to transaction priming in a non-shared queues environment). CQS issues XES calls to retrieve the message from the structure. *If the message is still there,* it is passed to IMS by CQS, control is given to the dependent region, and the transaction is processed normally.

- Scheduling priorities (PTRY)

  As described above, IMS system definition macros allow you to specify a scheduling priority on the PRTY macro. The scheduling priority determines the SMB’s position on the TCT. The PRTY macro also allows you to specify a limit count and a limit priority. The limit count is defined as the number of transactions on the queue. The limit priority is the scheduling priority to raise to when the *number of queued transactions equals or exceeds the limit count.* In a shared queues environment, IMS does not know how many messages are on the shared queues. Therefore, *these two parameters are ignored*.

- Parallel limit count (PARLIM) and maximum regions (MAXRGN)

The parallel limit count is defined as the number of queued transactions time the number of dependent regions into which this transaction is scheduled. When this product is exceeded, transactions can be scheduled into another dependent region until the MAXRGN parameter is reached, after which no new dependent regions will be scheduled for this transaction no matter how long the queue is. As with scheduling priorities, IMS does not know how long the transaction queue is. Instead, IMS counts the *number of consecutive successful get unique calls to the IO-PCB* as an indication of the queue length. For example, if PARLIM=3, when the first dependent region successfully processes three transactions per schedule, then a second dependent region can be scheduled. If the two of them together then successfully process six more (2 x 3) then a third can be scheduled, and so on. The total number is limited by the MAXRGN parameter.

- Processing limit count (PROCLIM)

  There are no changes in shared queues for interpreting processing limit count (PROCLIM).

Local scheduling

When a transaction arrives in IMS, there are conditions under which that transaction can be scheduled immediately without putting it on the Transaction Ready Queue. Those conditions are:

- A dependent region is available to process the transaction and there are no higher priority transactions waiting to be scheduled.
- There are no transactions currently on the Transaction Ready Queue (TRQ).
- The input message fits completely within a single Queue Buffer.
- The transaction is not defined as SERIAL.

When local scheduling is performed, IMS puts the input transaction directly on the lock queue (LOCKQ) instead of the TRQ. It keeps a copy of the message in its local QPOOL and proceeds to schedule that message locally. Note that this saves several accesses to the Coupling Facility, and may also save some false schedules in other IMSs trying to schedule the same transaction. Local scheduling is a good thing, and part of your planning should include how to configure your dependent regions and transaction classes to maximize local scheduling by trying to always, or as frequently as possible, have a dependent region available to process an incoming transaction.

False scheduling (pseudo-schedule failure)

This term is used to refer to the situation where an IMS thinks there is a transaction on the message queue, tries to schedule it in a dependent region, and then discovers that there are no more messages on the queue (IMS does not know its empty until it asks). This will happen once in each IMS when a queue goes from non-empty to empty unless another message arrives soon, so it cannot be completely eliminated. However, the number of occurrences can be minimized with local scheduling. When a transaction is scheduled locally, it is not put on the TRQ. This means the TRQ remains empty and other IMSs are not notified of a transaction arrival. Therefore, they do not try to schedule it (this is good since the first IMS has already scheduled it).

Planning considerations: It is, generally, *good practice to invoke local scheduling* to reduce the overhead of shared queues structure access and false scheduling. Local scheduling can be maximized by having many dependent regions available with the appropriate classes so that when a transaction arrives, it can almost always be scheduled locally. However, this will have the effect of eliminating some of the load balancing characteristics of shared queues. If it is important in your environment to distribute the workload somewhat evenly across the shared queues group, then you will need to limit the number of dependent regions available per class, to set the PARLIM value higher, and/or to set MAXRGN lower. This will force more transactions to the ready queue where they are available for global scheduling.

Scheduling serial transactions

Serial transactions are those that are defined with SERIAL=YES coded in the TRANSACT macro. When you define a transaction as SERIAL, IMS will guarantee that these transactions will be processed in the order in which they arrive. They cannot be parallel scheduled. If a serial transaction fails with a U3303 (data unavailable), the transaction is requeued to the SMB and the transaction coded is USTOPPED. Scheduling stops until the problem is fixed.

In a shared queues environment, serial processing is only guaranteed within the front-end IMS. Serial transactions are not made available to other IMSs in the shared queues group (there are special queues for serial transactions). If a transaction is defined as serial in multiple IMSs, each IMS will process transactions which it receives serially, but other IMSs may be processing the same transaction at the same time.

If you have serial transactions defined, you should understand why they are defined that way. If you really need them to be *serial across the IMSplex*, and be scheduled just as they are in a non-shared queues environment, then you must do the following:

- Remove the SERIAL=YES parameter from the TRANSACT macro (or code SERIAL=NO).
- Give the transaction a message class which is unique within the environment.
- Start a dependent region with that message class in only one IMS in the IMSplex. This is the only IMS which will try to schedule it. Others may still queue it, but will not try to schedule it since they will not have a dependent region with the right class.
- Write a Non-discardable Message Exit (DFSNDMX0) which, when the transaction abends with a U3303, will requeue the message and USTOP it (R15=12).

This will allow all IMSs to receive and queue the transaction, but only one IMS will ever try to process it. You may even want to /PSTOP the transaction on the other IMSs to keep them from registering interest and then being notified by CQS whenever there is work on the queue. Do not /STOP the transaction on these IMSs. They will be rejected and not placed on the TRQ.

Planning considerations: Identify all SERIAL transactions and determine whether they must be executed serially across the IMSplex. If so, then implement the steps above. If not, determine why they have been defined as SERIAL in the first place. Maybe it is not necessary and the SERIAL attribute can be removed, allowing these transactions to schedule in parallel and on any IMS in the shared queues group.

Scheduling: Fast Path EMH transactions

Fast Path transactions, sometimes called EMH (expedited message handler) transactions, are defined in the system definition as FPATH=YES on either the APPLCTN macro (Fast Path exclusive) or the TRANSACT macro (Fast Path potential). When a transaction arrives in the system from the network (they are not allowed from an MSC link or by way of an application program-to-program switch), it is passed to the Fast Path Input Edit/Routing Exit (DBFHAGU0). If it is Fast Path potential, this exit can decide to let the transaction process in a full function message (MPP) region, or scheduled it in a IMS Fast Path (IFP) region. Fast Path exclusive transactions can only be processed in IFP regions. If the transaction is to be processed in an IFP region, In a shared queues environment, this exit can also decide whether the transaction is to be queued locally or globally.

Sysplex processing code

DBFHAGU0, when running in a shared queues environment, can specify a sysplex processing code (SPC) for the transaction. The SPC has three possible values that determine how an input message is queued:

- Local only

  This means that the transaction is to be queued to a local balancing group (BALG) for scheduling and is not put on the shared queues at all. These transactions are processed just as they are in a non-shared queues environment. This differs from “local scheduling” of full function transactions in that full function transactions which are scheduled locally are still put on the shared queues (directly on the LOCKQ), and output from all full function transactions are put on the output shared queues (LTERM queue or REMOTE queue). Locally scheduled Fast Path transactions, and their responses, do not go on the shared queues and therefore can only be processed locally. This eliminates any shared queues overhead, but loses the benefits of workload distribution and availability.

- Global only

  If DBFHAGU0 sets the SPC to global only, then the transaction is put on the Program Ready Queue in the Fast Path shared queues structure even if a local dependent region is immediately available. It always incurs the overhead of shared queues but has the workload distribution and availability benefits of shared queues.

- Local first

  This third option tells the expedited message handler to queue the message on a local BALG only if there are fewer than six transactions already on the queue. If there are six or more, then queue the transaction on the shared queues. This has some of the benefits of both local and global. If the queues remain short, then the transactions process locally with no shared queues overhead. If the queues become longer, then the transactions are queued globally where another IMS can help with the processing workload.

  Planning considerations: When there are transactions on both the local queue and the global queue, those on the local queue will process first, even those that may have arrived later. If this does not present a problem for your applications, then local first is the recommended sysplex processing code (and it is the default).

Multiple systems coupling (MSC)

Multiple systems coupling (MSC) connects two IMSs together using an IMS private communications protocol. These two IMSs can be in the same data center, or they can be in some other city, state, or country. When in the same data center, the IMSs can send messages using a channel-to-channel (CTC) connection. When they are separated by long distances, VTAM is used. When planning for shared queues, there are several issues you must consider in your implementation plan:

- Defining MSC to the IMSplex
- Replacing a “same data center” MSC link with shared queues
- Using MSC to link IMSs in the shared queues group to remote IMSs

MSC definition

Although not all IMSs in the shared queues group must have *physical* MSC links to remote MSC locations, messages which come from an MSC destination will have an MSC prefix. In a shared queues environment, that message will be placed on the shared queues where any IMS in the shared queues group may read and process it. *It is, therefore, necessary for every IMS to have at least one of each of the MSC macros in its system definition.* This ensures that the required MSC code is available to process these messages. The required macros are: MSPLINK, MSLINK, and MSNAME.

As described in *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908, IMS systems in a shared queues group share MSC definitions. That is, when an IMS joins the group, it notifies other IMSs of its MSC definitions, and receives notifications from other IMSs of theirs. This notification process allows them to share MSC SYSIDs (system IDs) and MSNAMEs (remote destination logical link names) so that even an IMS which does not participate in a physical MSC connection can process input messages from, and send output messages (through another IMS in the shared queues group) to, remote destinations.

In a cloned environment, where all IMSs use the same system definition, then this is not an issue. The MSC links will be defined to all IMSs since all IMSs are using the same system definition. Note that it is OK for each IMS to define the same MSC logical and physical links. Only one of them will be able to activate that link at any given time. If that IMS fails, then those links can be activated on the surviving IMSs.

If the IMSs are not cloned, then as long as each IMS includes the three MSC macros, they can all participate in the MSC environment. As each IMS joins the shared queues group, it notifies other IMSs of its own MSC definitions. The result is that the SYSIDs and MSNAMEs are shared by all IMSs in the group, allowing them to correctly process the MSC message prefix and to queue input and output messages to the correct destination.

Planning considerations: Even if not all IMSs in the IMSplex will be connected through an MSC link to a remote IMS, every IMS should include the MSC system definitions. If they do not, then you cannot define remote transactions and LTERMs, and you cannot process messages with these destinations.

Shared queues replacing “same data center” MSC link

You cannot activate MSC links between IMSs which are in the same data sharing group. If you currently have these links, those definitions can be removed (actually, the definitions can stay - they just cannot be started). More importantly, *resources defined as remote transactions and remote LTERMs should be changed to local definitions.* For example, assume IMS1 and IMS2 are currently connected by an MSC link, and TRXN1 is defined as local to IMS1 and remote to IMS2. If IMS1 and IMS2 are reconfigured to participate in the same shared queues group, then TRXN1 should be redefined in IMS2 to be local (remove the SYSID from the transaction definition). The same applies to remote LTERMs.

If the shared queues completely replaces all MSC connections, then the MSC macros can be completely removed from the system definition. But if there still remains at least one truly remote MSC connection for at least one of the IMSs, then each IMS should continue to include at least one of each MSC macro.

If you have applications which use directed routing (issues CHNG and ISRT calls to an MSNAME destination) to route messages to IMS across an MSC link, and if those IMSs are now in the same shared queues group, then the use of directed routing must be discontinued. The MSNAME destination in the CHNG call must be replaced with a transaction or LTERM destination. Instead of changing the application program, you can also code the program routing entry point in DFSMSCE0 to tell IMS to ignore the MSNAME destination in the CHNG call and queue the message according to its destination in the message.

If you use DFSMSCE0 to do directed routing of input messages (the program routing entry point), then the exit should be changed.

Planning considerations: If shared queues are replacing MSC links between IMSs, the MSC definitions can be removed. Remember to change remote transactions and remote LTERMs to local and to discontinue the use of directed routing.

MSC between a shared queues group and a remote IMS

If *any* IMS in the shared queues group has an MSC connection to a remote IMS outside the group, then *every* IMS should include those MSC macros in their system definitions. For example, if IMS1 has an MSC connection to IMSR (a remote IMS), it may receive an input message from IMSR and place it on the shared queues. If the transaction code is defined to IMS2, IMS2 may retrieve that message to process it. It must be able to process the MSC information in the message prefix. In addition, the response to that message processed in IMS2 needs to be put on the remote destination queue with MSC routing information in its prefix so that IMS1 can deliver it.

As mentioned above, if there are any MSC connections anywhere in the shared queues group, then every IMS should include the three MSC macros - MSPLINK, MSLINK, and MSNAME. Even if the IMSs are not cloned, every IMS can have exactly the same MSC definitions since they share their definitions when joining the group anyway.

Planning considerations: If there are no MSC connections outside the shared queues group, remove the MSC definitions from your system definition. Be sure to change remote transaction and LTERM definitions to local. If there are remote MSC connections, then all IMSs must include the three MSC macros so that the MSC code will be included in the IMS nucleus. All IMSs can have the same MSC definitions even if those IMSs are not cloned.

Conversational processing

There are no changes to the concept of a conversation with shared queues. When a conversational transaction is entered, the front-end (F-E) IMS creates a conversation control block structure (CCB) and a scratch pad area (SPA) and queues the SPA as the first segment in a multi-segment message with the actual input message as the second segment (SPA+INMSG). Responses to conversational input are inserted by the application with the SPA as the first segment inserted, and the output message as the second segment (SPA+OUTMSG).

Queuing conversational messages

When a conversational transaction is received by a F-E IMS, the entire message (SPA+INMSG) is queued on the Transaction Ready Queue in the shared queues structure until it is scheduled by one of the IMSs in the shared queues group. When the transaction is scheduled by any IMS, the SPA+INMSG list entry is moved to the lock queue (LOCKQ) in the shared queues structure. When the conversational program responds, the SPA+INMSG is deleted and the new SPA and output message (SPA+OUTMSG) are put on the LTERM Ready Queue. When the F-E IMS is ready to send the response, the SPA+OUTMSG list entry is moved to the LOCKQ where it remains until the next input message is received and queued by the front-end. The SPA+OUTMSG also remains in the F-E IMS QPOOL until the next input is received.

Planning considerations: When sizing the shared queues structure, space needed for conversational transactions between iterations of the transaction (SPA+OUTMSG) must be accounted for in the total primary structure size. If conversations are HELD, they also require space on the LOCKQ. Note that messages on the LOCKQ are not eligible for the overflow structure, so space must be available in the primary structure.

Also, while non-conversational messages, once they are queued on the shared queues structure, are deleted from the QPOOL, conversational transaction messages are not deleted from the QPOOL. Therefore, the QPOOL must have enough space allocated to hold all active conversations.

Who owns the conversation?

One point to emphasize here is that a conversation is associated with the physical terminal (NODE and LTERM, or USER and LTERM for ETO) and the status of that conversation is known only to the IMS with which that terminal or user is in session (that is, the F-E IMS). This can cause confusion if the user logs off one IMS while in conversational status and logs on to another. That other IMS doesn't have a clue about that conversational status. Eventually the original SPA+OUTMSG will be discovered by the new IMS. It will invoke the Conversational Abnormal Termination Exit (DFSCONE0) which has several options for dealing with the SPA and message. None of these options includes keeping the conversation active.

IMS Version 8, when running with the Common Service Layer and a resource structure in the Coupling Facility, can keep conversational status in the resource structure available to all members of the shared queues group, allowing a user to move a conversation from one IMS to another. This function is part of sysplex terminal management (STM). See *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908, for a description of this function, and 5.1 “The Common Service Layer” on page 106, for planning considerations.

Planning considerations: When not running IMS Version 8 with sysplex terminal management active, users should be discouraged from logging off one IMS and logging on to another while still in conversational status, unless they are willing to lose their current conversation.

Holding and releasing conversations

Conversations can be held (/HOLD) and released (/REL) in a shared queues environment much the same as without shared queues. Note that the SPA+OUTMSG list entry of the held conversation remains on the LOCKQ in the shared queues structure. If too many conversations are held, it may impact the required structure size. Held conversations are not retained in the local QPOOL.

Planning considerations: Although holding conversations is supported, be sure to allocate enough space in the primary shared queues structure for the SPAs and last output message of all held conversations.

Exiting a conversation

Conversations may be terminated in several ways (in addition to abending). Some work just like non-shared queues, others work differently.

- If an application terminates the conversation, then everything works as normal - nothing different from non-shared queues (at least externally).
- If a conversation is terminated by operator command (/EXIT), then the results depend on what the current status of that conversation is. Because only the F-E even knows that the

terminal is in conversation, *the command must be entered on the F-E IMS*. But the results of the /EXIT command depend on whether the user has entered the next transaction already. If the conversation is between iterations (the end-user has not submitted the next input), then the /EXIT command works the same as with non-shared queues.

If the end-user has submitted the next input, then the status of the transaction is one of the following:

– Is sitting on the transaction ready queue waiting to be scheduled
– Is being processed on one of the active IMSs
– Has been sent across an MSC link to a remote IMS
– Has already been processed and the SPA+OUTMSG response is sitting on the LTERM Ready Queue.

In each of these cases, the F-E cannot complete exit processing because it does not know where the last message and SPA are. It does know that it will eventually be processed. So it exits the conversation and sends a message saying that the conversation has been terminated, but that the transaction is still active:

```
DFS577I  EXIT COMPLETED.  TRANSACTION STILL ACTIVE.
```

One of the IMSs will eventually find that response (SPA+OUTMSG) on the LTERM Ready Queue, but the conversation will have already been terminated and the user gone on to other things. When this happens, the Conversational Abnormal Termination Exit will be driven to handle the message and SPA.

- At session termination

  When a terminal user logs off a static terminal, or signs off a dynamic terminal, the Logoff Exit (DFSLGFX0), or Signoff Exit (DFSSGFX0) for ETO terminals, can elect to exit any active conversations. If the session terminates because of a session failure, and if GRAFFIN=VTAM (VTAM deletes the affinity whenever a session terminates), then the active conversation will be automatically terminated. If IMS fails with GRAFFIN=VTAM, then IMS will automatically exit the conversation during emergency restart.

Planning considerations: Carefully consider the implications of conversational transactions in a shared queues environment and whether it is necessary to continue a conversation on another IMS if the IMS in conversation fails. Be sure you understand the circumstances under which a conversation may be exited with an explicit /EXIT command.

If it is necessary to continue a conversation after an IMS failure, then users must be directed back to the original IMS prior to IMS Version 8. IMS Version 8 provides the capability to recovery conversational status on another IMS. This function is discussed in *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908.

Undefined message destinations

When a message arrives in IMS, before it can be queued, IMS must determine its destination. This is commonly referred to as FINDDEST. What IMS is looking for is a statically or dynamically defined transaction or LTERM. If FINDDEST fails in a non-shared queues environment, then the message is rejected. The message can be received by IMS from a network resource or from an application CHNG or ISRT call.

If FINDDEST fails in a shared queues environment, *the destination may still be valid.* It may be defined on another IMS (assuming the IMS systems are not cloned). The F-E IMS must determine whether an unknown destination is valid. If so, then it may dynamically create a control block structure to be used for enqueuing the message on the shared queues. The dynamically defined destination can be a local or remote transaction, or if ETO is enabled, it can be a local or remote LTERM. In both cases, the Output Creation Exit (DFSINSX0) makes the determination.

Dynamic transactions

Even without ETO, DFSINSX0 can be used to define a message destination as a transaction for purposes of letting that IMS queue the message on the Transaction Ready Queue. However, the exit must define all of the parameters of the transaction that would normally be defined on the TRANSACT macro. If it does not do this, then the defaults will be used and it is unlikely that they will be correct. The parameters the exit can define are documented in the *IMS Version 8: Customization Guide,* SC27-1294 (or equivalent earlier documentation), and include:

- Local/remote/SYSID
- Conversational/SPA length/truncated data option
- Response mode
- Fast Path

Because it would be difficult for an exit to know these attributes without prior knowledge, we suggest that it not be used for defining dynamic transactions. A better solution would be to add a static transaction definition using online change.

Dynamic LTERMs

To dynamically define an unknown destination as an LTERM, the extended terminal option must be enabled. In this case, the ETO support with and without shared queues is the same. That is, the Output Creation Exit (DFSINSX0) must define the characteristics of the logical terminal.

Planning considerations: Although it is possible to dynamically define transactions, you should avoid this and use online change to add a valid static transaction to an IMS system. Creating dynamic LTERMs is discussed in the ETO section of this chapter.

Extended terminal option

The extended terminal option (ETO) allows an IMS system to dynamically create a set of control blocks representing a VTAM terminal, user, and logical terminal not statically defined to that IMS. ETO is invoked at logon time, when a VTCB is created, and at signon time, when a user structure consisting of a USER control block (SPQB) and one or more logical terminal control blocks (CNTs) are created. ETO may also be invoked when a message with an undefined destination arrives in IMS from the network or from an application program issuing a CHNG and ISRT call.

ETO at logon and signon time

When a non-statically defined terminal (LU - logical unit) logs on, and then signs on, to an IMS system with ETO enabled, that IMS can dynamically define the required control block structures (VTCB, SPQB, and CNTs). The VTCB represents the VTAM node, the SPQB represents the user, and the CNT represents an LTERM *output message queue destination*. There can be multiple LTERMs defined per VTCB and USER, just as there can be multiple LTERMs statically defined for a single NODE. Except for parallel session ISC, there are no SPQB control blocks for statically defined terminals.

In a shared queues environment, the potential problem area is the LTERM definitions. These are the only ones which represent message destinations, and therefore which have queue headers in the shared queues structure. Since output messages are queued by LTERM name, and then delivered by *any IMS with interest in that LTERM name to whatever NODE that LTERM is assigned to,* if the same LTERM name is used by different IMSs for different NODEs, then delivery of messages could get confusing, as both IMSs would register interest in the same LTERM name (but for different NODEs). This could easily happen if the LTERM name is set to the USERID and the same user signs on to multiple IMSs.

In a single IMS, that IMS will not allow an LTERM control block to be created more than once with the same name. This is true even if the user is allowed to sign on multiple times. When a user does sign on multiple times, the Signon Exit (DFSSGNX0) must assign unique LTERM names *within that IMS*. One technique might be to use callable services to determine whether an LTERM name already exists within that IMS. If a user is only allowed to sign on once, then the LTERM name can be set to the user ID, guaranteeing uniqueness within that IMS. Another technique, even when multiple signons are allowed, would be to assign an LTERM name equal to the NODE name. Since a NODE can only log on once, the LTERM names will be unique.

But with shared queues, each IMS (with ETO enabled) creates its own set of control blocks *unknown to other IMSs* in the shared queues group. It would be possible to create the same LTERM name in two IMSs assigned to two different NODEs. Until IMS Version 8, users cannot be prohibited from signing on to multiple IMSs with the same user ID. For statically defined terminals, this is not usually a problem, unless the system definitions are wrong. But when ETO is used, the method of assigning LTERM names must guarantee uniqueness across the IMSplex. If only one LTERM per logged on NODE is required, then setting the LTERM name equal to the NODE name would work, since most NODEs can log on only once (the exception is parallel session ISC NODEs). But if multiple LTERMs are required per NODE, then the Signon Exit must create unique LTERM names. Again, in a single IMS, this is not difficult. But when multiple IMSs are part of a shared queues group, then the Signon Exit does not know what other IMSs may have already been assigned as LTERM names.

There are several possible solutions to this problem prior to IMS Version 8.

- One might be to suffix the LTERM name with a character unique to the IMS on which the exit is running. For example, all LTERMs on IMSA could end with the character `A` while all LTERMS on IMSB could end with the character `B`. This may not be appropriate for all IMS systems - especially if the applications are sensitive to the LTERM name (for example, they have a CHNG/ISRT call to insert to a hard-coded LTERM name).
- Set the LTERM name equal to the NODE name. This works if you only have a single LTERM per NODE. It also requires that the applications not be sensitive to the LTERM name.

Whatever technique you use, you must guarantee that LTERM names are unique across all IMSs in the shared queues group.

ETO at logon and signon time with IMS Version 8

IMS Version 8, when running with the Common Service Layer and a resource structure, provides some help in this area. When an LTERM becomes active on one IMS, an entry is created in the resource structure which prohibits other IMSs from activating the same LTERM name. It also supports *global callable services*. A Signon Exit, before it assigns an LTERM name to a user, can determine whether that LTERM name is already in use by another IMS. If it is, then the exit can use a different name. Single signon can also be enforced across multiple IMSs, allowing you to use the USERID as the LTERM name and know that it will be unique.

ETO and autologon printers

When ETO is used, SLU1 printers can be logged on automatically whenever there is output queued. This can be done by a user descriptor with AUTLGN=nodename or by the Output Creation Exit (DFSINSX0) when an application issues a CHNG/ISRT call. When OPTIONS=RELRQ (release request) is specified, IMS will release the session if another IMS requests it. This is the essence of shared printer support, allowing multiple IMSs to share the same printer.

In a shared queues environment, applications may execute on any IMS in the shared queues group. If the same application, running on two different IMSs, were to issue the CHNG/ISRT calls to an LTERM with the autologon NODE set to the same physical printer, each IMS would attempt to acquire the session with the printer. One would get the session, and the second would wait in a queue for the first to release the session. When all output messages are drained from the first IMS, it releases the printer (closes the session), allowing the second IMS to acquire the session. This can happen repeatedly, causing printer “thrashing” or “flip flopping” between the two (or more) IMSs.

To address this problem, it is best to let just one of the IMSs handle the printers. Each IMS should define the printers with OPTIONS=NORELRQ (meaning once the session is established, do not release it when another IMS requests it). That one IMS would then send all output to the printer. If that IMS terminates, the second IMS would acquire the session and take over responsibility for sending output to that printer.

Dead letter queue

Without shared queues, if messages stay on a dynamically defined queue more that some user specified time (DLQT), then they are identified as dead letters and can be displayed using the /DISPLAY USER DEADQ command. The purpose is to identify destinations which were dynamically defined by DFSINSX0, but which are not really valid and will never be delivered.

This does not work in a shared queues environment since messages are not queued locally. They are on the shared queues structure and IMS doesn't know what or how many are out there, or how long they have been there.

New parameters on the display command (QCNT and MSGAGE) is a request for IMS to query the structure for messages on a specified queue type which have been there for more than the user specified number of days. For example, specifying

```
/DIS QCNT LTERM MSGAGE 5
```

would identify all messages on the LTERM queue which have been there for more than five days. The result is the total count of messages queued for any LTERM which has “aged” messages, and the number which exceed the MSGAGE parameter. The timestamps of the oldest and most recent message are also displayed. These messages can then be deleted using the command:

```
/DEQ LTERM ltermname PURGE | PURGE1
```

When using shared queues, aged messages on the transaction queue can also be displayed using the same commands:

```
/DIS QCNT TRANSACTION MSGAGE 3
/DEQ TRAN trancode PURGE | PURGE1
```

Planning considerations: The DLQT parameter will be ignored. You should change your operating procedures to periodically check for messages which have been on a queue for an extended period of time. If the destination is invalid, or no longer in service, dequeue the unwanted messages.

Cold starting the shared queues

Cold starting IMS does not result in an empty message queue as it does when shared queues are not used. Some IMS installations use a cold start to eliminate unwanted messages from the queues and start IMS with a clean set of queues. To achieve the same effect with shared queues, you must terminate all CQSs and do the following:

- If there are any failed persistent connections to the shared queues structure, force the connections:

  ```
  SETXCF FORCE,CONNECTION,....
  ```

- Delete the structures (both primary and overflow if they exist):

  ```
  SETXCF FORCE,STRUCTURE,STRNM=structure-name
  ```

- Delete and redefine the structure recovery data sets (SRDSs). If you do not do this, when CQS is restarted, it will detect that the structures are gone and recovery them from the SRDS and the log stream.

Planning considerations: Include procedures for cold starting the shared queues in your operational procedures.

Online change

The online change process is used to activate updates made offline to the ACBLIB, FMTLIB, MODBLKS, and MATRIX data sets. When IMS systems are cloned, or even when they are using many of the same definitions (for example, the database descriptors), then it is critical that when resources are changed, the change be coordinated across all IMSs. Every effort should be made to clone these IMS systems. Even if not every system uses every resource, cloning reduces the chances of error for this as well as other processes.

Online change prior to IMS Version 8

When the online change process is run, the Online Change Copy Utility queries the MODSTAT data set to find out which libraries are inactive (A or B) and uses these to copy into from the staging data sets. Just to be sure, it gets a global enqueue on the data set name to be sure no other user is using it. When these libraries are shared, it is important that all IMSs use the same version (A or B). If they are not, for example if IMSA were using ACBLIBA and IMSB were using ACBLIBB, then the staging utility would not be able to get the enqueue. If they are not shared, then of course there is no problem with the staging utility, but it must be run for each of the non-shared libraries using the MODSTAT for that particular IMS. This could be an error prone process and it would be best, if possible, to share these libraries - more easily done if the IMS systems are cloned.

The online change process itself requires preparing each system for online change, then committing the change(s):

```
/MODIFY PREPARE xxxx
/MODIFY COMMIT
```

These commands are entered independently on each IMS. It is very possible that one IMS could succeed, while another fails. When this happens, each IMS is running on two different sets of libraries. Your choice is to either correct the problem with the failing IMS or return the successful IMS to the original set of libraries.

Planning considerations: Review your online change procedures and be sure that procedures are in place to handle the case where online change is successful on one or more IMSs and fails on one or more IMSs.

Global online change with IMS Version 8

When IMS is running with the Common Service Layer (CSL), IMSs making up the shared queues group can elect to participate in coordinated global online change by specifying OLC=GLOBAL in PROCLIB member DFSCGxxx (new with IMS Version 8 and CSL). When global online change is used, IMS and the Resource Manager coordinate the process across all IMSs in the IMSplex. This function is described in *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908.

Planning considerations: Refer to 5.3.2 “Coordinated global online change” on page 115.

IMS and CQS logging

Multiple IMSs and CQSs may be involved in the processing of a single IMS transaction. Each of these subsystems will log the activity that pertains to it. Many of the same log records found in a non-shared queues IMS will also be found in the shared queues IMS. However, since any transaction may be received on one IMS (the front-end), processed on another IMS (the back-end), and then have the response delivered by the front-end IMS, with multiple CQS and shared queues structure accesses, the total collection of log records associated with processing a single transaction will exist on multiple logs. Each IMS logs input and output messages on the OLDS. CQS logs activity to the shared queues using System Logger services to write to a common, shared log stream.

Each IMS logs the activity that pertains to that IMS, such as input messages, output messages, enqueues, dequeues, sync points, and so forth. Each CQS logs activity that pertains to the shared queues structures, such as putting a message on the shared queues, reading a message and moving it from one queue to another, and deleting a message.

And remember, CQS does not have its own internal logger. Instead, CQS requests the System Logger to write a log record to a log stream defined in the LOGR policy and in the CQSSGxxx PROCLIB member. All CQSs, since they are updating a common shared queues structure, write log records to a common log stream. See “System Logger (IXGLOGR)” on page 47 for a description of the System Logger and how CQS uses its services.

The net result of this cooperative processing of an IMS input transaction and the output response is that message queue log records will be found on the IMS log that received the message, the CQS log stream, and the IMS log that processed it.

Each IMS and CQS message queue log record contains a 32-byte Unit of Work ID (UOWID). The first 16 bytes identify the original input message. The second 16 bytes identify messages generated as a result of that first input message, including program-to-program message switches and other messages inserted to the IO-PCB and any ALT-PCB. By looking at all message queue log records with the same first 16 bytes, you can trace a transaction and all spawned transactions, with their outputs, through the entire IMSplex.

And, there is a new log record - x'6740'. This log record is created whenever an IMS cold starts following a failure and CQS moves the inflight (indoubt) messages to the COLDQ. These log records also contain the UOWID, which may be used for analysis to determine a course of action.

Note that CQS maintains two separate log streams - one for MSGQ activity and one for EMHQ activity. The log records created are somewhat different for each log stream.

Planning considerations: The System Logger will require a log structure and offload data sets, and optionally, staging data sets. Sizing of each of these components is required. Refer to the IMS chapter in the IBM Redbook *Everything You Wanted to Know about the System Logger,* SG24-6898 for a discussion on CQS’s use of the System Logger and how to size the logger structure and data sets.

Extended recovery facility

When using extended recovery facility (XRF), it is still necessary to define local message queue data sets for *both the active and the alternate.* Only the alternate uses them, but the active will not come up with XRF enabled if these data sets are not defined. For you non-XRF wizards, these are NOT the regular message queue data sets. The DDNAMEs end with the character `L`:

```
//QBLKSL DD
//SHMSGL DD
//LGMSGL DD
```

At initialization, the XRF alternate will start a CQS address space, but does not register or connect. The XRF CQS does not connect to the shared queues structures during tracking. At takeover, the alternate (new active) will register and connect to CQS, resynchronize, and merge its local messages with the global queues. Processing then continues with the new active processing global transactions.

Planning considerations: Be sure to include the "L" suffixed data sets in your active and alternate control regions. They will not be used by the active system, but are required anyway. You also need a CQS address space in the OS/390 image where your XRF alternate is running. If that OS/390 image also contains an active IMS, they can share the same CQS.

Remote site recovery

Remote site recovery (RSR) will work in a shared queues environment, but *the shared queues themselves are not recovered* - only the databases. At the remote site, for an unplanned takeover, an /ERE COLDCOMM is required and the shared message queues are empty.

It is possible, for a planned takeover, to recover the shared queues. This can be done by taking a structure checkpoint when the last CQS shuts down, then sending that SRDS to the remote site. When CQS is started at the remote site, it will discover an empty structure and recover it using the SRDS.

Planning considerations: Your procedures at the remote site should recognize that the shared queues will not be recovered in the event of an unplanned takeover. If you want to be able to recover your message queues after a planned takeover, establish procedures for taking a structure checkpoint after all message queue activity is quiesced and sending that SRDS to the remote site and making it available to CQS.

4.3.2 Sizing the shared queues structures

One of the more difficult tasks is *accurately* sizing the shared queues structures. For a database, you can predict how much data will be in the database and size it accordingly. For data sharing structures, you can identify quite accurately how much space is required and size them accordingly. However, for the shared queues structures, *there is no right answer*. Or at least no numerical right answer. The right answer is *big enough* to hold all your messages. But what is big enough? The answer to that is, it all depends. So, we will discuss in this section what it all depends on.

Components of the full function shared queues list structure

As shown in Figure 2-12 "List structure components" on page 38, shared queues structures are serialized list structures, and as such, are composed of the following components:

- Lock table

- List headers
- List entries:
  - List entry controls
  - Adjunct area
  - Data elements
- Event monitor controls

The first two of these can be considered list structure overhead and will require a minimal amount of storage in the list structure. Fortunately, you do not have to know these numbers. The CFSIZER tool will factor these into its size estimates. What takes the most amount of space are the list entries, consisting of list entry controls, adjunct areas, and one or more data elements, and the space allocated for event monitor controls.

List entries

The largest component of any shared queues structure, at least in the sizing calculations, are the list entries. There is *one or more list entries for every IMS message* on the shared queues, plus some additional list entries for CQS, although these tend to be minor in terms of space requirements. A list entry represents the contents of a single IMS queue buffer. If a message requires more than one queue buffer, then there will be more than one list entry. Although the size of the list entry controls may vary with the level of Coupling Facility Control Code, an estimate of 200 bytes is reasonable. Each list entry also has a 64-byte adjunct area. The third component, the data element(s), are 512 bytes apiece.

The CFSIZER tool will do some basic calculations based on input provided by you. This tool provides some help text and asks for input to be used in calculating an estimated size. It can be found at:

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

It will ask you to provide numbers based on current IMS experience with local message queue sizes. Since each buffer requires one list entry, and the number of data elements is never more than what will fit into one of these buffers, the formula should provide a size that is larger than would have been needed if both the short and long message high water marks were reached at the same time. This information comes from the x’4502’ statistics log records and from the system definition. While this does not produce an *accurate* calculation of the size requirements of the shared queues structure, it does provide a “ballpark” number (remember, there is no accurate number). You may want to make this larger or smaller, depending on your estimates of how many messages may be on the queue at one time, and how safe you want to be from a full message queue structure. A good approach may be to use this size to set an INITSIZE for the primary structure, with a larger maximum SIZE specified which will allow for altering the structure to a larger allocated size should the need arise.

Event monitor control area

Whatever size you select for your shared queues structure, 20% will be allocated by CQS during connect processing for event monitor controls. You can not affect this percentage. CFSIZER will factor this in, but if you decide to do your own calculation, you must include it in to your sizing calculations.

Overflow structure

This is a structure which you hope you will never have to use, but if you do, it should be large enough to handle the overflow queues. To be safe, it is best to make it at least as large as the primary structure.

4.4 Configuration, security, implementation, and operations

Because these planning activities are so often closely related to the same activities for other IMSplex planning activities, they are discussed in their own chapter, Chapter 7. “Putting it all together” on page 159.

5

Planning for the Common Service Layer

The Common Service Layer (CSL) was introduced in IMS Version 8 as a means of improving a system's manageability of an IMSplex. IBM Redbook *IMS Version 8 Implementation Guide, A Technical Overview of the New Features,* SG24-6594 describes the CSL, and it is reviewed again in *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908.

This chapter identifies some of the planning tasks necessary, either formally or informally, to implement the Common Service Layer architecture in your IMSplex environment. The assumption here is that block level data sharing, shared queues, and end-user connectivity have already been implemented.

This chapter addresses the first two activities in the planning process for CSL:

- Objectives and expectations
- Functional planning

The next two activities are addressed in Chapter 7, "Putting it all together" on page 159:

- Configuration planning
- Security planning

CSL implementation and operations are addressed briefly in this chapter, but are covered more fully in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929.

Like all of the other planning chapters, the objective here is to ask questions which you must answer when developing your plans. Each question is followed by some discussion of what you should consider when answering the question.

5.1 The Common Service Layer

The Common Service Layer, or CSL, was introduced in IMS Version 8 to meet the growing need for systems management functions in the ever more complex IMS Parallel Sysplex environment. By the time IMS Version 7 was available, IMS had exploited the Parallel Sysplex for:

- Block level data sharing

  Up to 255 IMSs on 32 LPARs sharing IMS full function databases and Fast Path data entry databases. VSAM, OSAM, and DEDB VSO (virtual storage option) databases use XES cache services and cache structures for buffer coherency. OSAM and DEDB VSO data can be kept in the cache structures for fast access. The IRLM uses XES lock services and a lock structure in the Coupling Facility to manage lock requests for their IMS clients.

- Shared queues

  Up to 32 IMS transaction manager control regions use XES list services to share a single set of IMS full function and Fast Path expedited message handler transactions and messages stored in list structures in the Coupling Facility. The Common Queue Server (CQS) uses the OS/390 System Logger to log shared queues structure activity in list structures in the Coupling Facility.

- VTAM Generic Resources

  VTAM end-users can log on to any IMS in the IMSplex using a generic name instead of having to know the APPLID of a specific available IMS. VTAM keeps track of active IMSs and end-users in a list structure in the Coupling Facility and balances the logons across the IMSplex.

- Rapid Network Reconnect

  VTAM multinode persistent sessions keeps IMS session data available in the Coupling Facility for fast automatic reconnect following an IMS failure. When that IMS is restarted, the user's session status is restored and the user is automatically logged back on without having to reestablish the session.

- Automatic restart management

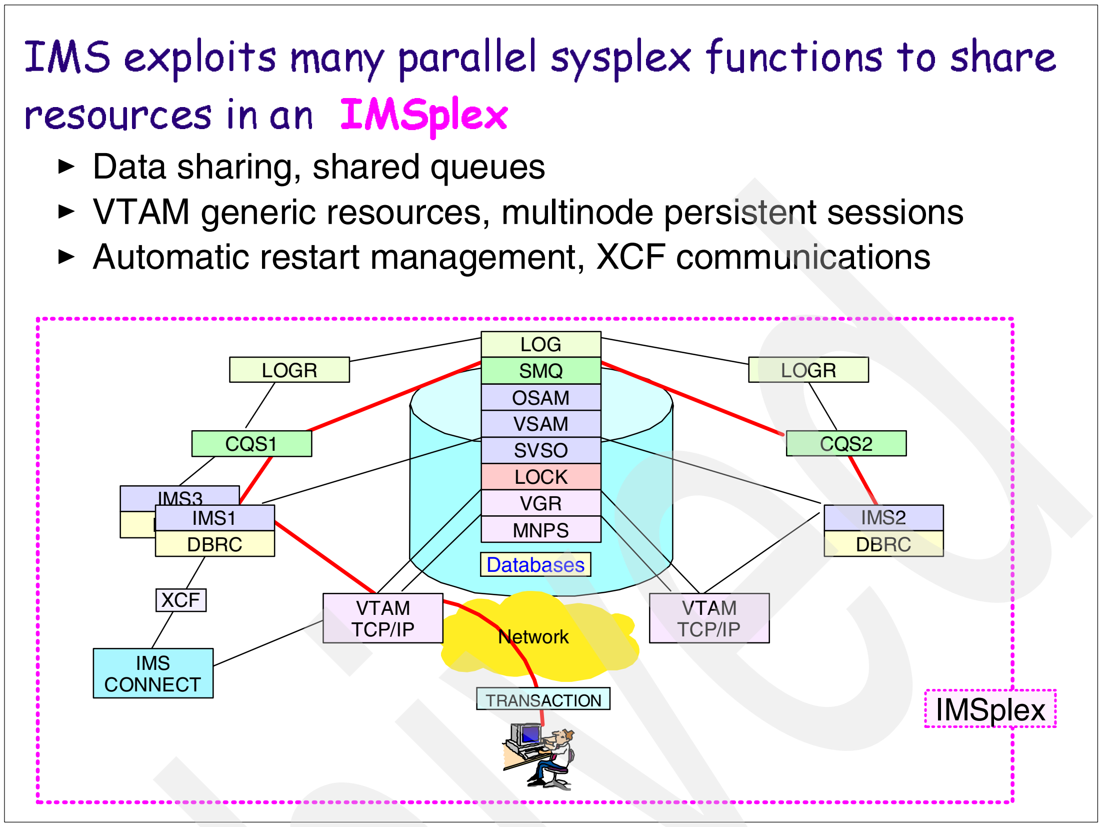
  IMSplex address spaces register with OS/390 Automatic Restart Manager for automatic local or cross-system restart following an address space, OS/390, or LPAR failure. Users can customize the restart support by updating and activating an ARM policy in the ARM couple data set.

- XCF communications

  Programs such as the IRLM, IMS Connect, and WebSphere MQ can communicate with each other or with any IMS in the IMSplex using XCF communication services.

Figure 5-1 is a highly simplified look at the IMSplex by the end of IMS Version 7. Although only two LPARs and three IMSs are shown, the complexity grows dramatically when these numbers get higher. As many as 32 LPARs and 255 IMSs can be involved in the IMsplex.

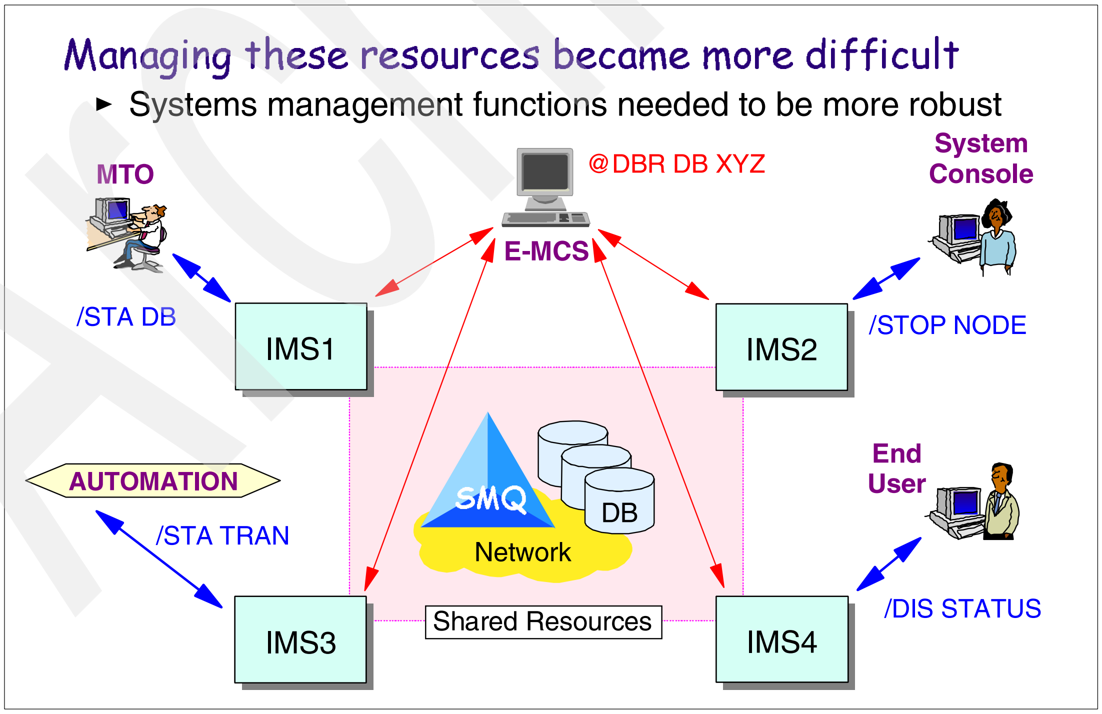
But while the benefits of the IMSplex to the end-user and applications developers grew significantly, so did the complexity of managing the environment. Figure 5-2 shows the numerous methods that IMS operations personnel used for managing the IMSplex.



IMS exploits many parallel sysplex functions to share resources in an **IMSplex**

- Data sharing, shared queues
- VTAM generic resources, multinode persistent sessions
- Automatic restart management, XCF communications

Figure 5-1 The IMSplex by the end of IMS Version 7



Managing these resources became more difficult

- Systems management functions needed to be more robust

Figure 5-2 Operational complexity

And although these IMSs functioned together in a single IMSplex, presenting a single image to the end-user, they did little to coordinate their definitions or activities. So, while the IMSplex presented a nice single image to the end-user, this was only true while that user's IMS remained available. In the event of an IMS or OS/390 failure, it was difficult, sometimes impossible, for end-users to switch IMSs and resume their status on another IMS. What was needed was an infrastructure that provided for better resource management and operations management.

5.1.1 Components of the Common Service Layer

The Common Service Layer is an extension of the IMS architecture, which provides improvements in IMSplex systems management functions. The infrastructure itself consists of *three new address spaces* and an optional (new) Coupling Facility list structure, called the *resource structure*. The address spaces *provide services* to their clients, and the resource structure is used to *store information* about the IMSplex itself, about the terminals and users in that IMSplex, and about processes critical to the IMSs in the IMSplex.

Figure 5-3 shows this CSL architecture, including the three new address spaces, and the new resource structure. This figure shows the configuration on a single LPAR, but as we have already indicated, there may be as many as 32 LPARs in the IMSplex. Some of these components may reside on multiple (or even every) LPAR, while others may exist only once in the IMSplex. Part of the planning process is to define your configuration, making decisions about which components reside where, and which CSL functions you need to meet your business requirements.

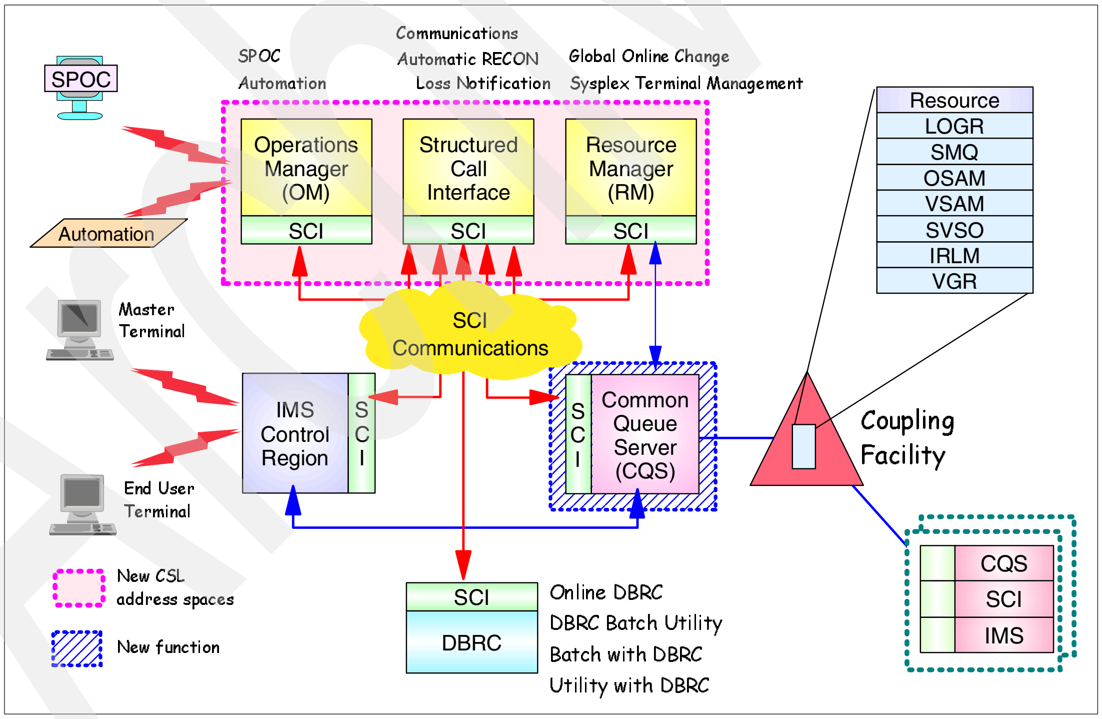


Figure 5-3 CSL architecture

IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology, SG24-6908, describes in detail the architecture itself and the services provided by the three new address spaces. This volume addresses the planning considerations for implementing the CSL architecture and exploiting the services available.

Structured Call Interface

The Structured Call Interface (SCI) has two primary functions in the CSL:

- Provide the entry point for IMSplex members to register as a member of the IMSplex.
- Provide for a common communications interface for registered IMSplex members to communicate.

Operations Manager (OM)

The Operations Manager provides a command entry and response interface to the IMSplex and is the basis for IMSplex operations in IMS Version 8 and in the future. OM performs the following functions:

- Provides services to command processing clients (for example, IMS) to register commands which they will accept from a command entry client (for example, a TSO single point of control application, or a user-written automation program).
- Provides an API for clients to enter commands and route those commands to any or all IMSs in the IMSplex.
- Consolidates the responses from the IMS command processing clients and presents them to the command entry client.
- Provides a facility for command authorization prior to forwarding the command to IMS.
- Provides user exit interfaces for input, output and security.

IMS Version 8 delivers two OM capabilities as part of the product:

- TSO Single Point of Control (SPOC) - an ISPF application which registers as a member of the IMSplex and interfaces with OM for the client services identified above.
- REXX functions and variables to allow the user to easily write REXX execs to interface with OM to enter commands and receive consolidated responses. An example would be a REXX exec invoked by NetView® based on an IMS message.

In addition, the DB2 Admin Client, includes an IMS Control Center capability which provides a graphical user interface (GUI) to OM and IMS. The DB2 Admin Client is available from the DB2 Web site for free:

http://www.ibm.com/db2

Resource Manager

The Resource Manager (RM) provides the infrastructure for IMSplex resource management and global process coordination. RM clients, such as IMS, exploit RM services to provide:

- Global process coordination supports a global online change process across all IMSs in the IMSplex requesting it through the OLC=GLOBAL parameter in DFSCGxxx.
- Sysplex terminal management provides resource type consistency checking, resource name uniqueness, and resource status recovery.
- Global callable services allows a user exit to receive global information about IMSplex terminal resources.

Resource structure

The resource structure is a list structure in the Coupling Facility, which is used by RM to maintain IMSplex global and local information about its registered members, and to keep sysplex terminal and user status information supporting the sysplex terminal management function. This structure is optional, but is required for sysplex terminal management.

The above address spaces and services are described in detail in *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908.

5.2 Objectives and expectations

Sometimes, when implementing a new function or feature, or a new technology, you can lose sight of the reasons why you are doing it in the first place and get too caught up in the technology. The first step in the planning process should be to document your reasons for implementing the feature - the Common Service Layer in this case. What problems does it solve, or how does it make operation of the IMSplex easier. Then, throughout the planning process, continually review your objectives for implementing CSL and what your expectations are for each component. When the migration process is complete, come back one more time and see if it meets your expectations.

5.2.1 Objectives and expectations for implementing CSL

Objectives for implementing any new technology generally falls into one or more of the following categories:

Performance

CSL should not be considered as a performance enhancement. Although performance studies are not available at this time, only sysplex terminal management has significant access to the Coupling Facility, and then only when SRM=GLOBAL (global status recovery mode) is in effect. Relative to other Coupling Facility accesses for data sharing and shared queues, this should not present significant additional overhead.

Capacity

Like performance, CSL should not be considered a capacity enhancement. Neither is it likely to diminish the capacity of any IMSplex members. However, because of some of its other features, like sysplex terminal management and resource status recovery, it may make implementation of an IMSplex for capacity reasons more practical, or at least easier to manage.

Availability

From a system perspective, CSL does not alter the availability of any IMSplex component. For example, when an IMS fails, it still must be restarted and in-flight work backed out. The automatic RECON loss notification (ARLN) function does enhance RECON availability by notifying all connected users of a RECON reconfiguration, allowing a spare to be restored. However, CSL's greatest availability benefit is for the connected end-user when SRM=GLOBAL is in effect. For example, if you are in an IMS conversation, and the IMS you are connected to fails, then you can continue the conversation on another IMS without waiting for the failed IMS to be restarted. For users entering Fast Path expedited message handler (EMH) transactions, a transaction can be entered on one IMS and the response retrieved on another. Similarly, if you are using a set and test sequence numbers (STSN) device, the sequence numbers can be resumed on another IMS in the event that the IMS you are connected to fails. Each of these availability functions is independently selectable.

Recoverability

As described above under availability, the recoverability benefits of CSL apply more the end-user than to any IMSplex component. For the end-user, recoverability of significant status is provided by the sysplex terminal management function.

Systems manageability

As the IMS systems and IMSplexes become more complex, they also become more difficult to manage. Just stopping a database on all IMSs in the data sharing group can be cumbersome and verification that it is complete is not straight-forward. Consistent system definitions become more important when end-users are provided with a single image of an IMS processing complex. Changing those system definitions in a consistent fashion is equally important and even more difficult. As the number of IMS control regions grows, so does the complexity of managing them.

So there may be several reasons for implementing an IMSplex, including performance, capacity, and availability. The main objective of CSL is to ease the operational and resource management of that IMSplex easier. Most users will implement CSL with this objective in mind.

Functionality

CSL functionality is primarily directed towards the improvement of the systems management requirements of an IMSplex. It does not offer the application system any function it does not already have - it just makes it easier to manage those systems.

The unique CSL functions are identified in the following question. It is perfectly valid to implement CSL in an environment for just some of its capabilities, and not use, or disable others. For example, a valid reason for CSL may be for the operations management capabilities, including single point of control (SPOC), and not to enable sysplex terminal management, or perhaps for the SPOC and global online change. Note that STM is not available if the IMSplex is not also sharing the message queues. Other functions do not require shared queues.

5.2.2 CSL functions

Your choice of CSL functions to use will depend on your objectives and why you are implementing CSL in the first place. Each of the functions identified below may address some of your objectives.

Automatic RECON loss notification

The automatic RECON loss notification (ARLN) feature is used to notify all DBRC instances immediately when any one DBRC instance detects an error on the RECONs and reconfigures, copying a good copy to the spare and deallocating the bad copy. Reconfiguration also occurs when the CHANGE.RECON REPLACE(RECONn) command is entered to force a switch of RECONn to the spare. In order to replace the spare, all jobs and subsystems which have allocated the RECONs must recognize the reconfiguration, switch to the spare (now the new active) and deallocate RECONn. This is especially useful when there may be long running batch jobs that access the RECONs only at the beginning and end and will not recognize the need to reconfigure for perhaps several hours.

Your decision as to whether or not to implement ARLN may depend on your experience with RECON failures or switches. There are no performance implications. However, once a set of RECONs is enabled for ARLN, the IMSplex name is stored in the RECON header and connection to those RECONs by DBRC instances which are not members of the same IMSplex (or any IMSplex) becomes more difficult.

Sysplex terminal management

Sysplex terminal management (STM) requires shared queues. Among the benefits of shared queues is the ability to shield the end-user from the need to know the actual configuration of the IMSplex. Any transaction can be entered from any terminal to any IMS in the IMSplex and it will execute on any available IMS with the response being returned to the input terminal. Sysplex terminal management furthers that objective by enhancing end-user recoverability when an IMS control region fails. There are three basic functions within STM. The first two of these are not independently enabled or disabled. If CSL is enabled in the IMSplex and shared queues is active, then they are automatically enabled. The third (resource status recovery) is controlled by the user with execution parameters.

Resource type consistency

This function prevents the same resource name from being defined for two different resource types on two (or more) different IMSs in the IMSplex. For example, it prevents a name from representing an LTERM on IMS1 and a transaction code on IMS2. This function is automatically enabled with CSL.

Resource name uniqueness

This prevents the same resource from being active on two or more IMSs in the IMSplex at the same time. For example, an LTERM cannot be active on IMS1 and IMS2 at the same time. This function is automatically enabled with CSL and cannot be disabled. Some users in a shared queues environment may intentionally have defined two physically different printers with the same LTERM name so that whenever a message is queued to that LTERM, either IMS can send it. If the printers are side-by-side in a printer room, this would be perfectly fine. This will not work with STM. The second printer LTERM could not be activated while the first one is active.

Resource status recovery

This function enables the recovery of terminal and user significant status when a session on a failed IMS is restarted (log on + sign on) on a new (surviving) IMS. Global status recovery requires a resource structure (list structure) in the Coupling Facility. Significant status may be end-user status or command status. An example of command significant status is a stopped NODE (/STOP NODE NODEA). This status is kept globally and is available to all IMSs in the IMSplex. NODEA is stopped everywhere. An example of end-user significant status is conversational status. If USERA is in a conversation on IMS1 and IMS1 fails, USERA can log on to IMS2 and continue the conversation without having to wait for IMS1 to be restarted.

While the first two STM functions are always enabled, resource status recovery can be tailored to your needs through execution parameters and the existence (or non-existence) of a resource structure. Activity to the resource structure is the only STM activity that could have any performance impact on your system, and you should understand what the overhead is before deciding on this function.

Global online change

Global online change is a combined function of IMS and the Resource Manager. Like its name implies, it coordinates the online change process across multiple IMSs in the IMSplex. Global online change is optional. It is controlled by parameters in an IMS PROCLIB member (DFSCGxxx) and a new IMS system data set (//OLCSTAT). It eases the burden of manually coordinating the online change process across multiple IMSs and then having to deal with a failure of the process on one of the IMSs while being successful on others.

Note that it is not necessary for every IMS in the IMSplex to use global online change. Some may be global while others are local. If your IMS systems are not clones of each other, then it is likely that you will not want them all changing libraries at the same time, and so would continue using local online change. The biggest impact is that an IMS cold start is required to either switch from local online change to global online change, or vice versa.

Operations management

Controlling an IMSplex environment can be very complex when multiple IMSs are sharing databases and message queues. The traditional means of controlling a single IMS system is through the master terminal operator (MTO) or automation (either user-written or vendor provided). Unfortunately, these techniques apply only to a single IMS. When multiples are involved, the same commands must be entered independently to each IMS. Verification that they have executed successfully is also an independent action. IMS Version 6 provided support for the entry of IMS commands from an E-MCS or MCS console to multiple IMSs with the response from each IMS returned to the console. Use of this support was awkward at best.

IMS Version 8 with the CSL provides a new interface through the Operations Manager. Users can enter commands to any or all IMSs with the responses returned from each IMS within a user specified time limit. The interface is documented and can be used for both terminal-entered commands and for automation programs or execs. IMS Version 8 provides a TSO Single Point of Control (SPOC) application which can be used to help manage the IMSplex. An IBM tool called IMS Command Center is also available which provides another GUI-based entry point for IMS commands. Refer to “IMS Control Center” on page 122 for more details.

You may choose to have multiple SPOCs, multiple IMS Control Centers, and of course multiple automation execs active concurrently within the IMSplex.

5.3 Functional planning

This section addresses some of the planning considerations for implementing CSL functions. Earlier in the planning phase you will have determined which of these functions you will implement.

5.3.1 Automatic RECON loss notification (ARLN)

ARLN is used to notify all other registered (with SCI) DBRC instances when any one of them reconfigures the RECONs. It allows the other DBRC instances to reconfigure immediately instead of waiting until the next RECON access. This is most important when there are long-running batch or utility jobs that may not access the RECONs for an extended period. A new spare cannot be created until all DBRC instances have closed and deallocated the old (bad) RECON.

When the first DBRC registers with SCI, the IMSplex name is place in the RECON header. Thereafter, unless permitted by exit DSPSCIX0, other IMS Version 8 DBRCs will not be able to access the RECONs without using the correct IMSplex name. This restriction does not apply to IMS Version 6 and IMS Version 7 DBRCs. They will not be denied access to the RECONs even if the IMSplex name is in the header:

- Will ARLN be used? For online? For batch? For utilities?

  A decision must be made whether ARLN will be used, and if so, will IMS batch jobs and utilities which run with DBRC enabled register with the correct IMSplex name. If they do not, they will be denied access unless authorized by the exit. If any DBRC registers with SCI, then all other DBRCs must either register with SCI and participate in ARLN, or have the DSPSCIX0 exit, which must be in the DBRC, batch, or utility address space’s STEPLIB, permit that address space to access the RECONs without registering. It is our recommendation that, if any DBRC uses ARLN, then all should use it.

- Will ARLN be used before all IMSs are at Version 8?

DBRCs prior to IMS Version 8 will never be denied access to the RECONs because of the IMSplex name. However, because they do not participate in ARLN, they will not be notified when a reconfiguration occurs and will act as they have in prior releases. You may want to consider running all your batch jobs and utilities under IMS Version 8 even if not all control regions are IMS Version 8.

- How will it be invoked? Execution parameter? DBRC SCI Registration Exit (DSPSCIX0)? Both?

  There are two ways to cause DBRC to register with SCI. One way is to include the IMSPLEX= execution parameter in the JCL. This is OK for online DBRCs since there are probably not very many of them. However, there may be hundreds or thousands of batch jobs and utilities and adding the IMSPLEX= parameter to each of them would be prohibitively labor intensive and error-prone. A better solution is to write exit DSPSCIX0 and include it in the STEPLIB of all batch and utility JCL (probably include it in the SDFSRESL library). Note that if you have both the execution parameter and the exit, the exit is passed the value of the execution parameter and can use that name or change it.

- How many sets of RECONs are in the IMSplex?

  At the time this document was written, it was possible to have multiple data sharing groups, and therefore multiple sets of RECONs, in the same IMSplex and shared queues group. If this is the case at your installation, then when any set of RECONs is reconfigured, all DBRCs will be notified, including the ones that are using different RECONs. This should not cause a problem, however, since DBRC will discover that there has not really been a reconfiguration and ignore the notification.

- Which LPARs (OS/390 or z/OS images) will need an SCI for ARLN?

  If you run batch jobs or utilities on LPARs which do not also have other IMSplex components (such as the control region), do not forget to include an SCI address space on those LPARs.

- Will DBRCs be allowed to access RECONs without joining the IMSplex?

  To do this requires that the DSPSCIX0 exit be written and specifically authorize the DBRC instance to access the RECONs without joining the IMSplex by returning RC=8. See next bullet for more discussion.

- Who is allowed to change the IMSPLEX name in the RECONs?

  The first DBRC to register with SCI for a set of RECONs sets the IMSplex name. If it is set incorrectly because of (for example) a typo in the IMSplex= field, it will be necessary to change it. Or if you decide later to change the name of the IMSplex, you must change it in the RECON header.

  The following command can be used to change the name in the RECONs. Note that this command can only be entered through the DBRC batch command utility (DSPURX00), not online (/RMC command), and you may need DSPSCIX0 to allow that utility access to the RECONs (RC=8) if you do not know what IMSplex name is in the header.

  ```
  CHANGE.RECON IMSPLEX(imsplex-name | NOPLEX).
  ```

  This command should be authorized to only a small number of people using DBRC command authorization.

- Is SCI registration security active?

  You can restrict SCI registration by defining the IMSplex name in the RACF FACILITY class, and then PERMITting only authorized users to join the IMSplex.

- What user IDs need to be defined and permitted to RACF?

  If SCI registration is protected, then it will be necessary to PERMIT all user IDs which will register with SCI. This must include:

- Online control regions
- CSL address spaces (Operations Manager and Resource Manager)
- DBRC address spaces
- Batch and utility jobs with DBRC=Y
- IMS Connect when using the IMS Control Center
- TSO SPOC user IDs
- User- or vendor-written SPOCs
- Automated operator programs and execs

5.3.2 Coordinated global online change

Global online change coordinates the online change process across all the IMSs in the IMSplex who have indicated in their DFSCGxxx PROCLIB member that they want to participate. It is not necessary for all IMSs to use global online change. Some can be global while others are local. However, it requires an IMS cold start to switch from local to global, or global to local.

- Will global online change be used?

  If the answer to this is no, skip to the next section.

- Are all IMSs cloned? Which ones aren't?

  If all IMSs are cloned, then is it reasonable to include all of them in the global online change group. If some are cloned and others not, then you must identify those that are clones and those that are unique.

- Will all IMSs participate in global online change? When? How/when will they join/leave?

  It is possible for all IMSs to participate in global online change even if they are not clones - that is, even if they have different online change libraries. This can be enabled simply by specifying NORSRCC=(ACBLIB FORMAT MODBLKS) in DFSCGxxx for those libraries that are unique. This prevents the checking of the online change libraries for consistency (that is, having the same data set name). However, if this is done, then each time the online change process is run, the libraries will be switched in all IMSs. It may not be desirable to do this if the IMSs are not defined the same.

  If each IMS is different, then global online change is probably not desirable. If some are clones and others not, then you must decide which ones you want to be global and set the other(s) to local. Remember that it requires an IMS cold start to switch from local to global, or global to local, so it is not something that you want to do casually.

- Will the online change libraries be shared? Will they be cloned? Are they unique? Which ones?

  Even if all your IMSs are clones of each other, you have several options for defining your online change libraries (ACBLIB, FORMAT, CTLBLKS, and MATRIX). Each IMS can use the same data set or each can use a copy of the same data set. For example, you may decide that you want the IMSs to share the same CTLBLKS and FORMAT library data sets, but you want each IMS to have its own copy of ACBLIB. If you have copies of any of these libraries, you can still use global online change by turning off resource consistency checking for those data sets which are copies. However, it is a user responsibility to make sure that the contents of these libraries are identical. The IMSs themselves do not check with each other to be sure, for example, that each one is using the same version of a PSB or DBD.

  If the libraries are unique because the IMSs themselves are not clones, then there are no decisions to make, other than if you have a mixture or some shared libraries, some copied libraries, and some unique libraries. The shared and copied libraries can participate in global online change. The unique ones cannot (or should not).

- Will resource consistency checking be active? For which libraries?

  If your online change libraries are not shared, then you need to turn off consistency checking by specifying NORSRCC=(libraries not to be checked). Note that this parameter identifies libraries for which consistency checking is NOT done, not those for which consistency checking IS done.

- Who will be authorized to initiate and terminate global online change? How will it be enforced?

  Although this is really a security issue, you need to determine who will have the authority to issue the online change commands:

  – INIT OLC
  – TERM OLC
  – QRY OLC

  INIT and TERM are especially important since they impact operation of the IMS systems. QRY is just a query command and so may be authorized to more users. This will require that security checking be performed by the Operations Manager by coding the CMDSEC= parameter in the OM initialization PROCLIB member (CSLOIxxx). You have a choice of using a security product like RACF, or a user exit defined in the user exit list for the OM component, or both.

- Will FRCNRML and FRCABEND be allowed? Under what circumstances? For which type? Who can make the decision?

  FRCNRML and FRCABEND allow the global online change process to continue even though some of the IMSs are not available. Care must be taken when using these parameters. If an IMS system misses one global online change process, it usually has to be cold started. The exception is if the global online change was for the FORMAT library only, in which case a warm or emergency restart is valid. If an IMS system misses more than one global online change process, then a cold start will always be required. The migration plan should include a decision as to the circumstances under which these parameters may be used, and who may use them.

  These parameters cannot be checked directly by RACF. If you want the INIT OLC command with either of these parameters to be further restricted (beyond what the INIT OLC and TERM OLC commands are), then an OM User Security Exit will need to be written to examine the entire command and make the final decision. At this stage of the planning, you merely need to determine whether FRCNRML and FRCABEND are to be treated differently from a security point of view.

5.3.3 Sysplex terminal management

Sysplex terminal management (STM) is a function of the IMS control region and the Resource Manager, and requires both a resource structure in the Coupling Facility and shared queues. Its primary purpose is to guarantee consistent definition of resources across the IMSplex, to guarantee that resources are active on only one IMS at a time, and to be able to recover terminal user status information following a session or IMS failure.

- Will sysplex terminal management be enabled?

  If the answer to this is no, skip to the next section.

- Will all the IMSs be version 8 before STM is enabled? Which IMSs will the end-users log on to? Can they switch from version 6 or version 7 to version 8 or vice versa?

  It can be confusing for the end-user to switch back and forth from IMS Version 6 or IMS Version 7 to IMS Version 8 if sysplex terminal management is active. Since IMS Version 8 is the only release to support CSL and STM, users would have to recognize that sometimes their status would be recovered, and sometimes not. Even more confusing might be when a user with significant status goes from version 8 to version 7 (no status recovery) then back to version 8. IMS Version 8 would recover the status that user had at the time the user last terminated his/her session on IMS Version 8. If you do have a mixture, you may want to consider using only IMS Version 8 as the front-end and using the older releases strictly as back-end processors.

- What types of terminals can connect to the IMSplex?

  Different terminal types are treated differently by sysplex terminal management. Some terminal types are not supported at all.

  – VTAM? OTMA? BTAM?

    STM supports only VTAM terminals. BTAM and OTMA are not supported.

  – STSN?

    STSN sequence numbers are considered significant status. This means that if SRM=GLOBAL, and RCVYSTSN=YES, the resource structure will be updated for each input and output message from STSN devices. Some users may decide that STSN recovery is not required globally, and set SRM=LOCAL for STSN terminals. This must be done in the Logon Exit (DFSLGNX0).

    Other users may decide that STSN recovery is not required at all, in which case they can set RCVYSTSN=NO in DFSDCxxx. This may be desirable when ETO is used for STSN devices and the STSN significant status keeps the control block structure from being deleted when the session is logged off.

  – APPC?

    There is only limited support for APPC by STM. The only support is for resource name consistency between APPC descriptor names and other resource names including transaction codes, LTERM names, and MSNAMEs. For planning purposes, you must ensure that these APPC descriptor names do not conflict with other resource names. This is probably not a problem you currently have if you are running on a single IMS. It also probably is not a problem you would have if your IMSplex IMSs are clones. But if they are not clones, it might be possible that these name conflict.

  – ISC?

    With single session ISC, the NODE is always unique within the IMSplex because the STM uniqueness requirement does not allow the same NODE to be active on more than one IMS.

    Care must be used when providing STM support for parallel session ISC. STM does not enforce uniqueness for NODE names since there are parallel sessions from the same NODE. However, STM will enforce USER (SUBPOOL) and LTERM uniqueness within the IMSplex. When IMSs are cloned, or at least have the same ISC definitions, there is a definite possibility that, if the same parallel session NODE logs on to more than one IMS, that each IMS may assign the same SUBPOOL and LTERM name to that NODE. Since this would violate the uniqueness requirement, the second logon would be rejected.

    There are different solutions to this problems for static and dynamic.

    • ***For static ISC sessions, single or parallel***, the user may use the IMS Initialization Exit (DFSINTX0) to tell IMS that it does not want to maintain any information at all in the resource structure. This would allow the same NODEs (including single session NODEs), USERs, and LTERMs to be active in each IMS. However, if an LTERM is active on multiple IMSs in a shared queues group, each IMS would register interest in that LTERM and deliver a message to whichever ISC NODE that LTERM was assigned to. This may or may not be a problem in your environment. If it really is the same NODE, then perhaps you will not care.

Since no entries are made in the resource structure, there can be no global recovery. Local recovery (or none) is still available. If you specify global, then IMS will change it to local.

Another approach for static ISC sessions would be to define each IMS with different ISC SUBPOOL and LTERM definitions. This would guarantee they are unique, but would limit the recoverability when one IMS fails and the NODE logs on to another IMS (which would assign a different SUBPOOL and LTERM name).

- ***The above capability does not apply to ETO ISC sessions.*** STM will continue to enforce resource consistency and uniqueness. To address this problem for ETO, the user can write a Signon Exit (DFSSGNX0) which would select a USER and LTERM name which could be unique to each IMS. Or the Signon Exit could use global callable services (control block services) to determine whether a particular USER (SPQB) or LTERM (CNT) is already defined in the IMSplex.

The point of all the ISC discussion is that ISC does present some particularly complex problems when sysplex terminal management is active and you have either static or dynamic (ETO) ISC sessions. And some of the solutions present new problems for static single session ISC.

- Are there any MSC sessions between IMSs in the IMSplex and remote IMSs?

  The only support for MSC provided by STM is resource type consistency. You will be prevented from defining an MSNAME which is also an LTERM, transaction, or APPC user descriptor name. Keep in mind that MSNAMEs, once they are defined to the resource structure, are never deleted. This is a problem only if you have a name defined as an MSNAME and then later decide you want to use that name as an LTERM name instead (probably not likely).

- Will ETO be enabled?

  Except for parallel session ISC, users will not be able to log on to multiple IMSs from the same NODE. However, if ETO is enabled, USER names (SPQBs) and LTERM names (CNTs) are created during the signon process. This will cause entries in the resource structure to be created, at which time resource name uniqueness would cause the logon/signon to fail if that resource is already active on another system. It is important that each IMS assign unique USER and LTERM names to its ETO sessions. There are a number of ways to do this with ETO. For example, the USER name and LTERM name can be set to the NODE name. Except for parallel session ISC, this would guarantee uniqueness, since the NODE could not be active on multiple IMSs at the same time.

  A Signon Exit (DFSSGNX0) could be written which is unique to each IMS and would assign USER and LTERM names which are unique.

- What types of end-user significant status will exist within the IMSplex? Conversational? STSN? Fast Path? How many terminals or transactions? What will be the transaction rates for end-user status updates to RM structure?

  You should understand the implications of end-user significant status. Each of them (conversational, STSN, and Fast Path response mode) requires multiple updates to the resource structure for each transaction. However, considering the Coupling Facility accesses already required for data sharing and shared queues, this should not be a significant increase in overhead, but you should be aware of it for structure placement purposes.

  When a session fails or when IMS (or IMS's platform) fails, the user needs to know how to get back online. This process could be different for the various values of SRM, and depending on whether or not the user has significant status, whether it is important that one be able to recovery that status. The process should be documented and end-users trained on the process they should follow. They also need to know how to bypass status recovery when necessary to get back online. For example, when SRM=LOCAL, if IMS fails while the user has end-user significant status, the RM affinity in the structure will prevent that user from logging on to another IMS. If the original IMS is going to be down for an extended period, the user must know how to force another IMS to accept the logon request. This can be done by coding the Logon Exit (or Signon Exit for ETO) to recognize user data in the logon request as a request to “steal” the NODE.

- What will the system default for SRM and RCVYx be? Will the Logon or Signon Exit override SRM and RCVYx for some terminals? Which ones? Will ETO descriptors specify SRM and RCVYx?

  IMS has its own defaults for SRM. You can specify a different default in DFSDCxxx. For ETO, you can override the system default on the user descriptor. You can override these defaults in the Logon Exit or Signon Exit if different terminals/users are to have different values for SRM. These decisions need to be made, although it will probably be sufficient to let the system default as specified in DFSDCxxx apply to all terminals. What might be worth overriding is the RCVYx parameter, for example, to turn off STSN recovery (RCVYSTSN=N) in DFSDCxxx.

- Will single signon be enforced within the IMSplex?

  Sysplex terminal management can be used to enforce single signon for a user across the entire IMSplex. Since single signon is the default, you do not need to do anything to get this capability. When a user signs on, a USERID entry is made in the resource structure which prevents that user from signing on again from another NODE. If you do NOT want single signon, then you must code SGM=M (or other similar parameter) in DFSPBxxx. It is important to note that the first IMS to join the IMSplex sets this value. Later IMSs, even if their SGN value is different, cannot override it. It can only be changed by shutting down all IMSs in the IMSplex and then starting up again with the correct value. So, be sure you get it right the first time.

- Is there an Output Creation Exit? If so, does it need to be updated? If not, is one needed?

  When a message arrives in IMS, and the destination field of that message is not defined locally, IMS will query the Resource Manager to determine whether that name is known somewhere else in the IMSplex. If it is found to be an LTERM defined by some other IMS, then the front-end IMS will enqueue it on the LTERM Ready Queue in the shared queues structure. If it is found by RM to be a transaction, then the Output Creation Exit is called and with a flag indicating that this unknown destination is defined in the resource structure as a transaction. The exit has several choices. It can allow the transaction to be queued with the default characteristics (may not be correct), it can set transaction characteristics (if it knows what they should be, but the exit probably does not), or it can reject the transaction.

  We think the last alternative is best. If a transaction is not defined locally, it should be rejected. If it is important for it to be defined locally, then online change should be used to add it with the proper attributes (for example, Fast Path, conversational, response mode, single segment, etc.).

- What will a user do if IMS fails while one is logged on? Wait for ERE? How long? Will the user be allowed to override RM affinities? Under what circumstances? Does the Logon or Signon Exit have to be updated to allow stealing?

  This is related to the above question about understanding what kind of significant status will exist within the IMSplex environment, and what users need to do to recover from session or IMS failures. If SRM=LOCAL, a Logon or Signon Exit will probably be needed to allow the user to override the RM affinity and let a new IMS “steal” the NODE.

- How will the RM structure be defined? Duplexed? Autoalter? Size (max, init, min)? Where? Is SMREBLD and SMDUPLEX allowed by OS/390 or z/OS systems? Who can make decisions about structure rebuild?

The structure must be defined in the CFRM policy. In that policy, you may specify several sizes, including an INITSIZE (initial allocation size), a SIZE (maximum size to which the structure can be altered), and a MINSIZE (minimum size to which the structure can be altered. A tool is available to help you size the structure at:

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

You may also allow the system to *automatically alter* the size of the structure when it either reaches a full threshold (the system alters the structure size larger) or the system needs the space and the structure itself has lots of unused space (the system alters the structure size smaller, but not below the minimum size). This is enabled by specify in the policy:

```
ALLOWAUTOALT(YES)
FULLTHRESHOLD(nn%) <<< the default is 80%
```

The resource structure contains a lot of information about the IMSplex and especially about the terminals and users in that IMSplex. It is not a disaster if it is lost. CQS will request RM to repopulate a new structure with RM-type information, and RM will request each client IMS to repopulate the structure with terminal/user recovery information. This may not be complete. For example, if an IMS is down at the time the structure is repopulated, then its information will, of course, not be repopulated. Duplexing the structure lowers the probability that this will happen, since it would take two failures for the structure to be lost. Duplexing does mean, however, that there is twice as much activity to the Coupling Facility for updates or queries to the resource structure. This may still not be a problem, since resource structure activity is probably low on the scale compared to the lock structure and the cache structures. Duplexing is invoked by defining in the policy:

```
DUPLEX(ALLOWED) -or-
DUPLEX(ENABLED)
```

Both autoalter and duplexing require that the CFRM CDS be updated with:

```
ITEM NAME(SMREBLD)
ITEM NAME(SMDUPLEX)
```

More information on how to do this can be found in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929.

- What is the peak RM structure update rates expected? Will it be an issue?

  As discussed above, we do not expect the activity to the resource structure to be significant enough to have a negative effect on performance, however, in the event that the Coupling Facilities or links are excessively busy with other work, this could be a factor and it is best to understand what to expect. Resource structure activity is usually related to the updating of end-user significant status. Other activity should be minimal.

- Do you expect any problems with resource type consistency or resource name uniqueness? How will you resolve it?

  If you are combining several independent IMSs into a single IMSplex, then it is possible that you might have resource type consistency or resource name uniqueness problems. Even if you are coming from an existing data sharing / shared queues environment, uniqueness was not enforced and you may have intentionally allowed the same resource name to be active on multiple IMSs. For example, you might have defined two printer NODEs with the same LTERM name on two different systems to improve throughput for that LTERM destination. This will not work with STM and an alternative must be found.

- Do any exits use callable services for sysplex terminal control blocks (VTCBs, SPQBs, CNTs)? Do any exits use callable services to find CCBs? Will the version 8 default of GLOBAL cause these exits any problems? Do they have to be updated?

  Callable services has been updated in IMS Version 8 to support *global* FIND and SCAN processing. When a resource is not found locally, or is found but is not active locally, callable services will query RM to see if it exists or is active somewhere else in the IMSplex by looking in the resource structure. If it is found in the structure, then control blocks are built representing that resource and passed to the exit. The default for this is global, but the exit can request that only local resources be searched. You should examine your exits which use callable services to determine whether or not they need to be updated.

5.3.4 Automated operations using OM interface

When running IMS with a Common Service Layer, commands may be sent to, and responses received from, IMSs in the IMSplex. This may be done by implementing an *Operations Manager client* - a program which interfaces with OM to send commands and receive responses from any or all IMSs.

In one respect this may be thought of as a *single point of control*. In reality, what it really is in IMS Version 8 is a *single point of command entry and response.* At this time (IMS Version 8) it cannot completely control the IMSplex since it does not receive messages originating within IMS and reporting, for example, a problem. An example would be the DFS554A message indicating that an application abend has occurred and that a PSB and transaction code have been stopped. Many users would like to automatically restart the PSB (program) and transaction code when this message is sent by IMS. Most automation tools available in the IMS environment can do this. However, this message would not be sent to the OM client.

It is, however, possible to capture those messages through other means. For example, NetView could detect the message and invoke a REXX exec which could then issue commands to start the stopped resources. Many variations of this technique are possible.

During this functional planning phase, you should consider the ways in which you intend to exploit this OM interface to the IMSplex. Note that you may decide at any time later whether to add new OM clients, but you should probably plan on at least installing and using the TSO SPOC program which comes with IMS Version 8.

TSO Single Point of Control (SPOC)

The TSO SPOC is provided with the IMS Version 8 product. It runs as an ISPF application under a TSO session.

- Will the TSO SPOC be used to control IMSplex operations?

  If the answer is no, skip to the next step.

- How many SPOCs will be started?

  Although the title of this program is *TSO **Single** Point of Control*, you can have as many of them as you want. Each user may start one or more SPOCs in his/her TSO session. For example, with the split screen support, you could have two SPOCs and toggle back and forth. This might be useful if that TSO user wanted SPOC sessions with more than one IMSplex.

- Who are the authorized users? How will it be controlled?

  OM security checking is done based on the TSO user ID. You must decide who will be allowed to do the following:

  – Register with SCI

    The TSO SPOC must register with SCI before it can do anything. You may define the IMSPLEX in the FACILITY class of RACF (or equivalent) and limit which user IDs are permitted to join the IMSplex (register with SCI).

  – Enter IMS commands

The Operations Manager has the capability to authorize (or reject) users from entering both IMS classic commands (those beginning with a slash) as well as the new IMSplex commands (INIT, TERM, UPD, DEL, and QRY). You must decide who is authorized to enter which commands and then set up your command security procedures accordingly.

- Will you create a standard set of “shortcuts?”

  The TSO SPOC gives you the ability to create “shortcuts” for commands. A shortcut is a shorthand notation for a longer command. For example, DBRACCTG might be set up as a shortcut for a /DBR DB command that would DBR all the accounting databases. STAACCTG might START all the accounting databases. Each user can set up his/her own shortcuts, but you may want to create a set of *standard shortcuts* which all (authorized) users would have.

IMS Control Center

The DB2 Admin Client is a DB2 tool which has been enhanced to include a control center for IMS. This IMS Control Center runs on a workstations provides a GUI interface for the user to enter IMS commands to the OM interface and receive responses. The tool itself communicates with IMS Connect across a TCP/IP connection. IMS Connect registers with the SCI as an AOP member of the IMSplex and forwards commands from the workstation to OM and forwards responses from OM (which got them from IMS) to the workstation.

- Will you use the IMS Control Center tool to control IMSplex operations?

  If the answer to this is no, then skip to the next section.

- Other questions similar to TSO SPOC

  If you are using this tool, then you must answer many of the same questions you did for the TSO SPOC. That is, who (what user IDs) will be allowed to enter which commands. Security must be established in OM the same way as for other OM clients.

IMS Control Center is a part of the DB2 Admin Client that you can download from the DB2 Web site for free:

  `http://www.ibm.com/db2`

You can find the details for the prerequisites and how to set up the IMS Control Center from the following Web site:

  `http://www.ibm.com/ims/imscc`

User- and vendor-written automation

The interface to the Common Service Layer is open and documented in *IMS Version 8: Common Service Layer Guide and Reference,* SC27-1293. You may write your own OM clients to provide for IMSplex automation, or other software vendors may write automation tools which use the OM interface.

In addition, IMS Version 8 provides REXX commands, functions, and variables to allow you to write REXX execs which register with SCI and send commands to the Operations Manager. These REXX execs can be executed as batch programs, they can execute under TSO, or they can be invoked dynamically by a program such as NetView.

- Will existing automation programs and execs continue to run in an IMSplex?

  If you already have some IMS automation programs running in your system, you should examine them to see if they will continue to work as needed in an IMSplex. If you are already running in a Parallel Sysplex with data sharing and/or shared queues, then it is likely that your automation programs will continue to run as you expect. If you are currently running in a single IMS environment, then your automation programs need to be evaluated to see if they still work as required.

- Do any of them need to be converted to use the OM interface?

  If you want these automation programs to have access to all the IMSs in the IMSplex, then they will have to be converted to use the OM interface. If they are vendor products, you should check with the vendors to see if they have plans to take advantage of this environment.

- Are any new automation execs needed? Who will write them? Will they be provided by vendors?

  Now that you are going to be running with a different environment, including three new address space types and probably several of each, plus an optional new structure in the Coupling Facility, you may want to consider additional automation. For example, you may want an automation script to start up the CSL address spaces in the correct order, or to detect the failure of one of the address spaces and restart it (although we recommend ARM for this). When any of the address spaces are started, if they try to connect to one that is not there, an error message will be issued. This could be the trigger for an automation program to start the missing address space.

Automation and the OM interface in general

Every AOP-type address space must be able to register with a local copy of SCI. This is true even if that AOP address space is running on an OS/390 or z/OS image that has no other IMSplex components. For example, you may have an OS/390 or z/OS image which your TSO users use, but which does not have any other IMSplex components. If you want one or more of those TSO users to be able to start the TSO SPOC, there must be an SCI address space on that OS/390 or z/OS image. Note that it must still be part of the same Parallel Sysplex.

It is possible for you to have multiple IMSs which do not share data and do not share message queues to define themselves as part of the same IMSplex. If this is done, then a SPOC or other AOP program could join the same IMSplex and be used to control all of the IMS that are otherwise unrelated.

5.4 Configuration planning

Although there is a separate chapter that addresses configuration (Chapter 7, “Putting it all together” on page 159), because CSL is so new to the IMS Parallel Sysplex environment, we will address it in a bit more detail here than later. This part of the planning process addresses the high level view of your IMSplex, the configuration. You can think of it as the framework within which you will build your IMSplex. It requires answering a series of questions about the existing and desired configuration.

- What IMSplex configuration is required to meet your objectives?

  As always, you should refer back to your objectives to answer this question. After answering each subquestion, you should be able to diagram your IMSplex, at least at a high level.

  – How many IMS control regions will you have in your IMSplex?

    The number of control regions is usually related to capacity, availability, or both. It may also be determined by the applications themselves. For example, while it is nice to be able to clone all your IMS systems and run a single defined image multiple places, it may be necessary, or desirable, to NOT clone then. Some may need to be unique, or at least be not identical to the others.

  – Will all your IMS control regions be IMS Version 8?

The IMSplex can consist of IMS Version 6, IMS Version 7, and IMS Version 8 control regions. They can all share the message queues and the databases, but only IMS Version 8 can have a CSL.

– What LPARs will be running IMS Version 8 control regions?

  Assuming that a Parallel Sysplex is already in place, identify the LPARs in that sysplex on which you will run your IMS control regions. There should, of course, be enough LPARs to meet the IMS control region requirements for availability and performance. It may be necessary to create additional LPARs.

– What LPARs will be running batch or utilities with DBRC?

  You may have some LPARs which do not run your IMS control regions, but do run IMS batch jobs or IMS utilities which use DBRC to access the same set of RECONs used by the control regions. The batch jobs may even be sharing data with the control regions. If you are running with a mixed environment of IMS Version 8 with IMS Version 6 and/or IMS Version 7, you may want to consider letting all of your batch jobs and utilities execute with IMS Version 8 code. This will enable them to participate in automatic RECON loss notification. If some are not running with IMS Version 8, then ARLN will work properly for all the IMS Version 8 members, but will be ignored for previous releases. This is not a problem, it just negates some of the advantages of ARLN.

– How will you take advantage of the OM interface to enter commands to IMSs in the IMSplex?

  The OM interface allows for several ways to manage the IMSplex through command entry and response. While you may not know at this stage which ones, or how many, you will use, you may want to plan for their eventual implementation. The options you have are identified below. There must be an SCI address space on each LPAR where any of these are executing.

  - TSO SPOC

    This is an ISPF application provided with the IMS Version 8 product which can be started on an authorized TSO session, connect to the IMSplex through SCI, and then issue commands and get consolidated responses similar to what you would get from the MTO terminal.

  - IMS Control Center (through IMS Connect)

    The IMS Control Center is a component of the DB2 Admin Client and provides a graphical user interface (GUI) from which to enter IMS commands to the IMSplex. It runs in a workstation environment and uses a TCP/IP connection to IMS Connect to access the IMSplex. IMS Connect registers with SCI and then forwards commands to OM and returns responses to the workstation. Like the TSO SPOC, there can be multiple IMS Control Centers in the IMSplex.

  - User-written or vendor-written automation programs or execs

    In addition to the IBM provided TSO SPOC and IMS Control Center, the user may implement user-written or vendor-written automation programs. These programs may use the published SCI and OM interface macros, or they may use the IBM-provided REXX OM interface to send commands and receive responses. For example, NetView may invoke a user written REXX exec to issue commands to the IMSplex based on DFS™ messages it receives.

– How many SCIs will be needed? Where?

  Now that you know how many and which LPARs will be running control regions, batch or utility jobs, or programs using the OM interface (including IMS Connect), you can determine the number of SCI address spaces you will need. Each one will require execution JCL and an initialization PROCLIB member (CSLSIxxx).

– How many OMs will be started? Where?

  While each LPAR requires an SCI address space, only one Operations Manager (OM) is required in the IMSplex. At least two are recommended for availability, and there can be one on every LPAR in the IMSplex. It is not likely that multiple OMs would be required for performance reasons since it is not likely that command entry will represent a very large workload. If you do not plan to have an OM address space on every LPAR, then you need to decide which LPARs they will be started on.

– Will a resource structure be required?

  The resource structure is a list structure in the Coupling Facility that is required to support sysplex terminal management. It is not required for any other CSL function, although global online change will make use of it for resource consistency checking if it exists. At this stage of planning, you need only decide whether or not you will be using STM and therefore require a resource structure. Note that if there is no structure, then only one RM address space is allowed.

– How many RMs will be started? Where?

  While each LPAR requires an SCI address space, only one Resource Manager (RM) address space is required in the IMSplex. At least two are recommended for availability. Since the RM may access the resource structure multiple times for each transaction, additional RMs may be advisable for performance reasons as well. As noted above, if there is no resource structure, then you can have only one RM in your IMSplex.

– Will CQS be used to support both shared queues and resource management?

  The IMS Version 8 Common Queue Server (CQS) can be used by the IMS Version 8 control region to support shared queues, and by the RM to support STM. Unlike RM, there must be a CQS on each LPAR where there is either a control region using shared queues, or a Resource Manager using the resource structure. If there is both a control region and a RM on the same LPAR, then only one CQS is required on that LPAR, although you may decide to have one for shared queues and one for RM.

  The IMS Version 8 CQS does not, however, support previous versions of IMS for shared queues. So, if you have an IMS Version 6 or IMS Version 7 control region on any LPAR, you will need an IMS Version 7 CQS also on that LPAR to support shared queues. You may have both an IMS Version 8 and an IMS Version 7 control region on the same LPAR, in which case you would require both a version 7 and a version 8 CQS.

- Will VTAM Generic Resources be used?

  IMS support for VTAM Generic Resources (VGR) was introduced in IMS Version 6. The IMSs could join a VTAM generic resource group to make logging on to the IMSplex easier for the end-user by using a generic resource name instead of the IMS APPLID. In your initial implementation, you may choose to use VTAM Generic Resources as a means of distributing the online workload to multiple IMSs in the IMSplex before implementing shared queues. If this is the case, then there are no CSL issues related to VGR. When shared queues are used with CSL, then the use of VGR does have some impact on sysplex terminal management.

  Beginning with z/OS 1.2, VTAM supports session level affinities which improve VTAM generic resource affinity management in the IMSplex by allowing IMS to set the affinity management to either IMS or VTAM for each VTAM session. If you are running at less than z/OS 1.2, you will need to specify an IMS parameter (GRAFFIN) to determine who will manage the affinities.

- How will the IMSplex implementation be phased in?

  If you are not currently using any Parallel Sysplex functionality in your IMS environment, you may want to consider phasing into it. The first step would (probably) be data sharing. Until data sharing is implemented, distributing the transaction (or BMP) workload around the IMSplex would not be practical.

  CSL could either be the next step after data sharing, before shared queues, if you want to take advantage of Operations Manager and single point of control capabilities, global online change, and/or automatic RECON loss notification, without needing sysplex terminal management (which requires shared queues). Alternatively, you could implement shared queues first and then follow up with CSL. If you are currently on IMS Version 6 or Version 7, then of course you cannot implement CSL until you get to IMS Version 8.

  – Is data sharing already implemented?

    This should be the first step in your migration. Refer to Chapter 3, “Planning for block level data sharing” on page 55 for planning information about data sharing.

  – Will shared queues be implemented? Is it already implemented?

    This could be the next step, or after CSL. Refer to Chapter 4, “Shared queues planning considerations” on page 77 for planning information about shared queues.

  – Will there be IMS Version 6 and/or IMS Version 7 systems in the IMSplex? For how long?

    Remember that IMS Version 6 and Version 7 can participate in data sharing and shared queues with IMS Version 8, but they cannot participate in any of the CSL functions. If CSL is implemented in this mixed environment, care must be taken not to confuse the end-user. For example, if the end-user is logged on to IMS Version 8 (for example, using VTAM Generic Resources), and then IMS Version 8 crashes and the user logs back on to IMS Version 7, any terminal/user status information will not be recovered. It will, in fact, remain in the resource structure (assuming you have one). When that user later logs on to an IMS Version 8 system, it will try to recover the status from the resource structure.

    So, if you are running in a mixed environment, you may want to consider not using a resource structure until everybody is on IMS Version 8. Alternatively, you may allow users to only log on to IMS Version 8 and use IMS Version 6 and IMS Version 7 as back-end systems only. This would, of course, reduce the availability benefits of multiple IMSs in an IMSplex.

  – Will all CSL functions be implemented at once? If not, what sequence and when?

    CSL does not have to include all of the functions available. Depending on your objectives, defined earlier in the planning process, you may decide to either delay implementation of some functionality, or just to not implement it at all. CSL functions for which decisions should be made include:

    - OM interface to IMS systems using SPOC, IMS Control Center, and/or user-written and vendor-written automation programs. This feature does not require a resource structure, but if one is available, some commands having global impact will update the structure.

    - Global online change also does not need the resource structure. However, if one exists, it may use it for online change data set resource consistency checking. Each IMS can independently decide whether or not to participate in global online change. However, changing from LOCAL to GLOBAL or vice versa requires an IMS cold start.

    - Automatic RECON loss notification requires ONLY an SCI address space. Other CSL address spaces are not required. It does require that participating DBRCs either specify the IMSPLEX name as an execution parameter, or that the DBRC SCI Registration Exit (DSPSCIX0) be coded and available to DBRC.

  - Sysplex terminal management is the only function that requires both shared queues and a resource structure. You may want to consider easing into CSL by implementing the other functions before defining a resource structure. Note, however, that adding a resource structure later on requires an IMS cold start.

- Will BPE PROCLIB members be unique or shared?

  Four address space types run on top of the Base Primitive Environment (BPE). These are CQS, SCI, OM, and RM. Two PROCLIB members define the BPE environment for each of these address space types. Each of these two members can be either unique to the address space, or shared across all address spaces.

  – BPE configuration (BPECFG)

    Each of these address spaces has a BPE configuration PROCLIB member that describes the BPE component of the address space. It identifies, for example, trace levels and user exits. All parameters have defaults and it is not even necessary to have a BPE configuration member.

  – BPE user exit list (EXITMBR in BPECFG)

    The BPE configuration member identifies for each component, a user exit list member. The user exit list identifies user exits that are to be invoked by BPE on behalf of that component (BPE, CQS, SCI, OM, RM).

- Are any user exits needed?

  If user exits are required, they must be included in the BPE user exit list PROCLIB member (the BPE configuration PROCLIB member points to the exit list member). *None of the BPE exits are required,* although the user may elect to code one or more of them based on the installation's requirements. CSL exit interfaces are documented in the *IMS Version 8: Common Service Layer Guide and Reference,* SC27-1293.

  Two of the exits that may be defined in the BPE user exit list member are BPE exits. The use of these exits, and the interface, is documented in *IMS Version 8: Base Primitive Environment Guide and Reference,* SC27-1290. You should review these exits and decide whether you need them. Our guess is that you will not.

  – BPE Initialization and Termination Exit?

    These exits get invoked late during BPE initialization in the address space and earlier during BPE termination. The termination exit does not get invoked if the address space abends. It is not clear what the user would use these exits for, but the initialization exit may, perhaps, be used to load a table that would be used later by other exits, such as the security exit. The termination exit could then delete the table.

  – BPE Statistics Exit?

    This exit, when it exists, can be used to gather BPE statistics. The exit gets driven based on a user specified statistics control interval.

  Note that each component (OM, RM, and SCI) also supports these two exits. Other user exits apply to particular components. For example, the Input Exit applies only to the Operations Manager. These other user exits are documented in *IMS Version 8: Common Service Layer Guide and Reference,* SC27-1293. Review each of these exits and decide whether or not you need them. Of all possible exits, the ones most likely to be useful are OM exits.

  – OM Security Exits

    The Security Exit is similar to the IMS Command Authorization Exit (DFSCCMD0) and gets invoked on all commands entered through the OM interface. In gets invoked after

RACF (if you are using RACF or an equivalent product) and can override RACF's decision.

– OM Input Exit?

  The Input Exit is invoked on all input from the OM client and can edit it, reject it, or let it go unaltered. For example, an Input Exit could recognize a certain "command" and alter it. The "command" does not even have to be a valid command as long as the exit makes it valid before passing it along. Not all forms of a classic IMS command are accepted by OM. That is, IMS does not register all forms of a command with OM. For example, IMS does not register /DISP with OM. If a user enters /DISP, OM would not pass it along to IMS. The exit could alter the command to the correct form. The Input Exit gets invoked before the Security Exit.

– OM Output Exit?

  The Output Exit gets invoked on all messages from the command processors and can edit the output prior to OM forwarding it to the client. The Output Exit also gets invoked for unsolicited output, or whatever OM thinks is unsolicited, such as a late response. At this time, there is no supported *action* that the output exit can take, such as forwarding an unsolicited or undeliverable message to another destination. Any such action would have to be coded entirely by the exit. There are no examples of how to do this.

– Any other component exits required?

  BPE supports a number of other exits which are documented in the *IMS Version 8: Common Service Layer Guide and Reference,* SC27-1293. It is not likely that any of these would be needed initially, although you may identify a need at some later time.

► Will ARM be used to restart CSL address spaces?

  Most IMSplex address spaces, including SCI, OM, and RM, support the Automatic Restart Manager (ARM). ARM can be used to quickly restart any supported address space in the event of an address space abend, an operating system failure, or a processor failure. ARM is activated in the Parallel Sysplex by creating an ARM policy in the ARM CDS, defining the various *elements* in that policy, and then *activating the policy* using the SETXCF command.

```
SETXCF START,POLICY,TYPE=ARM,POLNAME=policyname
```

  When ARM is active in the sysplex, *ARM restart is the default* for these address spaces. It can be disabled by coding an execution parameter (ARMRST=N). Because ARM restart applies to many of the IMSplex address spaces, planning considerations for ARM are addressed separately in 2.5.2, "Automatic Restart Manager (ARM)" on page 20.

  The following paragraphs on single points of failure provide further discussion of how and when to restart the CSL address spaces.

► Are there any single points of failure?

  Since availability is generally the highest priority in any IMSplex, and fault tolerance is one of the chief benefits of an IMSplex, you should examine your IMSplex configuration and *identify any single points of failure*. This is a necessary task for each of the major planning categories (block level data sharing, shared queues, and CSL), and is addressed in each corresponding planning chapter. For CSL itself, the potential failure points are:

– SCI, OM, and RM address spaces

  There must be an active SCI address space on each OS/390 or z/OS image in the IMSplex. There must be at least one RM and one OM address space available in the IMSplex. The unavailability of any of these (that is, if SCI is not available locally and/or if OM and RM are not available anywhere in the IMSplex), will impact the IMSplex in one or more of the following ways, depending on whether a resource structure exists. If there is a resource structure:

- Any attempt to update the structure will fail if either SCI or RM are not available.
- Logons and signons will be rejected. Note that this applies even if SRM is not set to GLOBAL; resource entries are always created for command status.
- Normal activities (for example, conversations, STSN, and Fast Path transaction processing) will continue with the status being maintained locally. When the failed component is again available, the status will be updated in the resource structure (as required).
- Logoffs and signoffs will wait until SCI, RM, and the resource structure are available.

Your IMSplex configuration must contain at least one of each of these address spaces, probably (should) contain at least two of each, and may contain one on each OS/390 or z/OS image. Since each of these are supportable by ARM, it is highly recommended that they all be defined with the ARMRST=Y parameter (the default). There are some caveats:

- If there are multiple OMs and/or RMs in the IMSplex, then cross-system restart should not be enabled. There is no advantage of having multiple instances on single LPAR. The CSL instance on another LPAR can take over the work for a failed CSL member in the IMSplex.
- If there is only one OM or RM in the IMSplex, then both local and cross-system restart should be enabled. This would allow IMS access to OM and RM as quickly as possible. Make sure that the candidate system is not the same one that already has an OM or RM.
- Since there is already an SCI on every OS/390 or z/OS image, cross-system restart should not be enabled for SCI (unless it is being restarted on an OS/390 or z/OS image that does not currently have any IMSplex components). Wherever you move IMS, there is probably already an SCI address space.

– RM structure

The resource structure is optional, but if you have one, it will be used for both command significant status and, optionally, for end-user significant status. As mentioned above, if you have a resource structure defined, then if it is not available, it may prevent some activities from occurring. Although status will be kept locally while the resource structure is unavailable, if IMS were to fail, that status could not be recovered globally.

You may want to consider duplexing the resource structure using system-managed duplexing. For more information about the system-managed duplexing, refer to 2.7, “Other connection services” on page 42.

5.5 Configuration, security, implementation, and operations

Because these planning activities are so often closely related to the same activities for other IMSplex planning activities, they are discussed in their own chapter, Chapter 7, “Putting it all together” on page 159.

6

Planning for IMSplex connectivity

Before there was the IMSplex, there was a single IMS to provide user services. It was easy to determine to which IMS to connect:

- Each IMS SNA terminal connects to a single IMS. Users needing access to that IMS simply log on to that IMS's APPLID
- TCP/IP clients are routed to an IMS Connect from which there will be only a single valid IMS DATASTORE
- WebSphere MQ clients are always routed to the only IMS in the OTMA XCF group
- BMPs are routed to the single OS/390 image where the IMS identified in that BMP's IMSID is currently executing
- CICS and ODBA clients are always started on the same OS/390 image as the IMS DB Control region

To some extent, this is also true in a Parallel Sysplex. Users still connect to a single IMS, but it may be any one of several IMSs in the IMSplex. Whether or not that IMS can process *any* user-submitted transaction depends on the configuration and definitions of the individual IMSs. Because of this, there is a need to plan for a much more detailed and robust configuration for connectivity. The easiest configuration to manage is one where all of the IMSs are clones and all of the databases are shared. Any transaction can execute on any IMS, any BMP can run on any IMS, and any DRA or ODBA client can run anywhere there is an available IMS. When applications are partitioned, or when databases are not shared, then the work must be routed to an IMS where the applications and databases are available.

In this chapter we will address some of the considerations for connectivity to an IMSplex, and some of the products and tools available to you, including:

- VTAM network connectivity
- TCP/IP network connectivity
- Database connectivity
- BMP routing and scheduling

6.1 Introduction to IMS network connectivity

In this section, we address techniques by which end-users can enter transactions to an IMSplex and receive responses. Connectivity to IMS data through the database resource adapter (DRA) interface, or the open database access (ODBA) interface, are addressed in 6.4, “IMS database connectivity” on page 154.

In this chapter we examine some of the major elements that provide the foundations of the IMSplex communications infrastructure. In this context, we are talking about users submitting transactions to an IMS in the IMSplex. There are two types of networks that end-users may use to submit message traffic to, and receive messages from, an IMS in the IMSplex. In some cases, both types may be involved in the delivery of the message. Messages can also be submitted directly by application programs running in IMS dependent regions.

- VTAM SNA network

  This is the traditional method for sending messages to the IMS DC component. IMS itself is a VTAM logical unit (LU). Logon requests are routed to one IMS and all subsequent message traffic is routed through VTAM directly to IMS. End-users may be terminals, such as a 3270 (LU2) terminal or SNA printer (LU1), or programs themselves, such as a FINANCE (LU0) terminal or an ISC client (LU6.1). Whatever the end-user LU type is, that user is in session with (logged on to) a single IMS and every message goes to and from that IMS.

  – SLUTYPE2 (LU2, 3270, 3270 emulators)

    These are 3270 terminals or 3270 emulators and have been the most common means of accessing IMS through a VTAM network. They are sometimes called “dumb” terminals since they have no internal programs to make decisions. The users typically just logs on to whatever IMS APPLID they want to be connected to, and then works from there.

  – SLUTYPE1 (LU1, printers)

    These are SNA printers which do not (usually) log on to IMS. Instead IMS acquires a session with the printer either by the MTO issuing an /OPNDST NODE command or IMS itself issuing a VTAM SIMLOGON command when output is queued. The subject of printers is addressed later in this chapter.

  – SLUTYPEP and FINANCE (LU0, ATMs)

    These terminals are not dumb - they are intelligent. Most ATMs are of this type. They can be programmed to make decisions about what IMS to log on to. They also utilize VTAM set and test sequence numbers (STSN) command processing to synchronize input and output message traffic during session initialization.

  – ISC (LU6.1)

    Intersystem Communication (ISC) supports communications between IMS and any other application using VTAM LU6.1 protocols. Typically, these “other” applications are CICS regions (more common) or other IMS regions (less common). They may be any user-written or vendor-written applications which obey the LU6.1 protocols. This is the only VTAM protocol that supports parallel sessions between the remote node and IMS.

  – APPC (LU6.2)

    APPC clients are programs running on any platform, including S/390, with VTAM network connectivity to the IMS host. When APPC protocols are used, the end-user is in session with APPC/MVS - not IMS. APPC/MVS and APPC/IMS (IMS’s support for APPC communications) use XCF for message traffic between them.

– Multiple systems coupling (MSC)

  MSC is used strictly for IMS-to-IMS communications. It is supported by a private VTAM protocol or a channel-to-channel (CTC) protocol between IMSs. End-users are in session with one IMS, which may then use MSC links to send that message to another (remote) IMS and receive a response.

- TCP/IP network

  Open transaction manager access (OTMA) was introduced in IMS Version 5 to process messages destined for IMS from non-VTAM sources. The most common of these is a TCP/IP client which may be part of the installation’s private network, a public network such as the internet, or a combination of private and public networks. Rather than being in session with a single IMS as with VTAM, the end-user is connected to an IMS OTMA client, such as IMS Connect or WebSphere MQ. Like APPC, these OTMA clients use XCF for message traffic to and from IMS. There are several significant differences, however, between the OTMA connection and the APPC connection:

  – IMS does not have to be on the same OS/390 image as the OTMA client.
  – There is no concept of logging on to IMS. Depending on the network hardware and software, and the OTMA client itself, each individual message may be routed to any of the available IMSs in the IMSplex.

Other IMS components and services which participate in IMS message processing include the following. This is not intended to be a rigorous description of the actual IMS components - it is largely conceptual. Many of these interact with each other, and with other unnamed components and services, to accomplish the total processing requirements for a message:

- IMS DC component

  This is the component of the IMS control region which manages network traffic. In the case of VTAM, it consists of device dependent modules, which are specific to each of the supported SNA protocols (LU0, LU1, LU2, or LU6.1), and APPC/IMS which supports the XCF communications with APPC/MVS. For non-VTAM network traffic, it includes OTMA which supports the XCF communications with the OTMA client.

- IMS queue manager

  The queue manager processes messages received by the DC component from the network or from local application programs inserting messages to a TP-PCB. Its function is to queue the message on a final destination such as a transaction (program) or a network destination. These destination queues can be local or, if shared queues are enabled, they may be global, physically residing in the shared queues list structures in a Coupling Facility.

  IMS has two queue managers: one for full function messages and one for Fast Path expedited message handler (EMH) messages. EMH messages are much less common that the full function message, and use different queuing techniques and different structures in a shared queues environment.

- IMS scheduler

  The IMS scheduler is responsible for scheduling IMS dependent regions to process messages which are queued on either local or global transaction queues. Like the queue managers, there are two schedulers: one for full function transactions and another for Fast Path EMH transactions.

- Common Queue Server (CQS)

  When shared queues are enabled, CQS acts as a server to IMS to manage the inserting and retrieving of messages to and from the shared queues structures. It supports both full function and Fast Path EMH messages. CQS also acts as a server to the Resource

Manager (RM) when running IMS Version 8 with a Common Service Layer (CSL) and a resource structure.

- Common Service Layer (CSL)

  CSL was made available in IMS Version 8 and provides systems management services to an IMSplex. One of its capabilities, sysplex terminal management, uses the Resource Manager with a resource structure in a shared queues environment to support cross-system status recovery for VTAM users with significant status (for example, IMS conversational status). Sysplex terminal management does not support transactions originating from TCP/IP or APPC clients.

Outboard of IMS are numerous hardware and software components and services which play a role in routing messages to and from IMS. Each of these will be described in varying amounts of detail later in this chapter, and include:

- Rapid Network Reconnect (RNR)

  IMS's RNR function invokes VTAM single-node or multi-node persistent session services to save VTAM session information either in a local data space (SNPS), or in a structure in the Coupling Facility (MNPS). It is used by IMS and VTAM to rapidly and automatically restore network connectivity following an IMS failure and emergency restart.

- VTAM Generic Resources (VGR)

  This VTAM service allows a VTAM user to use a generic name when logging on to the IMSplex. Use of the generic name allows VTAM to route the logon request to any available IMS in the generic resource group, relieving the end-user of the need to know specific IMS APPLIDs or knowing which IMSs might be available at any given time.

- Network Dispatcher - WebSphere Edge Server

  This software runs in a server on the “edge” of the TCP/IP network. It is conceptually the last component before the TCP/IP stack (in the TCP/IP address space) on an OS/390 host. It may route an inbound message to any available TCP/IP server application.

- Sysplex Distributor

  This component runs as one of the TCP/IP stacks in the sysplex and performs a function similar to that of the Network Dispatcher - that is, it can route an inbound message to any available TCP/IP server application.

- Telnet Server

  This program can run in the OS/390 host or on a distributed platform in the TCP/IP network. It is a TCP/IP standard application which receives inbound messages from a TCP/IP client and forwards the message to a VTAM application (for example, IMS) across a VTAM LU2 (3270) session.

- IMS Connect

  This TCP/IP server application runs in the Parallel Sysplex and is the destination component to which TCP/IP end-user clients can route input messages for IMS. IMS Connect can route incoming traffic to any IMS in the IMSplex which is a member of the same OTMA XCF group.

- WebSphere MQ

  This program connects to the network as an APPC (LU6.2) partner TP program or TCP/IP application. A message routed to WebSphere MQ uses XCF to queue the message for any MQ client (for example, IMS) in the Parallel Sysplex. MQ has two options for delivering messages to IMS. The MQ Bridge uses XCF and OTMA to put messages directly on the IMS message queues. The MQ Adapter uses an IMS “listener BMP” to trigger a transaction in IMS which then issues an MQ_GET call to retrieve the message from MQ, bypassing the IMS message queues.

The remainder of this section will address some of the planning considerations both for the native IMS network support and for the outboard connectivity components and services which you may want, or need, to include in your migration and implementation plans.

6.2 VTAM network connectivity

This section describes some of the special considerations when connecting to an IMSplex through VTAM. We only address the differences you should consider when configuring your network connectivity to an IMSplex versus a single IMS environment.

6.2.1 SLUTYPE2 (3270)

The typical 3270 user is instructed to log on to a specific IMS APPLID, then to sign on and begin submitting transactions. If that IMS fails or is shut down, the user waits for it to become available again, logs on again, and continues working. In an IMSplex environment with multiple IMSs acting as front-ends (IMSs with network connectivity), those users may have multiple IMS APPLIDs to which to log on and begin working. You have several choices about how to instruct that user to log on to the IMSplex:

- Continue to log on to a specific IMS APPLID, except do not tell all the users to log on to the same APPLID. Distribute the network responsibilities across all, or at least some, of the IMSs.
- Enable VTAM Generic Resources (VGR) and tell all users to log on to the generic resource name. VTAM will route the logon request to an available IMS. More information about VGR can be found in 6.2.8, "VTAM Generic Resources (VGR)" on page 141.

As with all users, the 3270 user will have to be trained on what action to take when the IMS one is logged on to fails, or is shut down. This may depend to some extent on that user's status at the time of the failure, and whether or not shared queues and sysplex terminal management are active.

6.2.2 SLUTYPE1 (printers)

Printers can require special consideration in an IMS Parallel Sysplex environment. It is likely that output to a particular printer can be generated by any IMS in the IMSplex, particularly if IMS systems are cloned. You may have the option of sharing or not sharing printers among all of the IMSs.

Shared printers

Statically defined printers can be shared by multiple IMSs in the IMSplex by specifying:

```
TERMINAL .....,OPTIONS=(SHARE,RELRQ)
```

ETO printers can be shared by coding the corresponding parameters on the logon and user descriptors:

```
L PRTNODEA ...,OPTIONS=(RELRQ) <logon descriptor>
U PRINTERA ...,AUTLGN=PRTNODEA <user descriptor>
```

With these options set, each IMS will attempt to log on to a printer whenever there is output queued. The RELRQ option tells IMS to release the printer (close the session) if another IMS requests it. The potential problem here is obvious - printer thrashing. You could spend a lot of time opening and closing printer sessions as they bounce back and forth between IMSs. A better option to consider might be to not share the printers. Then the issue becomes how to get the output messages to the IMS which owns the printer.

Non-shared printers

When messages for printers are generated in multiple IMSs in the IMSplex, but the printers themselves are owned by just one of the IMSs (that is, they are not shared), then the requirement becomes getting the output to the printer-owning IMS. There are several solutions to this requirement. The following are two suggested techniques.

- Non-shared queues environment

  Define all the printer LTERMs as remote MSC LTERMs and let each IMS send printer output to this remote (could be in the same data center) printer IMS. In this scenario, the printer IMS cannot be in the same shared queues group as the others, since MSC is not supported between members of a shared queues group. It can, of course, be in the same data center.

- Shared queues environment

  When using shared queues, code the SHARE option (or AUTLGN for ETO printers) but do not code the RELRQ option. This will allow the first IMS to successfully establish a session with the printer node. The others will try but will have to wait until the first IMS releases the printer (which it never will). In the meantime, while the other IMSs are waiting, the one with the active session will send all messages queued on the shared printer LTERM queue regardless of which IMS put the message on the queue. If that IMS fails, then the session will be released and one of the other IMSs will acquire the session and continue sending the messages.

6.2.3 SLUTYPEP, FINANCE, and ISC

SLUTYPEP, FINANCE, and ISC sessions all invoke STSN processing. STSN, an SNA command, is the mnemonic for Set and Test Sequence Numbers. When a SLUTYPEP device (for example, an ATM) or ISC application (for example, CICS) logs on to an IMS system, IMS sends a set and test sequence numbers (STSN) command to the terminal informing the terminal as to what IMS remembers the last input message and last output message sequence numbers to be (the sequence numbers are maintained within both IMS and the remote program).

The STSN terminal uses this information to determine if it has to resend the last input and to inform IMS whether IMS should resend the last recoverable output message. This STSN flow occurs whenever a STSN session is established with IMS, even if the session had been normally terminated the last time the terminal was connected to IMS.

There are several conditions where resynchronization cannot take place:

- When an IMS system is cold started (including a COLDCOMM restart), sequence numbers from the prior session are not available to IMS and IMS starts the session with sequence number zero (0) for both input and output.
- Sessions can be cold started by issuing the IMS command:

  ```
  /CHANGE NODE node-name COLDSESS
  ```

  This command was introduced in IMS Version 7 as a way to reset the sequence numbers to zero, allowing a cold session initiation without requiring an IMS cold start. This may be required when the terminal and IMS experience a non-recoverable attempt to warm start the session.
- When using VTAM Generic Resources with IMS, and VTAM is requested to manage affinities (GRAFFIN=VTAM specified as an IMS execution time parameter), sequence numbers are reset to zero by IMS at session termination or at IMS restart. They are not remembered by IMS when the session is reestablished.

Without IMS Version 8 running with the Common Service Layer, sysplex terminal management (STM), and global status recovery mode (SRM=GLOBAL) specified for the STSN device, once a STSN device is logged on to a particular IMS, only that IMS knows the sequence numbers. If it is important to the device to warm start the session (that is, to know the last sequence numbers), then that device must continue to log on to the same IMS with which its initial session was established. It it logs on to another IMS, that IMS knows nothing about the sequence numbers and will start the session cold with the sequence numbers set at zero.

With STM and SRM=GLOBAL, however, the sequence numbers are maintained in the resource structure in the Coupling Facility and are therefore available to any IMS for session initiation, allowing the user to log on to any IMS and continue with the same sequence numbers as of the prior session termination.

6.2.4 Intersystem communication (ISC)

ISC, or LU 6.1, is an SNA program-to-program protocol that allows IMS to communicate with "other" subsystems or programs. ISC is normally used to send transactions and replies back and forth between IMS and CICS subsystems, but could also be used, in lieu of MSC, for IMS to IMS communications. Since LU 6.1 is an externally defined SNA protocol, IMS really does not know what type of subsystem (CICS, IMS, other) with which it is in session. As far as IMS is concerned, the other subsystem is just another node.

There are several issues related to ISC in an IMSplex environment:

- STSN (already discussed in 6.2.3, "SLUTYPEP, FINANCE, and ISC" on page 136)
- Multiple IMSs
- Parallel sessions

ISC in an IMSplex

Prior to the IMSplex environment, ISC sessions were always with the single IMS. In an IMSplex, you can continue to establish the ISC session(s) between CICS (for example) and the same IMS that it was connected to before. However, if that IMS is not available, CICS has no access to the IMSplex. Therefore, ISC sessions between CICS and IMS may be, and probably would be desirable to be, with multiple IMSs, each with the capability to process the transaction. If one is down, then others may still be available to process the workload.

ISC with traditional queuing

In a traditional (non-shared queues) environment, because each IMS has its own set of message queues, responses to input from CICS will always be sent back across the session which received the input. Each IMS can have the same ISC definitions with the same TERMINAL, SUBPOOL, and NAME macros. The remote CICS definitions would have to be changed, of course, to recognize multiple IMS destinations and to divide the workload between them.

ISC with shared queues

The following discussion assumes the use of shared queues and a system SANJOSE, split into two IMS systems, SANJOSE and SANJOSE2.

If the original SANJOSE ISC definitions to the BOSTON CICS system are cloned, and the BOSTON CICS system definitions are not changed, only the SANJOSE IMS subsystem can have a connection to the BOSTON CICS system. This is true because the BOSTON CICS system has no definitions that point to SANJOSE2. BOSTON CICS would never initiate a session with SANJOSE2, and an attempt to initiate it from SANJOSE2 would fail because it is not defined in BOSTON. Therefore, one alternative is to make no changes to the BOSTON CICS system. That is, the BOSTON and SANJOSE systems have the active set of parallel

ISC sessions. Message traffic from SANJOSE2 to BOSTON can be achieved through the shared queues and delivered to BOSTON by SANJOSE. Message traffic from BOSTON to SANJOSE2 also flows through the shared queues through the BOSTON to SANJOSE ISC connection. The advantages of maintaining the ISC sessions only between BOSTON and SANJOSE are:

- The ISC definitions for the IMS systems in SANJOSE can be unchanged and clones of one another. This simplifies the IMS system definition process for SANJOSE and SANJOSE2. In addition, operational control of the connection between the SANJOSE IMSplex and the BOSTON CICS system is unchanged.
- The routing of messages between SANJOSE2 and BOSTON through the shared queues is transparent to the application programs in all three systems.
- The BOSTON CICS system definition does not have to change. As long as the SANJOSE IMS system can handle the communication traffic, there is no need for considering implementing another connection from BOSTON to SANJOSE2.

One can consider defining and implementing another set of parallel ISC sessions between BOSTON and SANJOSE2 to remove the SANJOSE to BOSTON connection as a single point of failure. To do this for cloned IMS systems in the IMSplex:

- On the CICS side, an additional DEFINE CONNECTION must be coded specifying SANJOSE2 as the NETNAME. Additional DEFINE SESSIONS must be coded to define the individual sessions. These should have different SESSNAMEs and NETNAMEQs than the connections to SANJOSE to avoid duplicate names being used.

```
DEFINE CONNECTION
       NETNAME(SANJOSE)

DEFINE SESSIONS
       SESSNAME(S1)
       NETNAMEQ(SP1)

DEFINE SESSIONS
       SESSNAME(S2)
       NETNAMEQ(SP2)

DEFINE CONNECTION
       NETNAME(SANJOSE2)

DEFINE SESSIONS
       SESSNAME(S3)
       NETNAMEQ(SP1)

DEFINE SESSIONS
       SESSNAME(S4)
       NETNAMEQ(SP4)
```

- On the IMS side, assuming cloned definitions, each IMS will require that the active SUBPOOLs and LTERMs be unique, but all SUBPOOLs and LTERMs can be defined to both IMSs. Of course, only some of them will be used by each IMS. Additional SUBPOOL and NAME macros should be coded in the common system definition. The SUBPOOL names should match the NETNAMEQ names defined in the BOSTON CICS.

```
TERMINAL NAME(BOSTON)
   VTAMPOOL
      SUBPOOL NAME=(SP1)
         NAME LT1
      SUBPOOL NAME=(SP2)
         NAME LT2
```

```
SUBPOOL NAME=(SP3)
   NAME LT3
SUBPOOL NAME=(SP4)
   NAME LT4
```

ISC and parallel sessions

ISC sessions can be defined with single session or parallel session support. When parallel session support is utilized, each set of parallel sessions from another subsystem (for example, CICS) is connected to the same IMS system. However, there may be other sets of parallel sessions between that remote subsystem and another IMS in the same IMSplex.

ISC and VTAM Generic Resources

When VTAM Generic Resource support is enabled (see 6.2.8, “VTAM Generic Resources (VGR)” on page 141), the first logon request to the generic resource name may be routed to any of the available IMSs in the same generic resource group depending on VTAM’s resource resolution algorithms. Each additional logon request using that generic name will be routed to the same IMS. This should only be a problem if the traffic between one CICS and one IMS is excessive and causes performance problems. To establish single or parallel sessions with another IMS, those logon requests must use the real IMS APPLID.

6.2.5 APPC (LU6.2)

Remote APPC programs do not establish VTAM LU6.2 sessions with IMS - they establish these sessions with an APPC/MVS address space. It is this address space which opens the VTAM ACBs enabling session initiation (logon) requests from remote APPC programs. Once a session is established, that remote program can initiate APPC conversations with an APPC/MVS client such as IMS. There may be multiple conversations during the course of a session, but only one conversation at a time is supported. It is therefore likely that there will be parallel sessions between a remote APPC program and APPC/MVS. In IMS terminology, each APPC conversation would be an instance of an IMS non-conversational transaction. For IMS conversational transactions, the APPC conversation persists until the IMS conversation terminates.

The term APPC/IMS refers to the support in IMS for communications with APPC/MVS and for APPC conversations between IMS and the remote APPC program. The term APPC/IMS ACB refers to the VTAM ACB in APPC/MVS that is associated with a particular IMS. This is defined in SYS1.PARMLIB member APPCPMxx by associating an IMSID with one or more APPC/IMS ACBs. When the /START APPC command is entered to IMS, IMS notifies APPC/MVS to open these defined ACBs and begin accepting session initiation requests. After establishing the LU6.2 VTAM session (the sessions can be single session or parallel sessions), the remote APPC program can initiate an APPC conversation with the IMS associated with that session (only one active APPC conversation per session is allowed). Communications between IMS and APPC/MVS are supported by XCF communication services.

There are only a few issues related to APPC and an IMSplex. They are similar to those of any IMS VTAM terminal.

- The APPC program session must be with an APPC/IMS which is on the same OS/390 image as the IMS with which it wants to initiate a conversation.
- If that IMS fails, or is not available for any other reason, the remote APPC program must either wait for IMS to be restarted, or establish a session with another available APPC/MVS. Like the 3270 user, training will be required to determine how (and if) to establish another session.

- There are some special considerations for APPC when using VTAM Generic Resources. These are addressed in “APPC/IMS and VTAM Generic Resources” on page 143.

6.2.6 Multiple systems coupling (MSC)

Multiple systems coupling (MSC) is a private protocol between IMS systems only. It supports the routing of messages between IMSs which are remote to each other based on the destination of the message (remote in this case means IMSs that are not a part of the same shared queues group). Typically, remote IMSs and remote destinations such as transaction codes and logical terminals are defined to each IMS during system definition. However, it is possible, using directed routing, to send any message to a remote IMS without having defined the message as remote in the local IMS. There are several considerations when migrating from a non-IMSplex environment to an IMSplex environment.

MSC message routing

In a data sharing environment, it is usually desirable to distribute the workload across all the IMSs in the data sharing group which have the necessary resources (shared databases and transaction definitions) to process the work. This can be done in several ways, such as requiring end-users to log on to different IMSs (either directly or through the use of VTAM Generic Resources, USERVAR processing, and/or USSTAB processing), or by using shared queues. Another technique is the use of MSC connections between the IMSs and the use of MSC exits to route messages around the IMSplex.

MSC workload balancing attempts to balance the workload among the IMS systems in the IMSplex by distributing the work to the various systems through MSC definitions and/or MSC exits. For example, one could have all of the end-users logged on to one of the IMS systems and have that system route some percentage of the transactions to another system or systems within the IMSplex. Let us assume, for example, that all of the end-users are logged on to IMS1. In addition, we want 40 percent of the transaction workload to process on IMS1 and the other 60 percent to process on IMS2.

Routing based on specific transaction codes (a possible solution when systems are partitioned by application) can be accomplished through the use of IMS system definition specifications. Routing based on percentages (a possible solution when systems are cloned) requires the use of the TM and MSC Message Routing and Control User Exit (DFSMSCE0).

For more information on planning for MSC use within your IMS Parallel Sysplex environment please refer to *IMS/ESA Multiple Systems Coupling in a Parallel Sysplex*; SG24-4750.

MSC and the IMS Workload Router

The IBM IMS Workload Router for z/OS Version 2 Release 3 (product number 5697-B87) functions with the IMS Transaction Manager to distribute an IMS transaction workload among two or more IMS online systems interconnected through multiple systems coupling (MSC) communications links.

The Workload Router offers the following features:

- Distributes IMS transactions on predefined paths via MSC links
- Provides for weighted distribution of IMS transactions
- Provides for assignment of transactions or groups of transactions to a designated server system
- Supports parallel MSC sessions between the router (front-end) and server (back-end) systems
- Facilitates MSC link load balancing

- Automatically recognizes and avoids routing transactions to unavailable server system/MSC links
- Provides for automatic workload reconfiguration in the event of both planned and unplanned outages
- Provides an online/real-time administrator interface for monitoring and dynamically updating the Workload Router configuration

Although the basic functions of MSC can be used to distribute application workload through the IMSplex, with this tool, you have more control over the execution of the MSC environment. The Workload Router supports IMS Versions 7 and 8.

MSC and shared queues

The use of MSC in a shared queues environment is described in “Multiple systems coupling (MSC)” on page 91. To summarize, you cannot have active MSC connections between members of the same shared queues group. If you currently use MSC, with or without the Workload Router, to distribute work around the IMSplex, when you implement shared queues, you will have to change the definitions of the remote destinations to local destinations and let shared queues handle it.

6.2.7 Rapid Network Reconnect (RNR)

Rapid Network Reconnect (RNR) was introduced in IMS Version 7 as an implementation of VTAM single node persistent session (SNPS) support or VTAM multi-node persistent session (MNPS) support. With this support enabled, VTAM session information is retained in either a data space (for SNPS) or a Coupling Facility structure (for MNPS). When automatic reconnect is requested (RNR=ARNR in DFSDCxxx) and there is an IMS failure followed by an emergency restart, VTAM uses this session information to reestablish the session without requiring the user to log back on again. SNPS requires IMS to be restarted on the same OS/390 image while MNPS allows IMS to be restarted on any OS/390 image in the Parallel Sysplex.

Until that IMS is restarted, that terminal is considered to be in session with the failed IMS and cannot log on to another IMS. Because of this, when running IMS in an IMSplex where the user would want to log on to another IMS without waiting for the failing IMS to restart, this parameter should be specified as RNR=NONE. *Do not specify RNR=NRNR, as this holds the session active until IMS is restarted and then terminates it, giving you the worst of both worlds.*

6.2.8 VTAM Generic Resources (VGR)

IMS Version 6 implemented the original IMS support for VTAM Generic Resources (VGR). When an IMS in the IMSplex is started, it may join a VTAM generic resource group by specifying the GRSNAME execution parameter. This allows end-users to log on to any available IMS in the generic resource group by using the GRSNAME instead of a specific IMS APPLID. The obvious benefit of this is that the end-user does not have to be aware of which IMSs are available at any time, or even how many IMSs are in the IMSplex, allowing you to add or remove IMSs as the workload changes without affecting user logon procedures.

VTAM routes a logon request to one of the IMSs in the generic resource group based upon the following algorithm:

- If the terminal has an existing affinity for a particular IMS, the session request is routed to that IMS (see “IMS and VGR affinities” below). When this is the case, the following steps are not invoked.

- In the absence of an existing affinity, VTAM:
  - Selects an IMS based on existing session counts. VTAM attempts to equalize the session counts among the generic resource group members. If the LU logging on is local to any member(s) of the VGR group, then VTAM will select one of these members. This is most likely to occur when using session managers which run on the same OS/390 host as one or more of the IMSs. See “Session managers with VTAM Generic Resources” on page 143 for more discussion of this.
  - Accepts a specific IMS as recommended by the OS/390 workload manager (WLM). The OS/390 workload manager must be in goal mode. If used, this overrides VTAMs selection in the first step.
  - Invokes the user's VTAM Generic Resource Resolution Exit, if present. This exit can select any of the IMSs in the generic resource group as the recipient of the session request. It overrides either of the above two, including the “local LU” selection.

The preceding algorithm does have one exception. If an end-user logs on to a specific IMS APPLID, that request will be honored and bypass the algorithm entirely, including the exit.

IMS and VGR affinities

Once a terminal is logged on to IMS, it is said to have a *VGR affinity* for that IMS. VTAM keeps track of these affinities in a Coupling Facility structure (ISTGENERIC is the default name of the structure). This has some significant consequences for IMS.

- There is an IMS execution parameter called GRAFFIN which determines whether IMS or VTAM is responsible for deleting the affinity when the session terminates.
  - GRAFFIN=VTAM

    When set to VTAM, the *affinity is always deleted by VTAM* when the session terminates. This means that the next time you log on using the generic resource name, there will be no existing affinity and VTAM will apply the resource resolution algorithm described above.
  - GRAFFIN=IMS

    When set to IMS, IMS determines when the affinity is deleted. If you have what IMS regards as significant status, then IMS will not delete the affinity and the next time you log on using the GRSNAME, VTAM will route your request back to the same IMS. This is generally good if you have, for example, a conversation in progress with IMS and either IMS or your session is terminated while the conversation is still active. Only the IMS with which you have the conversation knows about it and can continue the conversation after reconnecting.

    That’s the good news. The bad news is that if it is an IMS failure, you have to wait for IMS to be emergency restarted. If you are willing to lose your conversational status, then you can log on directly to an available IMS APPLID and resume processing, but you will no longer be in the conversation.
- IMS Version 8 considerations
  - When the IMSplex is running with sysplex terminal management (which means IMS Version 8, shared queues, the Common Service Layer, and the resource structure), and the terminal session is allocated as SRM=GLOBAL, and any value for GRAFFIN, the VTAM affinity will always be deleted (either by IMS or by VTAM) and you can log on to any IMS (using the GRSNAME) and resume the conversation (or any other significant status).
  - With z/OS 1.2 and later, VTAM supports session level affinities and the GRAFFIN parameter is no longer required. It is ignored by IMS if coded. Instead, IMS sets affinity

deletion for each individual session according to a set of rules which generally result in VTAM managing the affinity any time SRM=GLOBAL.

– Refer to *IMS Version 8 Implementation Guide, A Technical Overview of the New Features,* SG24-6594, for a complete description of sysplex terminal management and VTAM Generic Resources.

Restrictions when using VGR for IMS connectivity

There are a few restrictions associated with VGR that you must be aware of during the planning phase:

- The remote IMS target of an MSC link cannot be defined using the generic resource name. Real APPLIDs must be used.
- IMS systems participating in XRF cannot be members of a VGR group. VTAM does not allow an application to be known by both the GRSNAME and the XRF USERVAR.
- An IMS system cannot be a member of more than one VGR group, although there can be more than one VGR group within an IMSplex.

User training

It may be necessary for the user to re-connect after a failure. Ideally, this would be to the same VGR group name as used before, but this might fail. Logging on directly to another IMS system's APPLID may result in an affinity remaining on the first IMS. The user would need to be made aware that reconnecting to the first IMS (after an unknown period) may result in re-establishing the significant status that existed at the time of failure.

Another possibility, following an IMS failure, would be for the user to retrieve messages asynchronously. This could happen in both a shared queues and non-shared queues environment.

In all the above scenarios, the end-user should be trained to handle session reconnect as well as affinity management.

APPC/IMS and VTAM Generic Resources

VTAM Generic Resources is supported between remote APPC programs and APPC/MVS. When an APPC node initiates a session, it can specify a generic name instead of a specific APPC/IMS ACB name. For example, if two IMS systems have APPC ACB names of LU62IMS1 and LU62IMS2, they might have a generic name of LU62IMS. The LU 6.2 session request would be for LU62IMS. VTAM Generic Resources would select either LU62IMS1 or LU62IMS2 for the partner node. The generic name is specified by the GRNAME parameter in the LUADD statement in APPCPMxx. This name must be different than the GRSNAME specified for non-APPC IMS VTAM Generic Resources.

A single remote APPC node can establish multiple sets of single or parallel sessions with one or more APPC/IMS ACBs. For example, program APPCR1 can establish parallel sessions with LU62IMS1 and another set of parallel sessions with LU62IMS2. When using generic resource support, VTAM will select the ACB with which to establish the first session. Each additional parallel session will be routed to the same APPC/IMS ACB. Different sets of parallel sessions from the same remote APPC program (determined by using different mode tables in the session initiation request) can be routed to different APPC/IMS ACBs even when using the APPC generic resource name (GRNAME).

Session managers with VTAM Generic Resources

Many installations use session managers to allow a single VTAM logical unit to log on once (to the session manager) and then gain access to several different VTAM applications, such as IMS, CICS, or TSO. The session manager creates its own sessions with these other VTAM applications and then routes requests from the end-user to whichever application that end-user wants. The end-user does not have to log off one application and log on to another.

Session managers can negate many of the advantages of VTAM Generic Resources. VGR's algorithm for distributing logon requests is first to try to balance the logons equally across the IMSs in the generic resource group. However, if the LU making the logon request is local to that VTAM, and if there is a local IMS, then the logon request will be sent to that IMS regardless of the number of logons already active there. If there were only a single session manager, then all logons would be to the same local IMS. There are a couple possible solutions to this problem:

- Make sure the session manager is running on a different LPAR than any of the members of the IMS VTAM generic resource group. When this is the case, the session manager LUs are not local and normal VTAM resource resolution algorithms apply.
- VTAM provides the ability to write a VTAM Generic Resource Resolution Exit (IXGEXCGR) which can override VTAM's decision about the IMS to which to route the logon request. This exit can be used to distribute logon requests from all LUs, including local session managers.

6.3 TCP/IP network connectivity

TCP/IP has evolved to become the accepted standard for networking. As a result, there is now an ever-increasing need to support interoperability over a TCP/IP network to access applications and data on host environments such as IMS. This section describes some of the special considerations when connecting to an IMSplex through TCP/IP. For all TCP/IP traffic that comes into IMS OTMA using XCF communications, we do not have the variety of special IMS considerations that we have for VTAM with its many LU types. However, you should consider what happens outside IMS. That is, how do your users get to OTMA in the first place?

Working from the inside out, we will discuss the following products, programs, and services that utilize XCF and OTMA to enter transactions. There is also the special case of the WebSphere MQ Adapter. Again, we are interested primarily in how their implementation might change from a non-IMSplex to an IMSplex environment.

6.3.1 Open transaction manager access (OTMA)

IMS OTMA can be defined (in its simplest form) as a transaction-based, connectionless client/server protocol whose domain is that of the MVS Cross System Coupling Facility (XCF). It is the functional equivalent of the traditional IMS VTAM device dependent module, except in this case, the communications protocol is XCF. OTMA clients (such as IMS Connect or WebSphere MQ) use XCF protocols to deliver messages to IMS for processing and to receive the IMS application responses. The real end-user, for example, someone connected to a TCP/IP network such as the internet, identifies itself to OTMA by a name coded in the message prefix which IMS refers to as the transaction pipe (TPIPE) name. It is similar to a logical terminal. When IMS delivers the transaction response through OTMA to the TCP/IP server application, the server associates this TPIPE name with the originating device (similar to the association of an LTERM name with a NODENAME). The TPIPE name in the response tells the server application where to send the output reply.

There are three types of OTMA clients:

- IMS Connect (TCP/IP server application)
- WebSphere MQ (TCP/IP server application or APPC/MVS client application)

- User applications written in C/C++ and using the OTMA callable interface documented in *IMS Version 8: Open Transaction Manager Access Guide,* SC27-1303.

6.3.2 IMS Connect

This TCP/IP server application runs on an OS/390 or z/OS platform in the Parallel Sysplex and is the destination component to which TCP/IP end-users can route input messages for IMS. IMS Connect is an IMS OTMA client, using XCF to route incoming traffic to any IMS in the IMSplex which is a member of the same OTMA XCF group (defined by the IMS execution parameter GRNAME and the equivalent IMS Connect DATASTORE configuration parameter GROUP).

Figure 6-1 shows that a client in a TCP/IP network, such as the Internet, entering a transaction through one of two IMS Connects to an IMS in the IMSplex. A description of the physical connection to the TCP/IP stack is discussed later in this chapter. However, in this configuration, that user may be connected to either of the two IMS Connect TCP/IP server applications. Whichever IMS Connect receives the message can route it to any IMS in the same OTMA XCF group to which IMS Connect itself belongs.

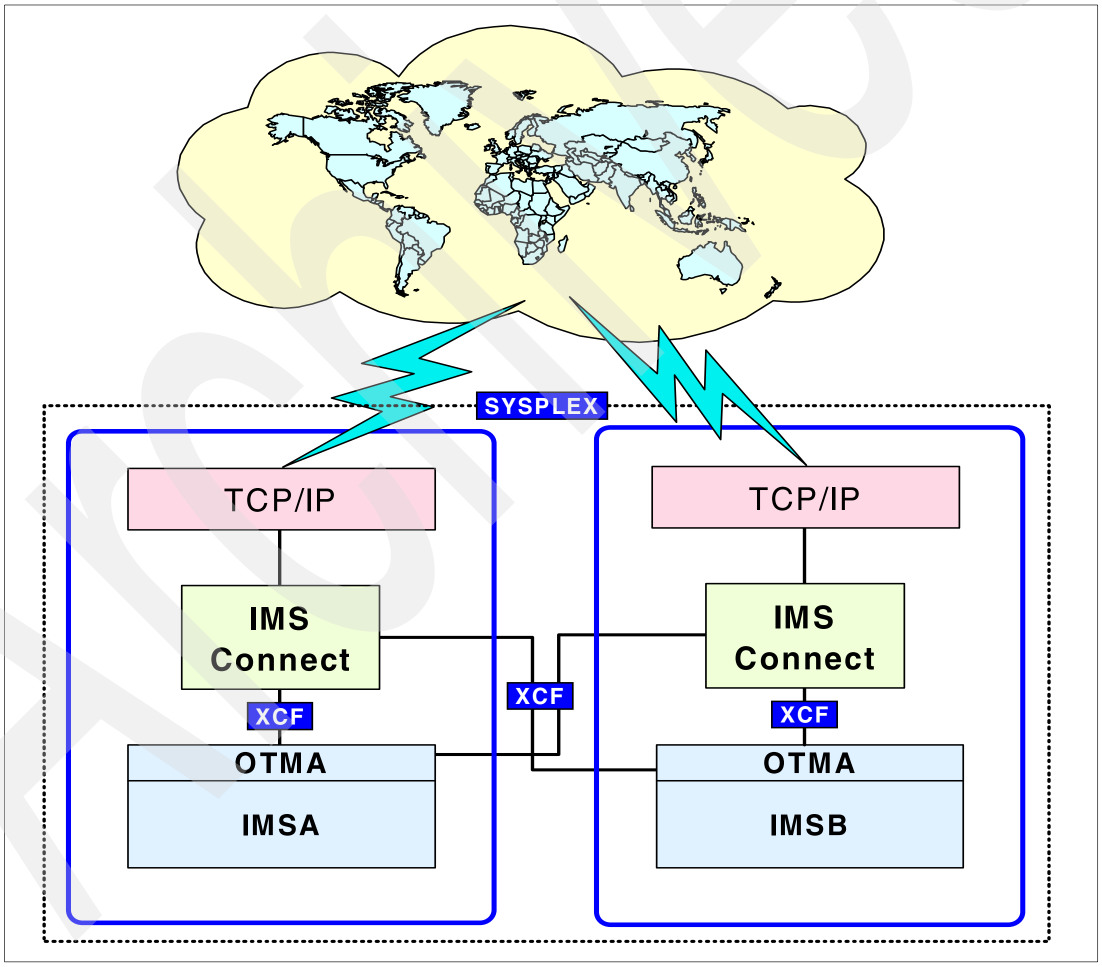


Figure 6-1 IMS Connect and OTMA

IMS Connect maintains a DATASTORE Table which identifies the status of each IMS connected to the same XCF group. A User Message Exit in IMS Connect can use this table to determine which IMSs are available and to decide to which of the available IMSs in the IMSplex to route the message. In this way, if IMSA (for example) is unavailable, inbound messages can be sent to IMSB. If both are available, the exit can balance the workload between IMSA and IMSB. The exit may be coded to recognize transactions that can only go to one or the other IMS. In short, the exit, if coded, has complete control over which IMS is the ultimate destination of the input message.

Once a message is received by IMS, it will queue that message on a local or shared queues where it will be scheduled for processing in a dependent region. The response to the input is then sent back to the remote client through OTMA and IMS Connect. The TPIPE name in the input message prefix is included with the output response, telling IMS Connect where to send the response. IMS does not know the ultimate destination of the message.

6.3.3 WebSphere MQ

WebSphere MQ can connect to the network as a VTAM APPC (LU6.2) partner TP program or as a TCP/IP server application. Once a message is received and queued by MQ, there are three methods by which it can deliver these messages to IMS:

- IMS Bridge
- IMS Adapter
- Native MQ API call from IMS application program

WebSphere MQ queue sharing group

WebSphere MQ for z/OS V5.2 (and above) exploits the Parallel Sysplex and Coupling Facility (CF) technology to allow queue managers to be configured to form a cooperating queue sharing group (QSG). Within a QSG, queues may be defined as *shared queues which are accessible to all queue managers* in the QSG. Queues which remain accessible by individual queue managers only are non-shared or private queues. A new class of channels known as shared channels are also provided. These channels offer enhanced behavior and availability over non-shared or private channels.

WebSphere MQ IMS Bridge

In this implementation, WebSphere MQ is an *OTMA client*, routing inbound messages to IMS OTMA using XCF protocols. IMS puts the message on the local or shared message queue for scheduling and processing. Output response messages from IMS dependent regions also go through the IMS message queues. To the application program, there is no difference between these messages and those received from, for example, a 3270 terminal.

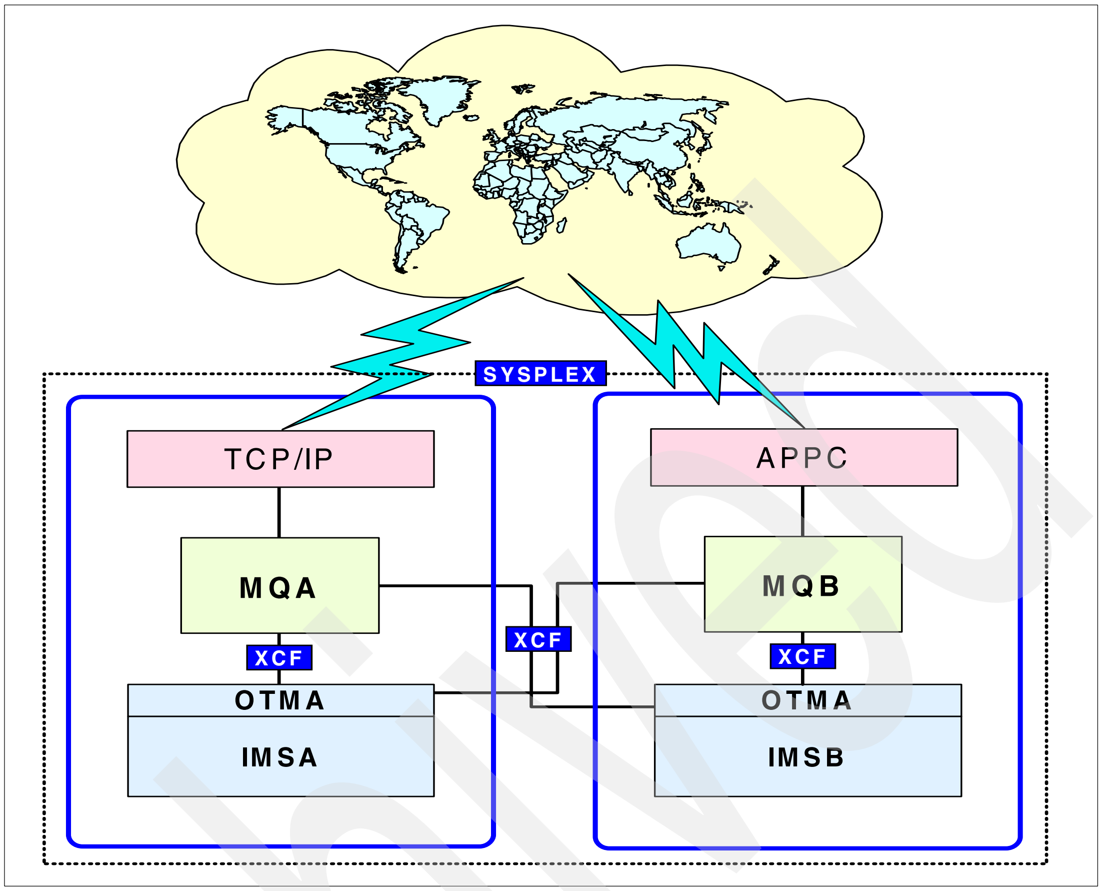


Figure 6-2 WebSphere MQ IMS Bridge via OTMA

Some characteristics of this implementation are:

- One WebSphere MQ queue manager can connect to multiple IMS control regions
- One IMS control region can connect to multiple WebSphere MQ queue managers
- WebSphere MQ and IMS control regions must be in the same XCF group
- WebSphere MQ and IMS can be on different OS/390 images within the same Parallel Sysplex
- Application programs use the IMS message queues for all input and response messages

WebSphere MQ queue sharing group and IMS Bridge

A particularly common type of server application makes use of the IMS Bridge. Requesting applications construct messages in a format suitable for running legacy IMS transactions, maybe using the MQIIH header to provide additional control of the transaction environment. These messages are placed on “IMS Bridge” queues. IMS Bridge queues are linked by a WebSphere MQ storage class definition to a particular IMS control region. When messages arrive on the IMS Bridge queues, they are pulled off and transmitted via XCF to the appropriate IMS (OTMA) region by special IMS Bridge tasks inside the MQ. IMS queues the transactions locally or globally for scheduling and processing. Once a transaction is executed, the response that it generates is returned to the MQ queue manager which places it on the appropriate “reply to” queues. In a queue sharing group, the IMS Bridge can be configured for either capacity or availability, on a per queue basis.

Configuring a IMS Bridge queue for capacity

- Define the IMS Bridge queue to be a shared queue to allow it to be accessed by internal IMS Bridge tasks on each queue manager in the shared queues group.
- Define the queue with DEFSOPT(SHARED) and SHARE to allow multiple tasks to concurrently have it open for input.
- Ensure that the storage class definition on each queue manager targets a different IMS control region (ideally one which resides on the same LPAR as the queue manager).

Now, each queue manager can run its own internal IMS Bridge task against the shared queues and pass messages from the shared queues to a back-end IMS in parallel with its peers. Within IMS, shared IMS message queues may be used, as required, to increase availability or solve issues relating to affinity of message processing regions to LPARs.

Configuring a IMS Bridge Queue for availability

For some applications it is not appropriate to run multiple servers against queues. This typically is the case where messages must be processed in strict sequence. The IMS Bridge can be configured for these types of applications as follows:

- Define the IMS Bridge queue as a shared queue, but with options DEFSOPT(NOSHARED) and NOSHARE. This restricts access to the queue, so that only a single task can open the queue for input at a time.
- As above, ensure that the storage class definition on each queue manager targets a different IMS control region.

Now, only a single IMS Bridge task can run against the shared queue at a given time. But, if the queue manager where the task is running should abnormally terminate, the other queue managers in the queue sharing group will be notified and a race will occur to start another instance of the IMS Bridge task to serve the shared queue.

WebSphere MQ IMS Adapter

In this implementation, WebSphere MQ uses the IMS External Subsystem (ESS) interface to signal a "listener BMP with an outstanding MQGET" running in a local IMS dependent region to insert a *trigger* transaction on the IMS message queue. The trigger transaction is then scheduled and retrieves the real transaction input from the MQ queue using the MQ API (MQ_GET). The response is sent using MQ_PUT directly to WebSphere MQ. Neither the transaction nor the response go through the IMS queue. Two-phase commit processing between IMS and WebSphere MQ is managed by the ESS interface. The application programs are obviously different than those processing messages from the IMS message queues.

Figure 6-3 shows two WebSphere address spaces, one on each of two systems in the Parallel Sysplex. One is TCP/IP connected and the other is APPC connected. In this configuration, IMS and MQ must be on the same OS/390 image. If IMSB, for example, fails, then MQB would not be able to route transactions to IMSA.

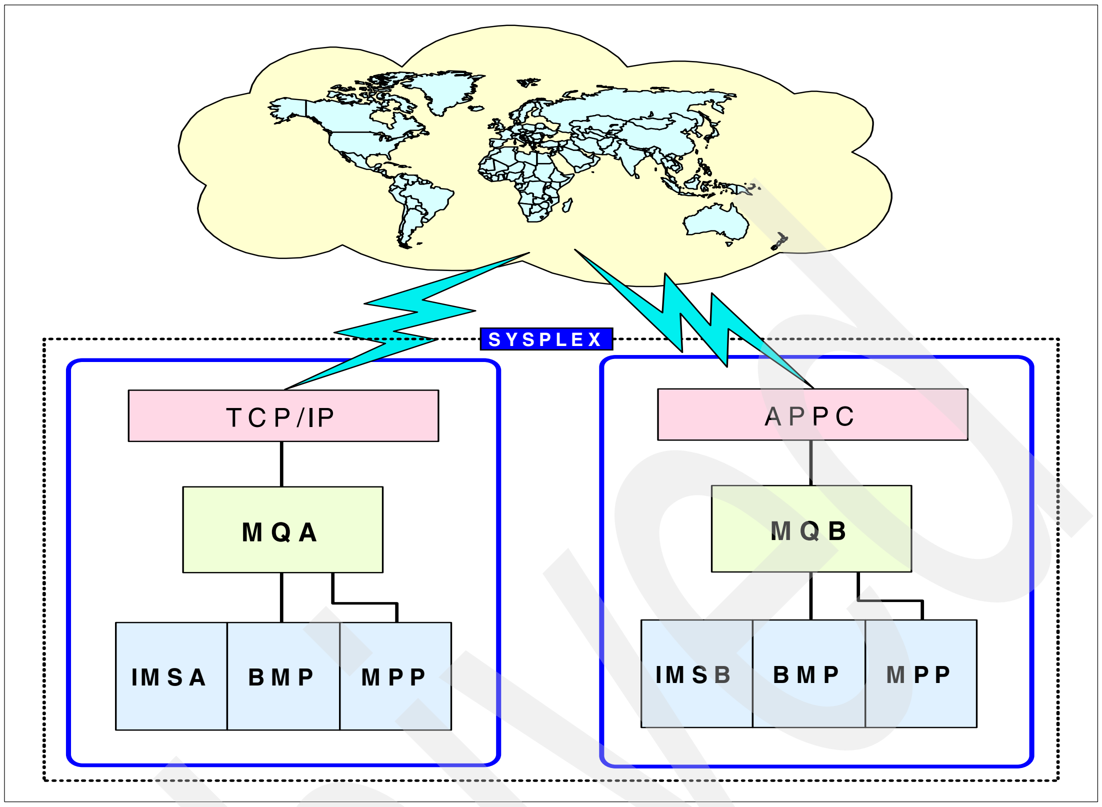


Figure 6-3 WebSphere MQ IMS Bridge via ESS

WebSphere MQ queue sharing group and IMS Adapter

When using the IMS Adapter, IMS connects to a given queue manager. There is no group attachment available for IMS, it is only supported for batch applications. So, your IMS transaction management systems should continue to connect to a specific, named queue manager that is on the same LPAR as the transaction management system.

If you are planning to create a QSG with cloned queue managers, then you should ideally have 'cloned' transaction managers that connect to each of these queue managers. So, for example, it should be possible to run instances of your transactions on each transaction manager. Additionally, if you are using triggering, then you should have trigger monitors started on each transaction manager.

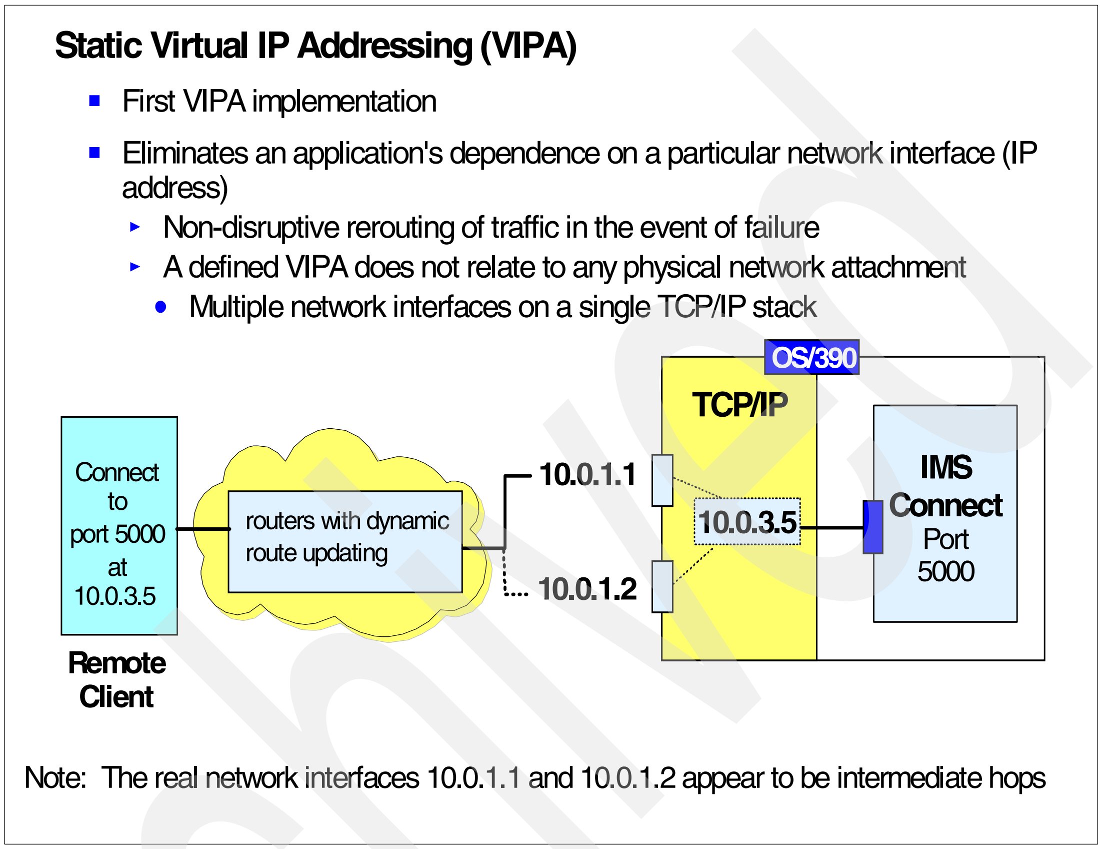
For a more detailed description of using the WebSphere MQ queue sharing group, refer to the IBM Redpaper *WebSphere MQ Queue Sharing Group in a Parallel Sysplex environment,* REDP3636.

6.3.4 Virtual IP Addressing

Virtual IP Addressing (VIPA) allows for the definition of a virtual IP address that does not correspond to any physical interface but rather is associated with the stack as a whole. The VIPA address appears to be on a separate subnetwork with the stack itself as a gateway to that subnetwork. The use of VIPA presumes that a TCP/IP stack has multiple physical links or interfaces. Client packets that are targeted to the VIPA address will be routed to one of the available physical interfaces. If that interface fails the connection continues to remain intact and subsequent packets continue to be routed to the VIPA address using one of the other active interfaces.

Static VIPA

Static VIPA is the term used to refer to the first, and actually simplest, implementation of VIPA. It supports failover from one network interface to another on a single stack as shown in the following figure.


Static Virtual IP Addressing (VIPA)

- First VIPA implementation
- Eliminates an application's dependence on a particular network interface (IP address)
  - Non-disruptive rerouting of traffic in the event of failure
  - A defined VIPA does not relate to any physical network attachment
    - Multiple network interfaces on a single TCP/IP stack

Note: The real network interfaces 10.0.1.1 and 10.0.1.2 appear to be intermediate hops

Figure 6-4 Static VIPA fundamentals

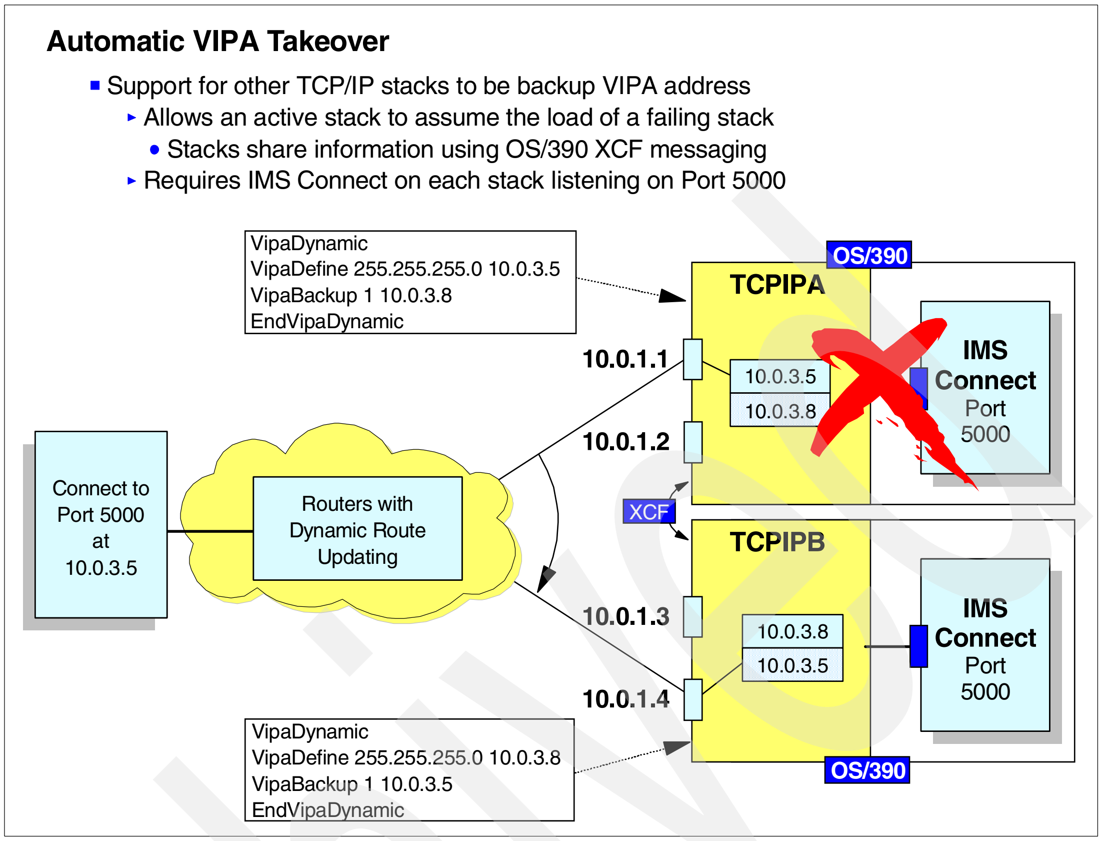
Note that the client application references the target server using the VIPA address or a hostname that resolves to the VIPA address. In this example it is 10.0.3.5. The real network interfaces 10.0.1.1 and 10.0.1.2 appear to be intermediate hops. When the connection request is made, one of the network interfaces (for example, 10.0.1.1) is chosen for the connection and all subsequent data transfer.

The definition and activation of VIPA is done through configuration statements in the *hlq.profile.tcpip* file. IMS Connect is unaware of VIPA. It simply connects to a TCP/IP stack and relies on the stack to perform the appropriate network function.

Dynamic VIPA

Beginning with OS/390 V2R8 the VIPA concept was extended to be more dynamic and to support recovery of a failed TCP/IP stack. Dynamic VIPA takeover was introduced to automate the movement of the VIPA to a surviving backup stack. The use of this capability presumes that server application instances (for example, IMS Connect instances), exist on the backup stack and can serve the clients formerly connected to the failed stack.

In the example shown in Figure 6-5 on page 151, the Dynamic VIPA IP address 10.0.3.5 is defined as having a home stack on TCPIPA and a backup on TCPIPB.



Automatic VIPA Takeover

- Support for other TCP/IP stacks to be backup VIPA address
  - Allows an active stack to assume the load of a failing stack
    - Stacks share information using OS/390 XCF messaging
  - Requires IMS Connect on each stack listening on Port 5000

```
VipaDynamic
VipaDefine 255.255.255.0 10.0.3.5
VipaBackup 1 10.0.3.8
EndVipaDynamic
```

```
VipaDynamic
VipaDefine 255.255.255.0 10.0.3.8
VipaBackup 1 10.0.3.5
EndVipaDynamic
```

Figure 6-5 Dynamic VIPA

Likewise, TCPIPB is defined as the primary for 10.0.3.8 and TCPIPA as the backup. Both stacks share information regarding the Dynamic VIPAs through the use of XCF messaging services. Each TCP/IP stack, therefore, is aware of all the Dynamic VIPA addresses and the associated primary and backup order.

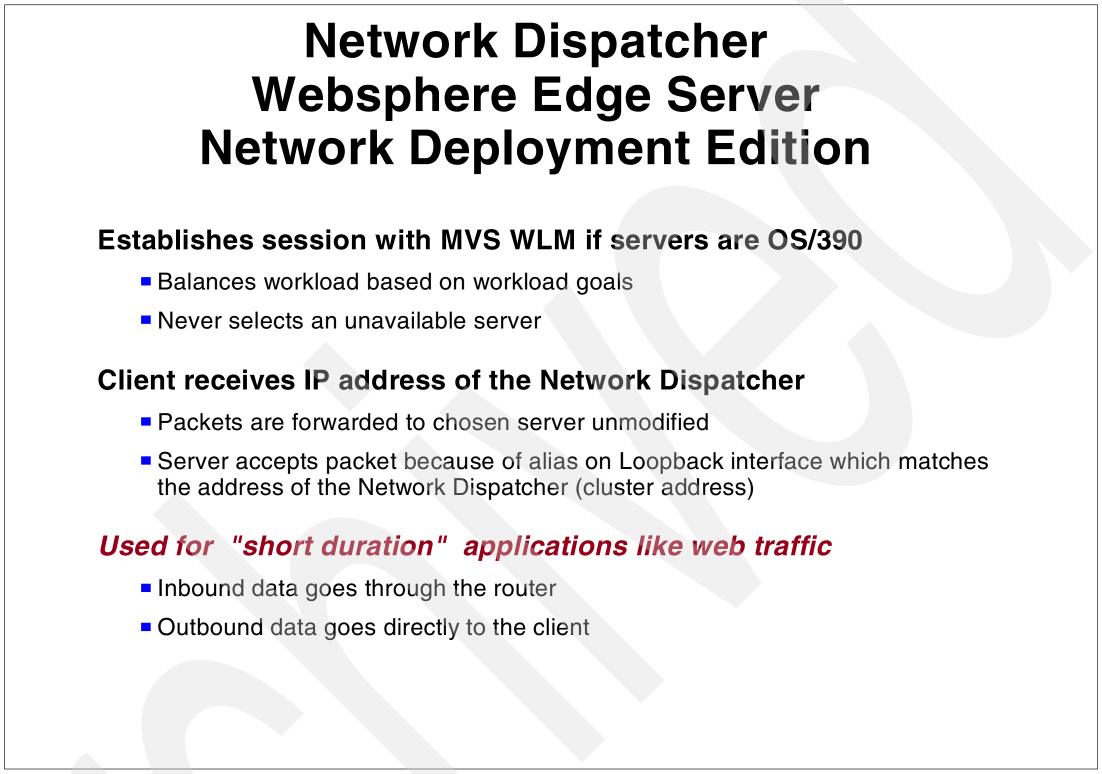
If a stack or its underlying OS/390 or z/OS fails, all other stacks in the sysplex are informed of the failure. The VIPA address is automatically moved to the backup stack which receives information regarding the connections from the original stack. All new connection requests to the VIPA address are processed by the backup which becomes the new active. Instances of the server applications (for example, IMS Connect systems), listening on the same ports should be automatically started if they are not already active on the backup. Additionally, the network routers are informed about the change. From a remote client application perspective, a connection failure is received when the primary stack fails but the client can immediately resubmit a new connection request which will be processed by the backup (new active) stack.

6.3.5 Network Dispatcher: WebSphere Edge Server

Connection requests to IMS Connect can be short-lived, as is the case for transaction sockets and non-persistent sockets, or they can be long-lived, as is the case for persistent sockets. This is important to understand because the decision for load balancing, that is, picking a specific server instance to process the work, occurs during the up-front connection phase and not during the data transfer of individual messages.

The Network Dispatcher is a function that intercepts connection requests and attempts to balance traffic by choosing and then forwarding the request to a specific server in the sysplex. This helps maximize the potential of a sysplex because it provides a solution that can automatically find new servers as they are enabled and added to the sysplex. It can also detect a failed server and route new connection requests only to the available servers.

The Network Dispatcher function was originally implemented and delivered in IBM networking hardware such as the 2216 and the 3745 MAE. More recently, it has been delivered as an integrated component of the IBM WebSphere Edge Server, Network Deployment Edition, which can be implemented on platforms such as AIX®, Windows NT, Windows 2000, Sun Solaris, and Linux. Figure 6-6 identifies some of its characteristics and capabilities.



Network Dispatcher
Websphere Edge Server
Network Deployment Edition

Establishes session with MVS WLM if servers are OS/390

- Balances workload based on workload goals
- Never selects an unavailable server

Client receives IP address of the Network Dispatcher

- Packets are forwarded to chosen server unmodified
- Server accepts packet because of alias on Loopback interface which matches the address of the Network Dispatcher (cluster address)

Used for "short duration" applications like web traffic

- Inbound data goes through the router
- Outbound data goes directly to the client

Figure 6-6 Network Dispatcher - WebSphere Edge Server

In this environment, clients send their connection requests and data to a special IP address that is defined as a *cluster address* to the Network Dispatcher. This same address is further defined as a *loop back alias address* on all the sysplex IP stacks that contain copies of the target application server, IMS Connect in this case. When a request resolves to this special address, the Network Dispatcher selects one of the backend servers and forwards the packet to the appropriate port and server.

The selection of a server for load balancing requirements can be controlled through several mechanisms. It can be based on simple round-robin scheduling to available servers, or it can use more sophisticated techniques. These can be based on the type of request (HTTP, FTP, Telnet, and so on...), by analysis of the load on the servers through information obtained from the MVS Workload Manager (WLM), or even through an algorithm based on weights assigned to each server. Figure 6-7 shows that a remote TCP/IP client sends a message to the IP address of the Network Dispatcher, which then selects a host TCP/IP defined with the same Loopback Alias.

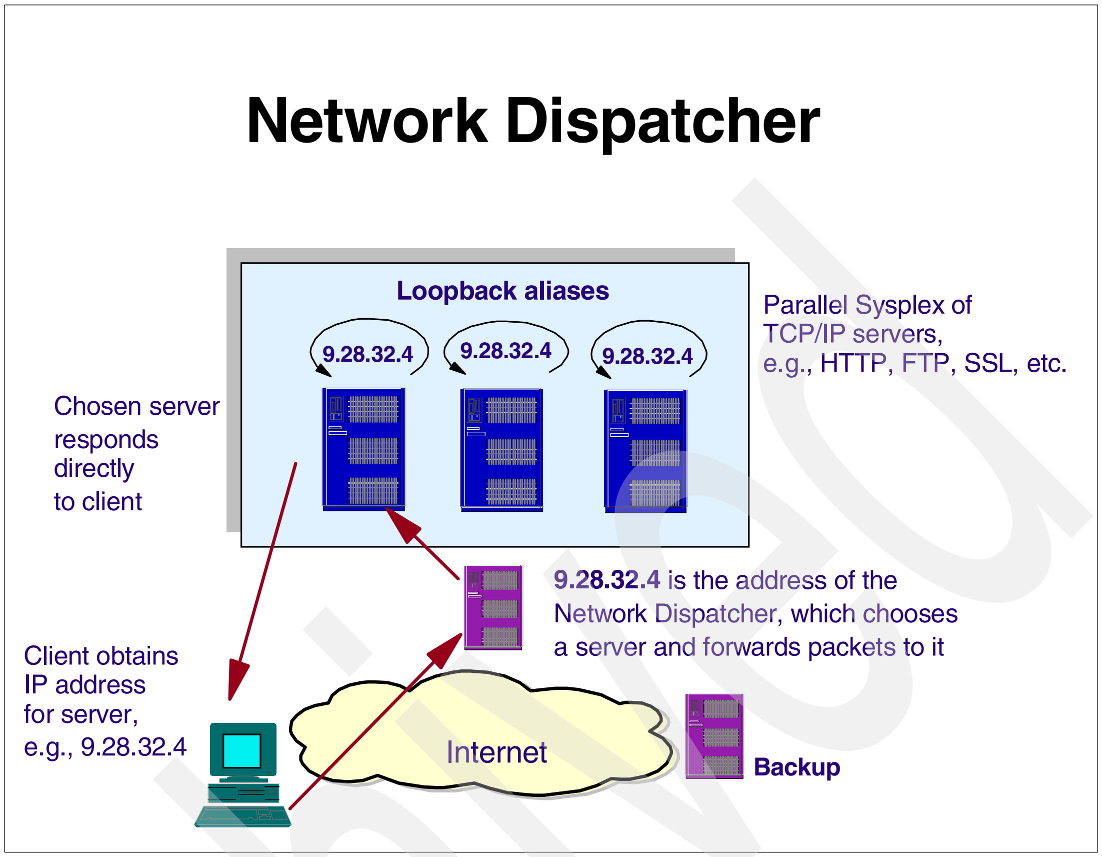

Network Dispatcher
Loopback aliases
9.28.32.4
9.28.32.4
9.28.32.4
Parallel Sysplex of TCP/IP servers, e.g., HTTP, FTP, SSL, etc.
Chosen server responds directly to client
9.28.32.4 is the address of the Network Dispatcher, which chooses a server and forwards packets to it
Client obtains IP address for server, e.g., 9.28.32.4
Internet
Backup

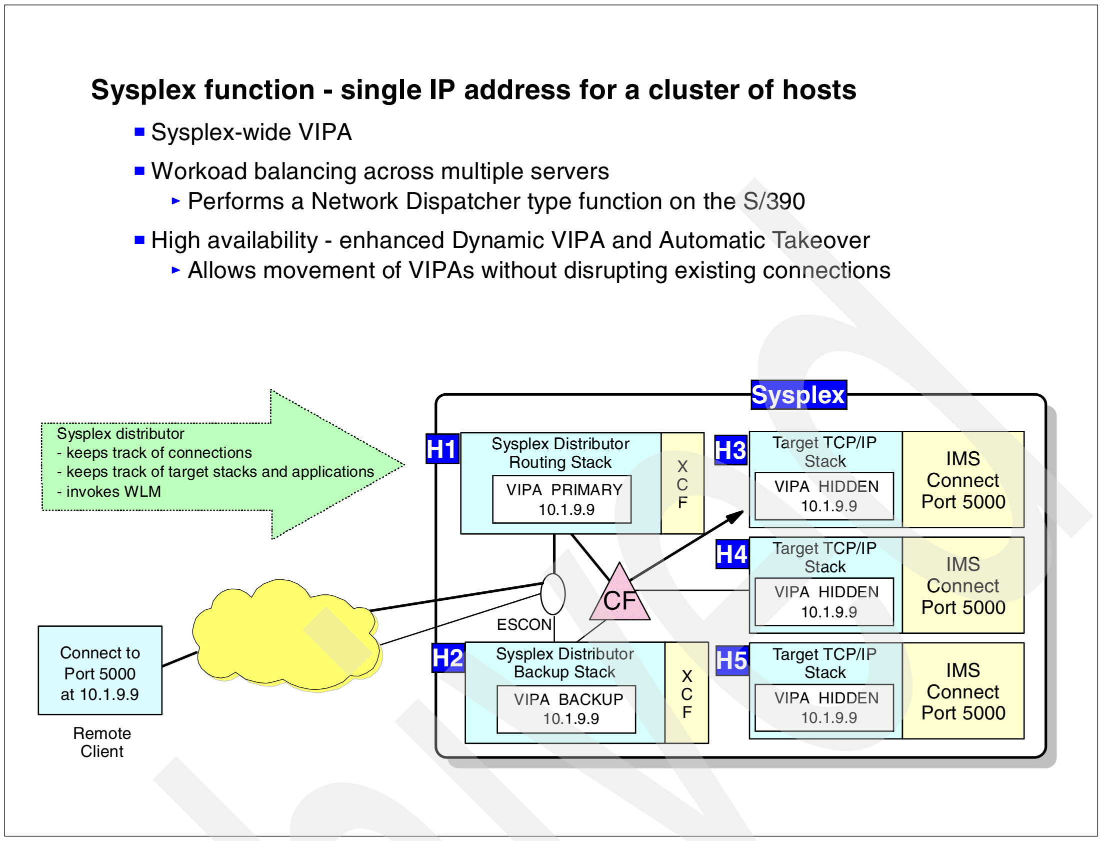
Figure 6-7 Network Dispatcher controls workload distribution and load balancing

Once a server is selected, the connection request and all subsequent data packets on that connection are routed to that server. Since IP packets contain the originating IP address of the requesting client, the server can reply directly to the client without sending the output back through the Network Dispatcher.

6.3.6 Sysplex Distributor

The Sysplex Distributor is an OS/390 V2R10 enhancement to sysplex IP support and is a function that resides on a sysplex host. It addresses the requirement for a *single network-visible IP address for the sysplex cluster* and also provides a complete solution to workload balancing and high availability. In terms of the concepts we have just discussed, the Sysplex Distributor provides a Network Dispatcher type of function with the benefits of dynamic VIPA all rolled up into a single host-based solution.

As a sysplex function, it removes the configuration limitations associated with the Network Dispatcher (XCF links rather than LAN connections are used between the distributing stack and the target servers) and further removes the requirement of specific hardware in the wide area network (WAN). It also enhances the Dynamic VIPA capability and takes advantage of the takeover/takeback support for its own distributing and backup stacks. Figure 6-8 illustrates the concept.



Sysplex function - single IP address for a cluster of hosts

- Sysplex-wide VIPA
- Workoad balancing across multiple servers
  - Performs a Network Dispatcher type function on the S/390
- High availability - enhanced Dynamic VIPA and Automatic Takeover
  - Allows movement of VIPAs without disrupting existing connections

Figure 6-8 Sysplex Distributor function

As part of the implementation, the Sysplex Distributor is configured with a distributing stack in one of the sysplex images (for example, H1) and a backup stack on another image (H2). The other images (H3-H5) can be configured as secondary backups. Only the active distributing stack takes the responsibility of advertising the dynamic VIPA (DVIPA) outside the sysplex.

The stacks in the sysplex communicate with each other using XCF services. H1 detects which stacks in the sysplex have active listening PORTs (for example, Port 5000 for IMS Connect) that are defined as part of the DVIPA environment. The distributing stack builds a table to keep track of server information and also of all connection requests associated with the DVIPA.

When an inbound connection request arrives, the distributing stack selects an available target server with a listening socket and uses XCF services to send the connection to the selected stack. The connection is actually established between the remote client and the target server, in this case H3. The distributing stack on H1 updates its connection table with the information. This allows H1 to know where to route subsequent data packets that are sent on that connection. When the connection terminates, H3 notifies H1 that the connection no longer exists so that H1 can update its table.

6.4 IMS database connectivity

The companion to transaction, or network, connectivity is database connectivity. When transactions are scheduled by IMS, they have connectivity to databases through the PSB. However, there are means of accessing databases other than by scheduling a transaction:

- BMP access

- CICS access
- ODBA access

In a data sharing environment there are a few special considerations.

6.4.1 BMP access to shared data

BMPs must connect to an IMS on the same OS/390 image as the BMP itself. However, when there are multiple IMSs sharing data, and all have the correct databases and PSBs available, the BMP may be scheduled on any image which has an available IMS. This means, however, then that IMSID in the BMP procedure must be correct.

BMP connection to IMS

IMS provides GROUPNAME support for dependent regions which may be started on the same OS/390 image. In the IMSPBxxx PROCLIB member, you specify the GROUPNAME parameter which should be different than the real IMSID. Then in the BMP, code the group name for the IMSID. When IMS is initialized, it invokes MVS token name services to associate the group name with the real IMSID. Obviously, each IMS will make a different association. When the BMP is scheduled, it will invoke MVS token name services using the group name to see if there is a local IMS and what its IMSID is. It then uses this IMSID to connect.

BMP connection to DB2

IMS BMPs must also be able to connect to a local DB2. In IMS Version 8 and IMS Version 7 with APAR PQ42180, IMS supports the use of DB2 MVS group names for DB2 Version 5 Release 1 and later versions. The DB2 group name may be used in all online regions. IMS uses the subsystem name from its SSM member entries to build a group name when the named subsystem is not attached to the control region. The group name is then used to call MVS Token Name Services. If MVS returns a positive response to the group inquiry, IMS will select a member of the DB2 group that is connected to the IMS control region, and connect that DB2 to the dependent region with the following restrictions:

- The selected DB2 was not explicitly defined by another statement within the SSM member.
- The selected DB2 was not previously selected from an MVS group.

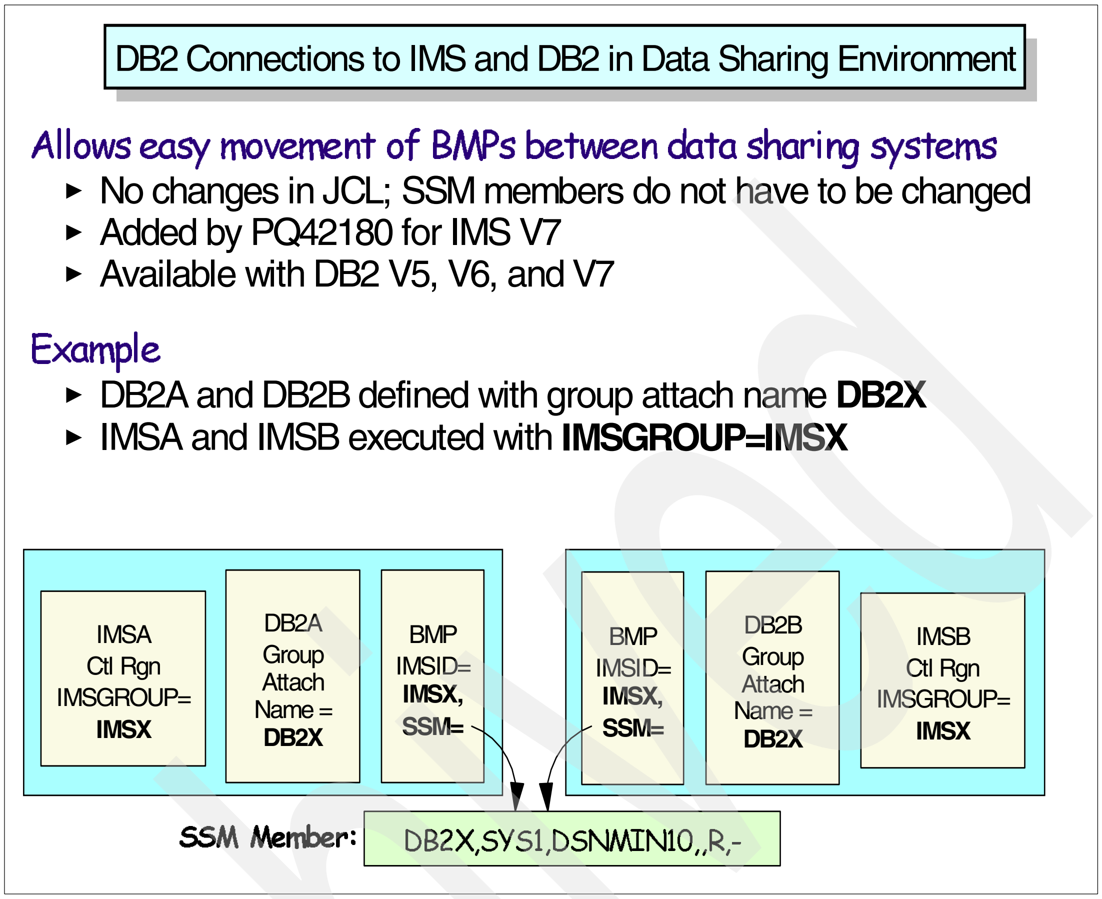
If either of the preceding restrictions is encountered, IMS will ignore the statement that caused the MVS call (no messages will be issued).

The following apply to SSM definitions when using group names:

- SSM= must be specified when starting the dependent region
- SSN= must specify the one- to four- character DB2 group name
- When CRC= is specified and a DB2 has been selected from a group, the CRC will be changed to that of the control region's corresponding SSM member statement

The advantage of this feature is that it allows easy movement of BMPs between IMSplex members. This results in the ability of applications running in BMPs to continue to access DB2 data using any available DB2 defined within the DB2 group. This increases end-user availability to data and easier workload balancing when necessary.

Figure 6-9 shows an example of a BMP which might be scheduled on either of two OS/390 images with data sharing IMSs and DB2s.



DB2 Connections to IMS and DB2 in Data Sharing Environment

Allows easy movement of BMPs between data sharing systems

- No changes in JCL; SSM members do not have to be changed
- Added by PQ42180 for IMS V7
- Available with DB2 V5, V6, and V7

Example

- DB2A and DB2B defined with group attach name **DB2X**
- IMSA and IMSB executed with **IMSGROUP=IMSX**

IMSA Ctl Rgn IMSGROUP= IMSX
DB2A Group Attach Name = DB2X
BMP IMSID= IMSX, SSM=
BMP IMSID= IMSX, SSM=
DB2B Group Attach Name = DB2X
IMSB Ctl Rgn IMSGROUP= IMSX
SSM Member:
DB2X,SYS1,DSNMIN10,,R,-

Figure 6-9 BMP connectivity to IMS and DB2 data sharing environment

6.4.2 CICS database control connectivity

Like BMPs, CICS must connect to an IMS on the same OS/390 image. However, CICS does not support the IMSGROUP. Therefore, it is necessary to explicitly define the IMSID to each CICS. This also means that, if using ARM for cross-system restart, IMS and CICS should be in the same restart group.

6.4.3 Database connectivity through ODBA

Open database access (ODBA) is used by non-IMS and non-CICS application programs to connect to an IMS control region and schedule a PSB to access the databases. Like CICS, the IMS control region must be on the same OS/390 image. This must be considered when invoking programs which run as DB2 stored procedures in a Workload Manager address space or Enterprise JavaBeans (EJBs) in a WebSphere Application Server address space.

6.5 Summary of connectivity options

The IMSplex provides many capacity and availability benefits over the single IMS environment. However, to take maximum advantage of these benefits, a robust plan for end-user connectivity (both transaction and database) must be implemented. We have tried to describe some of the products, programs, and services available to help you do this. There are undoubtedly more, some of them not IBM provided, that you could use. What is important is that you include connectivity as an integral part of your IMSplex migration plan. All of the following can play a role:

- Data sharing
- Shared queues
- Common service layer resource manager
- IMS Connect
- WebSphere MQ
- Dynamic VIPA
- Sysplex Distributor
- WebSphere Edge Server - Network Dispatcher
- IMS and DB2 group attach

7

Putting it all together

In 1.1.1, “Planning phase” on page 2, we identified several activities that should be a part of the planning phase of your IMSplex migration. The first of the two of these activities have been addressed in individual chapters. They are:

- Defining your objectives and expectations
- Functional planning

This chapter addresses the next two activities in migration planning and includes activities that apply to the entire IMSplex as well as to individual functional components such as data sharing, shared queues, and the Common Service Layer. Many of the planning activities apply to non-IMSplex implementation as well as IMSplex migration, but we will try to identify where you may see some differences:

- Configuration planning

  What will the intermediate and final configuration of your IMSplex look like? What are the IMSplex components and how will users be connected to the IMsplex?

- Security planning

  How will you provide access security to the IMSplex? What new resources will require protection, and how will existing resource security requirements change?

The final two activities are addressed in this volume in only a cursory fashion. The details of these activities can be found in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929:

- Implementation planning

  What tasks are required to implement an IMSplex? What are the requirements of the new IMSplex components and what are the new requirements of existing IMSplex components?

- Operational planning

  What are the operational considerations for execution in an IMSplex environment? How do operations change? How are new operational components such as the TSO Single Point of Control, or the IMS Control Center going to be used?

7.1 Configuration planning

Throughout the planning phase, you will be envisioning an IMSplex configuration as your final target. This configuration is likely to change as you learn more about the IMSplex and its unique characteristics and capabilities. During this part of the planning process you will define what your *final IMSplex configuration* will be, and any *intermediate configurations*. For example, your first configuration may be just data sharing, then shared queues, and followed finally by the Common Service Layer. Or you may implement the CSL before shared queues to establish an IMSplex operating environment. The configuration should include the Parallel Sysplex hardware and software, subsystem software, connectivity to the IMSplex, and a decision about where applications will run. Of course the decision may be, and should be (if practical) that each application runs on every IMS image.

You should also include a plan for degraded mode processing if not all components are available. For example, if the LPAR containing one of your IMS control regions is unavailable for an extended period, what will you do? Will applications have to be moved, will end-users wait or log back on to another IMS? How will those end-users reestablish their status on the new IMS?

7.1.1 Data sharing configuration

The following are components of a data sharing configuration, which you must consider in developing your data sharing configuration.

Online control regions

How many online control regions will you have, including DB/DC as well as DBCTL? For each online IMS control region address space, you will also have (of course) a DLI address space, a DBRC address space., and multiple dependent regions.

Fast Database Recovery regions

We also recommend a Fast Database Recovery (FDBR) address space to perform dynamic backout and release IRLM locks if IMS fails. This FDBR address space should be on another LPAR to enable it to work even in the case of an OS/390 failure.

BMP regions

With data sharing, BMPs can be scheduled by any IMS in the data sharing group with access to the data required by the BMP. The IMS control regions must specify an IMSGROUP parameter, and the BMPs must use this as the IMSID. If these BMPs are accessing DB2, they should code the DB2 group attach name in their SSM PROCLIB members.

Batch regions

If you have batch jobs for which, prior to data sharing, you DBRed the databases from the non-sharing online subsystem, you may want to consider converting them to BMPs. BMPs are much easier from both a management perspective and from an availability perspective. They share the same logs as the other online IMS applications, and no matter how the BMP might fail, the IMS control region can dynamically back out the databases (not always true for batch jobs). If you decide to keep them as batch jobs, then you should identify on which LPARs they will run and be sure that there is an IRLM available on that LPAR. Note, however, that CPU time charged to the BMP will probably be higher than that charged to a non-sharing (or even a sharing) batch region and may affect any CPU-based “charge-back” algorithms you might be using with your clients.

Shared databases

It may be that not all databases are shared. The section titled “What to share?” on page 63 identified some reasons why some databases will not be shared. DBRC registration should be done based on how these databases will or will not be shared. If not all databases are shared, then special attention must be given to connectivity and BMP scheduling. If a database is not shared at the block level, register it in the RECONs at SHARELVL(1) to allow non-update access, such as concurrent image copy, to run while the database is online.

IRLMs

For each data sharing group, you should plan to have one, and only one, IMS IRLM (IMS and DB2 can not share the same IRLM) on each LPAR where there is a data sharing IMS in the same data sharing group. Each IRLM requires a different IRLMID, but the IRLM name (IRLMNM parameter) should be the same for all IRLMs. This allows all IMSs that are running on the same LPAR to connect to the local IRLM. This is particularly important when IMS, after an LPAR failure, is being restarted (perhaps by ARM) on another LPAR where there is already an IMS and an IRLM. In the event of a system failure, you should not let ARM perform a cross-system restart of the IRLM on a system which already has an IRLM in the same data sharing group.

Data sharing structures

There are several Coupling Facility structures required with block level data sharing. These structures must be defined in the active CFRM policy:

- IRLM lock structure

  One is required for the data sharing group. You may define this structure as duplexed in the CFRM policy.

- OSAM cache structure

  One is required for the data sharing group. An OSAM structure is required even if you have no OSAM databases. This structure does not support duplexing.

- VSAM cache structure

  One is required for the data sharing group. A VSAM structure is required even if you have no VSAM databases. This structure does not support duplexing.

- Shared VSO structures

  One is required for each DEDB area registered to DBRC as VSO and SHARELVL(3). You may optionally define a *duplicate* structure for each area. That duplicate structure should not be confused with the *duplexed* structure support provided by z/OS, which is an alternative for Fast Path managed duplicate structures.

CICS and database control

If your IMS control region is performing database control functions for a CICS application owning region (AOR), then that CICS and IMS control region must be on the same OS/390 image. If you are using ARM for cross-system restart, then the IMS control region and the CICS AOR should be in the same restart group.

IMS and DB2

If IMS applications access DB2 databases, then the IMS control region and DB2 must be on the same OS/390 image. If you are using ARM for cross-system restart, then the IMS control region and DB2 should be in the same restart group.

Open database access (ODBA)

ODBA allows an application running in a non-IMS dependent region to connect to an IMS control region, schedule a PSB, and issue DL/I calls. Some examples of ODBA clients include applications scheduled as DB2 stored procedures and Enterprise JavaBeans (EJBs) running in a WebSphere Application Server address space. These ODBA clients must be on the same OS/390 image as the IMS control region.

Sample data sharing configuration

Figure 7-1 shows a possible data sharing configuration with IMS connected to CICS and DB2. CICS uses the database resource adapter (DRA) interface to connect to the Database Control function of the IMS control region. The ODBA clients could be DB2 stored procedures running in a Workload Manager (WLM) address space, or Enterprise JavaBeans (EJBs) running in a WebSphere Application Server address space. Each of the above invoke the DRA or ODBA interface to schedule PSBs in IMS and then issue database calls. The applications themselves run in these other address spaces, not in IMS dependent regions.

The transactions may be from a VTAM network, a TCP/IP network (via IMS Connect and OTMA), or from the MQSeries® bridge (via OTMA). IMS is sharing the databases (and the RECONs) with IMS batch jobs and with other IMS data sharing configurations on other OS/390 images. Also shown (partially) is a Fast Database Recovery (FDBR) address space that would be supporting an IMS running on one of the other images. LCV is the local cache vector used for buffer invalidation notification.

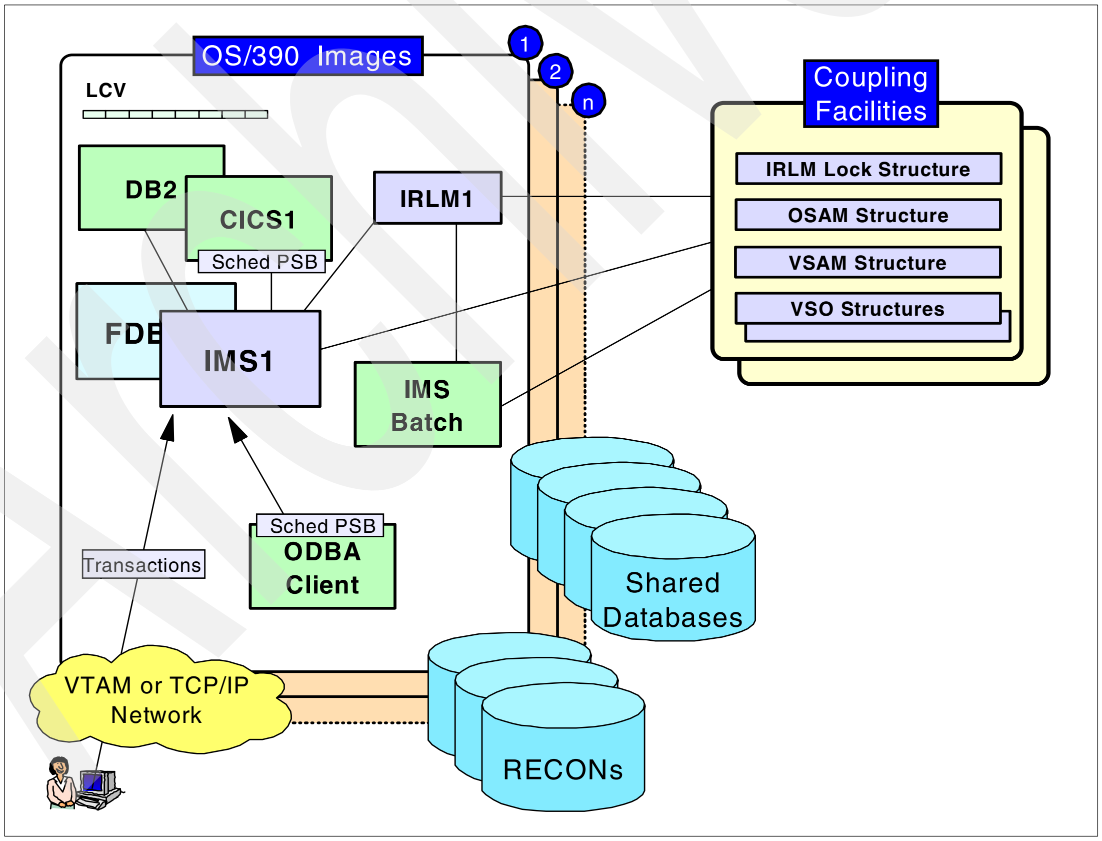


Figure 7-1 Data sharing configuration

7.1.2 Shared queues configuration

Shared queues makes it easier to distribute the workload across the IMSplex and data sharing group and makes the issues of connectivity easier to implement while the careful planning is still important. Shared queues *adds* its own components to the IMSplex configuration.

IMS control region

You have several choices for your shared queues configuration:

- Cloned configuration

  The region type can be DB/DC or DC Control, although DC control regions do not have access to IMS databases. All IMSs are cloned - that is, their system definitions are identical. This is the easiest to manage and the most flexible. Any user can log on to any IMS and get access to any transaction and any database. You need only a single IMS system definition and can make any required differentiations (for example, the MTO node name and LTERM name) in the execution parameters and PROCLIB members.

- Front-end/back-end configuration

  Front-end systems can be DB/DC or DC Control. The front-ends provide network services and queue transactions to the shared queues. Only the front-ends would join the VTAM generic resource group and the OTMA XCF group. The back-end systems must be DB/DC. They require the transaction manager services not offered with DB Control and the database services not offered with DC Control. The back-ends retrieve messages from the shared queues and process them, putting the response back on the shared queues for the front-end to deliver. The advantage of this is that back-ends can be started and stopped as needed to reflect changing workloads without affecting any end-users.

- Hybrid configuration

  This configuration can consist of one or more front-ends which also process transactions, and zero or more back-ends which only process transactions. All would have to be DB/DC systems. The IMSs do not have to be clones. If they are not, then some transactions may have to be processed on a subset of the IMS systems (that is, wherever they are defined). One use of this might be to let the front-end systems process transactions requiring the best response times, maximizing the efficiencies of local processing, while the back-ends handle overflow from the front-end transactions as well as other transactions not requiring such high levels of response. Another possibility might be to let the Fast Path EMH transactions process in the front-end with a sysplex processing code of *LOCAL ONLY* or *LOCAL FIRST*, and send all full function transactions to the back-end(s).

Common Queue Server (CQS)

You need one CQS on each OS/390 image where there is an IMS front-end or back-end control region in the shared queues group. Beginning with IMS Version 7, one CQS can provide support for multiple control regions on the same image. If your IMS Version 8 systems are running with a Common Service Layer (CSL), one CQS (per OS/390 image) can provide services for both shared queues structure access and resource structure access. In a mixed environment consisting of IMS Version 7 and IMS Version 8, each version of IMS requires its own matching version of CQS. That is, the version of CQS delivered with IMS Version 8 cannot provide shared queues services for an IMS Version 7 control region.

Shared queues structures

CQS uses one or more list structures to contain the shared queues (messages and control information). They must be defined in the active CFRM policy:

- Primary full function shared queues structure

This structure is *always required* in a shared queues environment.

- Overflow full function shared queues structure

  This structure is *not required but is recommended* to handle conditions where the primary structure is reaching a user-defined full threshold to keep that primary structure from filling up.

- Primary Fast Path EMH shared queues structure

  This structure is used to queue global Fast Path EMH transactions. Note that even if you do not have any EMH transactions defined, if your system includes Fast Path at all (FPCTRL macro in the system definition), then this structure is required.

- Overflow Fast Path EMH shared queues structure

  This has the same purpose for EMH transactions as the full function overflow structure has for full function transactions. It is never required, even with Fast Path in the system.

You may also define any of these structures as duplexed (requires z/OS 1.2 or later). However, since they are recoverable, and there is some additional overhead for duplexing, this may not be a good choice.

System Logger

There must be a System Logger (IXGLOGR) address space on each OS/390 image where there is a CQS address space supporting shared queues. If you have a Parallel Sysplex, this address space probably already exists. A single System Logger address space provides logger services to all users on the same OS/390 image. System Logger users such as CQS write log records to a log stream.

Log streams

CQS requires a log stream for updates to the full function shared queues structures, and a second log stream for Fast Path EMH if an EMH shared queues structure is defined. The log streams must be defined in the LOGR policy.

Logger structures

One or two structures must be defined in the active CFRM policy for use by the System Logger for CQS's log stream(s). The LOGR policy associates the log streams with the logger structures. These structures are:

- Full function logger structure

  CQS requires a log stream for updates to the full function shared queues structures. The System Logger requires a logger structure for the log stream.

- Fast Path EMH logger structure

  CQS requires a log stream for updates to the Fast Path EMH shared queues structures (if Fast Path is enabled). The System Logger requires a logger structure for the log stream.

Like the shared queues structures, these can be duplexed. However, the System Logger does its own duplexing and structure duplexing should not be required.

Sample shared queues configuration

Figure 7-2 shows an example of a shared queues configuration with two IMSs. There can, of course, be as many as 32 (the limit on the number of connectors to a list structure). The System Logger will use either a data space (recommended) or a staging data set for duplexing that logger's portion of the data stream (for recovery purposes in the logger structure fails). The LNVs are List Notification Vectors for notifying CQS when a registered queue transitions from empty to non-empty.

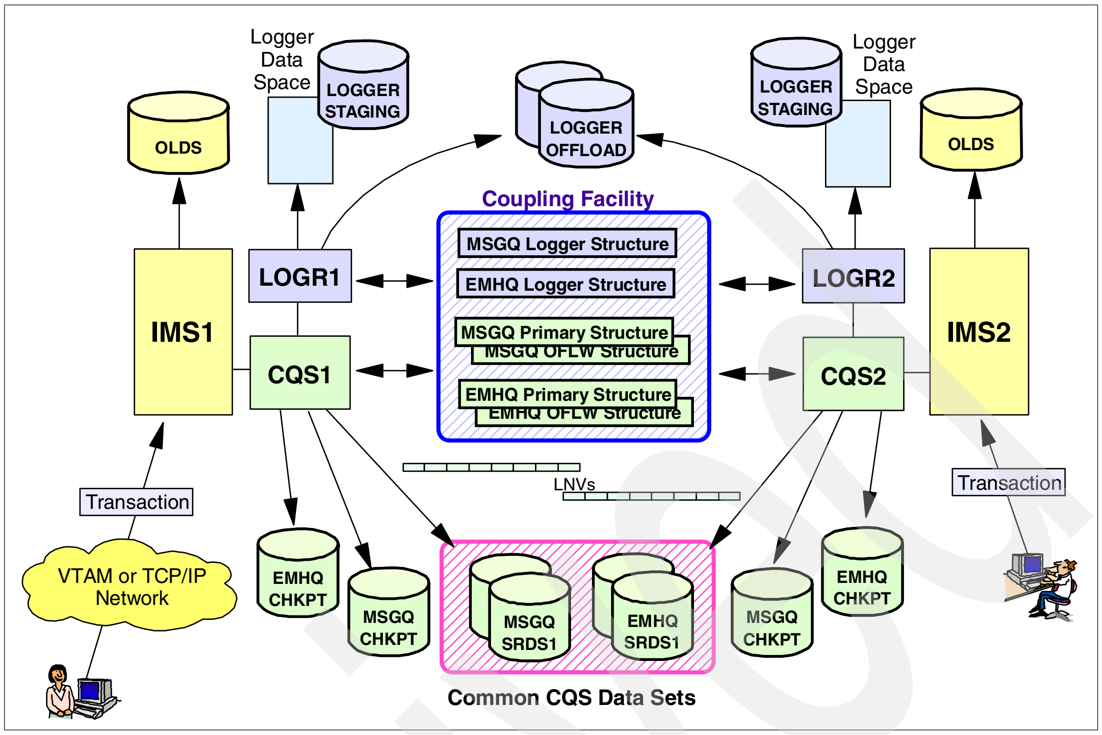


Figure 7-2 Shared queues configuration

7.1.3 Common Service Layer configuration

The Common Service Layer, available beginning with IMS Version 8, is intended to improve the systems management and operations of the IMSplex. The following components will be a part of your configuration. More details of configuration information can be found in 5.4, "Configuration planning" on page 123.

Structured Call Interface

There must be an Structured Call Interface (SCI) address space on every OS/390 image where there is any IMSplex member. Do not forget that the TSO SPOC, IMS Connect when providing services to the IMS Control Center, and any user- or vendor-written automated operations (AO) clients that join the IMSplex and so require a local SCI. If you are using automatic RECON loss notification, then you need an SCI address space wherever you run DBRC, including batch jobs and utilities that use DBRC.

Resource Manager

Only one Resource Manager (RM) address space is required in the IMSplex. If you do not define a resource structure, only one RM is allowed in the IMSplex. If you do have a resource structure, then at least two RMs are recommended for availability and performance. They can be on any image where an SCI exists.

Operations Manager (OM)

Like RM, you need at least one OM in the IMSplex, however, two or more are recommended for availability. Performance is probably not a concern with OM since it is not likely you would be entering commands fast enough to cause any problems. SCI will route any input commands to the target IMS and route any responses to the AO client.

Resource structure

The resource structure is optional. Sysplex terminal management, a function of IMS Version 8, requires a resource structure and shared queues. Although not required for global online change, If you have one, then consistency checking for the online change data sets is enabled. Note that the structure is *not required* for global online change, only for the resource consistency checking feature.

TSO single point of control

Any authorized user can start the TSO SPOC ISPF application and enter commands to the IMSplex. There is no limit on how many there are, however, they must be in the same Parallel Sysplex and must be on an OS/390 image with an SCI address space.

DB2 Admin Client with IMS Control Center

This tool is provided with DB2 and can be obtained separately even without DB2. It will run in several environments, including Windows, AIX, and Linux, and uses TCP/IP to enter commands through IMS Connect to the IMSplex. IMS Connect registers with SCI as a member of the IMSplex.

User or vendor supplied automation

Your environment may include either user-written or vendor-written automated operations software. These programs or execs must register with SCI as a member of the IMSplex.

Connecting to the CSL environment

End-users do not connect to any of the CSL address spaces. Transactions are entered and IMS itself invokes CSL services where required, such as for sysplex terminal management. However, there is some connectivity planning required for operations. Figure 7-3 shows a single image IMS running with a Common Service Layer. All the CSL address spaces are shown, although only SCI is required on every image. Operations users, such as the TSO SPOC, connect by registering with SCI and then use the OM interface to submit commands to IMS and to retrieve the responses. This is also true of any user-written or vendor-written automation programs that use the OM interface. The DB2 Admin Client, which includes an IMS Control Center, accesses the IMSplex through IMS Connect, which registers with SCI and then forwards commands to OM and returns responses from OM.

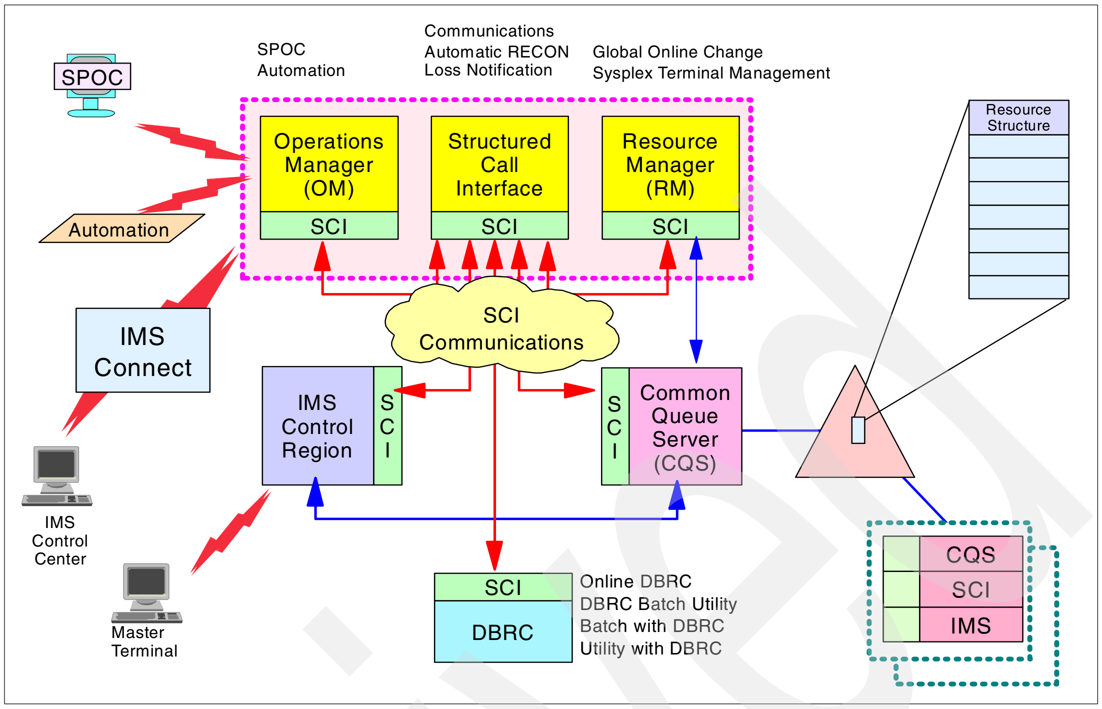


Figure 7-3 Common service layer configuration

7.1.4 Getting the work to IMS

Your configuration plan should include how to get the work to each of the IMSs in the IMSplex. When there was a single IMS, it was easy to decide to which IMS an end-user should connect, or to which IMS a BMP should connect. When your network was entirely VTAM, it was even easier. When there are multiple IMSs, and both VTAM and TCP/IP networks, as well as other connection architectures such as ODBA, different users may connect to different IMSs through different paths and protocols. Although with shared queues it is not absolutely necessary for network users to connect to different IMSs in the shared queues (and data sharing) group, it will probably be desirable for better availability and workload balancing.

Connectivity to the IMSplex

As shown briefly in the preceding sections, your configuration must include a path for end-users to *submit transactions or database requests* to one or more of the IMSs in the IMSplex. Some of these access paths will be transaction entry - others will schedule a PSB through the database resource adapter (DRA) interface or the open database access (ODBA) interface. *In all cases, IMS PSBs are scheduled, under which database calls can be issued.* Chapter 6., “Planning for IMSplex connectivity” on page 131, discusses several options for connecting to the IMSplex. Some of the techniques include the following protocols, services, and products.

Network connection to submit an IMS transaction and schedule an IMS PSB:

- VTAM:
  – LU0, LU1, LU2 (including emulators such as Telnet 3270)
  – LU6.1 (ISC)
  – LU6.2 (APPC)
  – MSC

- TCP/IP:
  - Network Dispatcher/WebSphere Edge Server
  - Sysplex Distributor/VIPA
  - Linux z/OS Web server
  - IMS Connector for Java
  - IMS Connect, XCF, OTMA
  - Telnet 3270 (looks like VTAM LU2 to IMS)
- WebSphere MQ (from APPC client or TCP/IP sockets client):
  - MQ Bridge, XCF, OTMA
  - MQ Adapter, listener BMP
- Callable OTMA interface
  - Any user-written application which uses the callable OTMA interface

Database access using the database resource adapter (DRA) to schedule a PSB

- CICS Transaction Server to DB Control

Database access using open database access (ODBA) to schedule a PSB:

- DB2 stored procedures running in Workload Manager address space
- Enterprise JavaBeans (EJB) running in WebSphere Application Server address space
- Other user-written or vendor-written program running in OS/390 address space

7.2 Security planning

There are several aspects of security which have different connotations in an IMSplex. Examples are joining the IMSplex, connecting to structures, and submitting commands through the Operations Manager interface. Shared queues introduces some security issues concerning authorization processing when a transaction entered on one IMS executes on another. The migration plan should include tasks which identify how security will be implemented in the IMSplex.

7.2.1 Data sharing security

When you implement data sharing, you will have to allocate your database data sets with DISP=SHR (and VSAM SHAREOPTIONS(3,3) for VSAM). You may have had DISP=OLD when not data sharing, which would have provided some minimal level of security.

Database security is generally provided through DBRC authorization processing and application (transaction) security, however, you may be providing data set security for your database data sets using RACF or equivalent security product. Data sharing should not affect this other than you need to be sure to authorize all sharing subsystem to access the data sets.

7.2.2 Shared queues security

There are several issues with security that must be considered when developing your migration plan for shared queues.

Transaction authorization security

For transactions entered from non-APPC and non-OTMA sources, there is no change to transaction authorization processing on a front-end IMS. When a transaction arrives on the front-end, authorization is done *before it is put on the shared queues*. When the back-end processes it, it assumes it is authorized.

There are some cases, however, where the back-end must perform some type of security checking. Specifically, when an application issues a call that would require additional security, such as:

- CHNG call to a transaction destination
- ISRT SPA call with a different transaction code (deferred conversational program-to-program switch)
- AUTH call

How these cases are treated depends on whether or not you have coded a *Build Security Environment Exit* (DFSBSEX0). The exit gets invoked during transaction scheduling and determines how the above described CHNG, ISRT, and AUTH calls are handled in a back-end IMS. The exit has several options:

- Do not build an ACEE in the dependent region during scheduling. Build an end-user ACEE in the dependent region later if needed for CHNG, ISRT, or AUTH calls. Use the ACEE for authorization processing.
- Build an end-user ACEE in a dependent region before the message given to application. Use the ACEE for authorization processing.
- Do not build an end-user ACEE at all. Call SAF for transaction authorization if required. Use dependent region ACEE (if one exists) or control region ACEE (if not). Does not use end-user ACEE. Call security exits (DFSCTRN0 and DFSCTSE0) if they exist.
- Do not build ACEE at all. Bypass call to SAF. Call security exits if they exist.
- Do not build ACEE at all. Bypass all security, including security exits.
- Do not build an ACEE at all. Call SAF for transaction authorization if needed. Use dependent region ACEE (if one exists) or control region ACEE (if not). Does not use end-user ACEE. Bypass calls to security exits.

When the DFSBSEX0 exit is not coded, then when security is required in the back-end, an ACEE will be built for each call, and then it will be deleted. If a transaction on the back-end issues five CHNG calls, then the ACEE will be built and deleted five times.

If DFSINSX0 dynamically defines a transaction in the F-E, RACF security checking will be performed, but no SMU security checking will be done - either in the F-E or in the B-E.

APPC and OTMA transaction authorization security

Transactions which originate from APPC or OTMA sources are eligible for global shared queues processing, with some restrictions prior to IMS Version 8. The level of security can be specified through the APPCSE or OTMASE parameters in DFSPBxxx, and can be overridden by the /SECURE APPC or /SECURE OTMA command. Unlike transactions entered from other sources, DFSBSEX0 is not called in the back-end system. Whatever security is active for the system on which the transaction executes is in force. If it is a back-end system, then the ACEE will be created at the time it is required and then deleted as soon as the call completes.

Command authorization security

It is also possible for a transaction running on the back-end to issue calls that issue IMS commands. These transactions are often referred to as automated operator programs (AOP) and may be used to help control the local IMS environment. Calls types are:

- ICMD call

Security for this call is governed by the AOIS parameter, which can request SAF, Command Authorization Exit (DFSCCMD0), both, or none. When SAF is requested, SAF will be called on the back-end using an ACEE. If the ACEE already exists because DFSBSEX0 has requested it, then it will be used, Otherwise, an ACEE will be built and then deleted at completion of authorization processing. If the Command Authorization Exit (DFSCCMD0) has been requested, then it will be called on the back-end the same as if it were executing on the front-end.

- CMD calls

  For CMD calls, which use only SMU security, this will also be checked in the back-end, but that back-end must be using a MATRIX data set that has the appropriate security defined.

Shared queues structure security

You may provide security for the *primary* shared queues structures using a SAF product (such as RACF). To do this, you must RDEFINE the structure to RACF in the FACILITY class and then PERMIT access it. The user that you authorize, however, is not the CQS address space, it is the user ID of the IMS control region. CQS will issue the RACROUTE call to RACF to authorize whatever control region(s) is connected to access the shared queues structures. Example 7-1 shows a sample definition of shared queues structure security. The overflow structure is not included. CQS assumes that if a user has access to the primary structure, that same user has access to the overflow structure, and so does not make a call to SAF. Note that the resource name begins with the constant “CQSSTR.” followed by the structure name.

Example 7-1 Securing the shared queues structures

```
RDEFINE FACILITY CQSSTR.primary-msgq-strname UACC(NONE)
PERMIT CQSSTR.primary-msgq-strname CLASS(FACILITY) ID(imsid1) ACCESS(UPDATE)
PERMIT CQSSTR.primary-msgq-strname CLASS(FACILITY) ID(imsid2) ACCESS(UPDATE)

RDEFINE FACILITY CQSSTR.primary-emhq-strname UACC(NONE)
PERMIT CQSSTR.primary-emhq-strname CLASS(FACILITY) ID(imsid1) ACCESS(UPDATE)
PERMIT CQSSTR.primary-emhq-strname CLASS(FACILITY) ID(imsid2) ACCESS(UPDATE)
```

Log stream security

CQS is the user of the log stream. You can provide security for the log stream(s) by defining it to RACF in the LOGSTRM class with universal access of NONE, and then authorizing your CQSs to access the log streams with UPDATE access. The System Logger will issue the appropriate authorization calls to SAF. You must also provide READ access for any readers of this log stream, such as someone using the DFSERA10 log print utility, or perhaps another utility or tool. Example 7-2 shows a sample of defining log stream security.

Example 7-2 Securing the log stream

```
RDEFINE LOGSTRM msgq.logstreamname UACC(NONE)
PERMIT msgq.logstreamname CLASS(LOGSTRM) ID(cqsuserid1) ACCESS(UPDATE)
PERMIT msgq.logstreamname CLASS(LOGSTRM) ID(cqsuserid2) ACCESS(UPDATE)
PERMIT msgq.logstreamname CLASS(LOGSTRM) ID(readuserid1) ACCESS(READ)
PERMIT msgq.logstreamname CLASS(LOGSTRM) ID(readuserid2) ACCESS(READ)
```

7.2.3 Common Service Layer security

There several new security options when running in a CSL environment. They must be considered and plans put in place to implement them.

Structured Call Interface

The Structured Call Interface (SCI) address spaces determine who is allowed to join the IMSplex and participate in the CSL services. You may define the IMSplex-name in the RACF FACILITY class and then control who (what user IDs) are allowed to register with SCI and thereby join the IMSplex. This process is described in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929. The following is an example:

Example 7-3 Securing access to the IMSplex

```
RDEFINE FACILITY CSL.CSLPLXO UACC(NONE)
PERMIT CSL.CSLPLXO CLASS(FACILITY) ID(IMS1) ACCESS(UPDATE)
PERMIT CSL.CSLPLXO CLASS(FACILITY) ID(OM1) ACCESS(UPDATE)
PERMIT CSL.CSLPLXO CLASS(FACILITY) ID(RM1) ACCESS(UPDATE)
```

You can, of course, identify all users as members of a group, and then just PERMIT the group access to the IMSplex. Note that UPDATE access is required.

- Do you want registration security for SCI?

  If the answer to this is no, skip to the next section. Remember, however, that without registration security, any program can register with SCI as an AOP and issue commands to the IMSplex. While the unwanted user may still be thwarted by command security, it is probably best not to let that user join in the first place.

- What are the user IDs of the address spaces that will register with SCI?

  If you do want registration security, then you have to define the IMSplex name in the RACF FACILITY class and then PERMIT all valid users to register with that IMSplex name. Or add all users to a RACF group and PERMIT the group. Every IMS control region, CQS, OM, RM, and AOP user must be permitted.

  – Are you going to implement automatic RECON loss notification?

    If you plan to implement ARLN, then you must permit the user IDs of all the DBRC instances that will be registering with SCI access to the IMSplex name.

  – Will you be using the IMS Control Center?

    If you are, then you must register the user IDs of all the IMS Connect address spaces that the IMS Control Center will be using.

  – Will you be writing your own automation programs?

    If you do, then these programs must pass a user ID to SCI when they register, and that user ID must be permitted access to the IMSplex in RACF. For the TSO SPOC, this will be the user ID of the TSO user.

Operations Manager (OM)

The Operations Manager forwards commands entered from an OM AO client to one or more IMSs. You have the choice of *if, where, and how* you want command authorization to be performed. Authorization for the new IMSplex commands can ONLY be done in OM. IMS will not perform command authorization for these new commands. Classic command authorization can be performed in IMS, in OM, or both.

- Will you be performing command authorization for commands entered through the OM interface?

  If the answer to this is no, then skip to the next section.

- Where will IMS command security be done? OM? IMS? Both?

You can decide whether you want command authorization for commands entered through OM to be done by OM, by IMS, or by both OM and IMS. Remember, however, that IMS will not do command authorization processing for the new IMSplex commands.

An advantage to performing classic command security in OM is that you can be more specific in defining the command to RACF. When performed in IMS, you can only specify the command itself (for example, the STOP command). When performed by OM, you can protect not only the command but also the resource type. For example, you can define STOP DB and STOP NODE independently. Example 7-4 shows how IMS commands can be defined and secured when entered from OM.

Example 7-4 Securing IMS commands entered through OM

```
RDEFINE OPERCMDS IMS.CSLPLX0.STO.DB UACC(NONE)
RDEFINE OPERCMDS IMS.CSLPLX0.STO.NODE UACC(NONE)
PERMIT IMS.CSLPLX0.STO.DB CLASS(OPERCMDS) ID(USER1) ACCESS(UPDATE)
PERMIT IMS.CSLPLX0.STO.NODE CLASS(OPERCMDS) ID(USER2) ACCESS(UPDATE)
```

Another advantage is that it is more efficient to do it in OM rather than send the command across the SCI link to be done by IMS. This is true especially since some commands may be routed to multiple IMSs. If security is done in OM, it need be done only once.

- What type of command security will be used in OM? SAF (RACF)? Exit? Both?

  Here you have the same kind of choice you have with command authorization in IMS. You can use SAF (RACF or equivalent), a command authorization exit, or both. When you choose both, the RACF security will be done before the exit. The exit can override the decision made by RACF.

  The type of security you want must be defined in the OM initialization PROCLIB member (CSLOIxxx) with the following parameter:

  ```
  CMDSEC=N|E|R|A (the default is none)
  ```

  If you decide to use a command authorization exit in OM (E or A), then you must define that exit in the user exit list PROCLIB member identified in the BPECFG PROCLIB member in use by OM. For example:

  ```
  BPECFG PROCLIB MEMBER
     EXITMBR=(SHREXIT0,OM)
  SHREXIT0 PROCLIB MEMBER
     EXITDEF=(TYPE=SECURITY,EXITS=(OMSECX),COMP=OM)
  ```

  You then must write the exit (OMSECX in this example) and put it in the OM STEPLIB. If you have multiple OMs, do not forget to put it in all STEPLIBs.

- What user IDs will be authorized for command security in OM?

  Since RACF security, and probably any exit security, is based on user ID, then you need to determine what user IDs can issue what commands and define the commands and permit the users in RACF, as shown above.

- What OM user IDs must be authorized to join the IMSplex (register with SCI)?

  If you have decided to use SCI registration security, then the user ID of each OM address space must be permitted to access the IMSplex.

Resource Manager

Like the Operations Manager, the Resource Manager (RM) must be authorized to join the IMSplex if the decision was to require authorization.

- Is authorization required to connect to SCI?

  If the answer is no, skip to the next section.

- What RM user IDs must be authorized to join the IMSplex (register with SCI)?

  You must identify the user IDs for each RM that is in the IMSplex and either add that user ID to the IMSplex GROUP, or permit that user to access the IMSplex.

Resource structure

The resource structure is optional, but required if you want to enable sysplex terminal management. If you do not have a resource structure, skip to the next section.

- Will connection to the resource structure be protected by RACF?

  Connection to the resource structure can be protected by defining the structure in the RACF FACILITY class. Access to the resource structure can be permitted either by individual user ID or by group. The required access is UPDATE.

- What RM user IDs will be connecting to the resource structure?

  The resource manager does not connect directly to the resource structure. RM uses CQS to access the structure much the same way IMS uses CQS to access the shared queues structures. The same CQSs that support IMS access to the shared queues can support RM access to the resource structure.

  You must determine the user IDs of the RMs that will connect to the resource structure and either add them to a group which is authorized access to the structure, or permit each user ID access to the structure.

- Besides RM, what other applications (for example, vendor products) must be authorized to connect to the structure?

  CQS, acting as a server for RM, is the only address space in the IMSplex requiring access to the resource structure. However, since the interface is open, any user or vendor can write a program which accesses the resource structure through CQS. The user IDs of these address spaces must also be defined to RACF and permitted access.

CQS

CQS can be a server to IMS for shared queues and a server to RM for the resource structure. You must have a CQS on every system where there is an RM requiring access to the resource structure. You must also have a CQS on every system where these is an IMS requiring access to the shared queues. These can be the same CQSs.

- What CQS user IDs must be authorized to join the IMSplex (register with SCI)?

  Identify the CQS user IDs and permit them access to the IMSplex.

- Is resource structure connection security required?

  Identify the CQS user IDs and permit them access to the resource structure (See 7.2.4, “Structure security” on page 174).

IMS control region

There are two CSL security issues for the IMS control region.

- What control region user IDs must be authorized to join the IMSplex (register with SCI)?

  Like all IMSplex address spaces that register with SCI, the user IDs of the IMS control regions must be permitted access to the IMSplex, either individually or by being a member of an authorized group.

- Where is OM command security being done? OM? IMS? Both?

  Earlier you decided whether, and how, commands would be authorized in OM by coding a parameter (CMDSEC=) in the OM initialization PROCLIB member. You must also tell IMS whether, and how, to authorize commands *received from OM*. IMS treats these commands differently than those received from traditional sources. For commands received from OM, a CMDSEC parameter in PROCLIB member DFSCGxxx tells IMS whether or not, and how, to do command authorization.

If commands are authorized in OM, there should be no need to reauthorized them in IMS. In this case, you would code CMDSEC=N in DFSCGxxx.

Remember that this applies only to classic commands. IMS will not do authorization processing for the new IMSplex commands. That can only be done in OM.

7.2.4 Structure security

Structure security is generally *provided by the connector* to the structure. For example, CQS will provide access authorization to the shared queues structures based on the user ID of the IMS control region. However, it is possible for a program to issue the IXLCONN macro to connect to a shared queues structure without going through CQS. To provide the additional security to prevent unauthorized programs from accessing the shared queues, or any other, structures, a different profile must be created in RACF. If this is used, then XES itself will call RACF to authorize connection to a structure.

To do this, you must define the structure resource in the FACILITY class and then PERMIT the connectors to access the structure. The resource name, in this case, has a high-level qualifier of IXLSTR instead of CQSSTR. Connectors to be authorized must include all of the CQSs in the IMSplex plus any other programs which will be connecting to the structure directly (not through CQS); for example, IMS Queue Control Facility (QCF) product. Example 7-5 shows how you might provide access security for the primary and overflow full function shared queues structures.

Example 7-5 Securing structures

```
RDEFINE FACILITY IXLSTR.primary-msgq-strname UACC(NONE)
PERMIT IXLSTR.primary-msgq-strname CLASS(FACILITY) ID(cqsid1) ACCESS(UPDATE)
PERMIT IXLSTR.primary-msgq-strname CLASS(FACILITY) ID(cqsid2) ACCESS(UPDATE)
PERMIT IXLSTR.primary-msgq-strname CLASS(FACILITY) ID(qcfuser1) ACCESS(UPDATE)

RDEFINE FACILITY IXLSTR.overflow-msgq-strname UACC(NONE)
PERMIT IXLSTR.overflow-msgq-strname CLASS(FACILITY) ID(cqsid1) ACCESS(UPDATE)
PERMIT IXLSTR.overflow-msgq-strname CLASS(FACILITY) ID(cqsid2) ACCESS(UPDATE)
PERMIT IXLSTR.overflow-msgq-strname CLASS(FACILITY) ID(qcfuser1) ACCESS(UPDATE)
```

Similar security can be defined for the other structures, such as the OSAM, VSAM, VSO, IRLM, and resource structures.

7.2.5 User IDs for started procedures

Most of the security described above uses the user ID of the address space as a basis for authorization. When the address space is a JES submitted job, then the user ID is defined by the USER= parameter on the JOB card. But when it is a start procedure, as most of these are, then the user ID must be defined in the RACF STARTED class. Example 7-6 shows a sample definition for several IMSplex started procedures with user IDs.

Example 7-6 Sample RACF definitions for IMSplex started procedures

```
RDEFINE STARTED *.IMSAPROC STDATA(USER(IMSA) GROUP(IMSPLEX1) PRIVILEGED(NO) TRUSTED(NO))
RDEFINE STARTED *.CQSAPROC STDATA(USER(CQSA) GROUP(IMSPLEX1) PRIVILEGED(NO) TRUSTED(NO))
RDEFINE STARTED *.SCIAPROC STDATA(USER(SCIA) GROUP(IMSPLEX1) PRIVILEGED(NO) TRUSTED(NO))
RDEFINE STARTED *.RMAPROC STDATA(USER(RMA) GROUP(IMSPLEX1) PRIVILEGED(NO) TRUSTED(NO))
RDEFINE STARTED *.OMAPROC STDATA(USER(OMA) GROUP(IMSPLEX1) PRIVILEGED(NO) TRUSTED(NO))
RDEFINE STARTED ** STDATA(USER(=MEMBER) GROUP(SYSPLEX1) PRIVILEGED(NO) TRUSTED(NO))
```

The last statement defines a generic started procedure. This is used when a started procedure is not defined in the STARTED class (or in the RACF started procedure table) and says to use the member name as the user ID.

7.3 Implementation and operational planning

Implementation is discussed in detail in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929. In this section, we just highlight the major activities in planning for the implementation of the IMSplex.

Implementation planning should include both *preparing the environment* for the IMSplex, and the *actual implementation* of the IMSplex in that environment to operational status.

The preparation phase is where you get the S/390, OS/390 or z/OS, and IMS environments ready for cutover to operational status. This includes hardware, software, libraries, JCL procedures, operational procedures, and any changes required to applications and databases. This goes on while the current environment is still operational and it can get very confusing keeping libraries straight, installing new hardware and software, preparing changes to procedures, changing applications, and so on. A well documented plan for how this will occur can save a lot of problems later.

Define the tasks to implement your configuration (including any intermediate configurations) *through all phases of the migration*. Plans for the implementation phase must include (if necessary) termination of the existing environment and initiation of the new environment (for example, IMS shutdown and a cold start). Do not forget to include the end-users in your implementation plan.

Although this document does not prescribe a format for the implementation plan, the following elements should be included:

Tasks	What must be done?
Responsibility	Who is responsible to see that it gets done?
Schedule	When must it be done - start/complete/drop-dead dates?
Dependencies	Is any task a prerequisite to another task?
Duration	How long should each task take?
Status	A mechanism for monitoring progress?

7.3.1 Preparing for implementation

At this point, you have made your decisions about what IMSplex functions you need to meet your objectives and what configuration will best provide those functions. If you followed this guide, you reached these decisions by answering a series of questions. Based on the answers to these questions, an implementation plan can now be developed to enable the IMSplex, or whatever part of it this plan includes. The implementation plan is a set of tasks that must be performed to take you from your current environment to your target environment. Identified below are *planning tasks* for creating an implementation plan. The result of performing these tasks should be a set of *implementation tasks*. For example, a planning task may be to identify PROCLIB members which must be created or updated. The implementation task would be to create or update those members. Implementation is discussed in detail in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and*

Operations, SG24-6929. In this section, we just highlight the major activities in planning for the implementation of the IMSplex.

In the following list are some of the tasks required to prepare your environment for IMSplex implementation and operation. This is only a selection of tasks, some of which (but not all) are described in more detail below, and all of which are described in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929.

- Prepare the OS/390 or z/OS environment for the IMSplex:
  - Program properties table
  - Couple data sets
  - SYS1.PROCLIB (and concatenations)
- Identify new and existing procedures and parameters which must be created or updated for the IMSplex:
  - IMS control region environment
    - Control region (CTL, DBC, DCC), DBRC, DLISAS, dependent regions (MPP, IFP, BMP, JMP, JBP), IMSRDR, IMSWTxxx, FDBR, XRF alternate
  - IRLM
  - CQS
  - CSL (SCI, OM, RM, TSO SPOC, other AOPs)
  - IMS Control Center and IMS Connect
  - IMS batch
  - IMS utilities
- Identify new and existing PROCLIB members and/or execution parameters that must be created or updated:
  - IMS control regions
    - DFSPBxxx, DFSDCxxx, DFSVSMxx, DFSFDRxx, DFSSQxxx, DFSCGxxx
  - IRLM
    - Execution parameters
  - Base Primitive Environment (BPE)
    - BPE configuration
    - BPE user exit list
  - CSL address spaces (SCI, OM, RM)
    - BPECFG, CSLSIxxx, CSLOIxx, CSLRIxxx
  - Common Queue Servers (CQS)
    - BPECFG, CQSIPxxx, CQSSLxxx, CQSSGxxx
- Identify new and existing exits that must be coded or modified:
  - IMS control region exits
    - IMS Initialization, Logon/Logoff, Signon/Signoff, Command Authorization, Output Creation, Conversational Abnormal Termination, others
    - Identify existing exits which use callable control block services for FIND or SCAN. Can they handle the GLOBAL default when sysplex terminal management is active?
  - BPE user exits
    - Initialization/Termination, Statistics
  - SCI user exits

    - Client Connection, Initialization/Termination
  - OM user exits
    - Client Connection, Initialization/Termination, Input, Output, Security
  - RM user exits
    - Client Connection, Initialization/Termination
  - DBRC SCI Registration Exit (DSPSCIX0)
  - IMS Connect
    - User message exits (HWSxxxx)
- Define CFRM policy parameters for the structures
  - Define OSAM, VSAM, lock, shared queues, logger, and resource structures:
    - Name, sizes (max, init, min), duplexing, autoalter, CF preference list
    - Use CFSIZER to help with sizing structure
- Create or update ARM policy
  - Define restart groups, restart methods, restart attempts
- Update the LOGR policy
  - Define shared queues log streams
- Review and update database procedures
  - Backup and recovery procedures
    - DBRing and starting databases and DEDB areas
    - Making image copies (standard, concurrent, Image Copy 2, other)
    - Running change accumulation (required to recover shared database)
    - Using IMS Online Recovery Service
  - Database monitoring and reorganization
- Review and update operational procedures
  - Starting and stopping IMSplex components
  - Monitoring and controlling the IMSplex environment
    - MTO, TSO SPOC, IMS Control Center
  - Recovery from component failure
- Review and update automated operations
  - NetView execs
  - New OM interface AO programs
  - Vendor-supplied programs
- Review and update end-user connectivity procedures
  - Connecting cold
  - Reconnecting after failure
  - Disconnecting
- Review, update, and/or create structure management procedures
  - Structure rebuild procedures
  - Structure recovery procedures
  - Structure monitoring (size and performance)

The following sections address the OS/390 and IMS subsystem environment in more detail.

OS/390 or z/OS environment

Assuming that the OS environment is already a Parallel Sysplex, there is not much else to do to prepare it for the IMSplex. Verify that the following OS/390 definitions are in place:

- OS/390 or z/OS release levels

  Verify that all operating system components are at the correct level, with required maintenance, for the functions you want to use. Call the IBM service center if you are not sure.

- Program properties table

  BPEINI00 should be included in the PPT

- Couple data sets and policies

  The IMSplex makes extensive use of XCF and XES services and Coupling Facility structures. We discussed this is some detail in Chapter 2., "Introduction to the Parallel Sysplex" on page 11.

  To take advantage of system managed processes for IMSplex structures, be sure the CFRM CDS is formatted for SMREBLD or SMDUPLEX (SMDUPLEX includes SMREBLD). In addition, the following policies must be created or updated:

  – CFRM policy - defines the structures
  – LOGR policy - defines the CQS log stream(s) and their relationship to the logger structures
  – ARM policy - defines IMSplex component restart requirements

- SYS1.PROCLIB

  All IMS started procedures must either be in SYS1.PROCLIB or a defined concatenation of this library, such as the IMS procedure library.

IMSplex environment

This section describes the IMSplex environment, including the control region, subordinate address spaces, dependent regions, and other related address spaces.

IMS system definitions

As a minimum, the following system definition macros should be reviewed, and updated if necessary, before implementing any of the IMSplex functions. A full description of the macros and parameters can be found in *IMS Version 8: Installation Volume 2: System Definition and Tailoring,* GC27-1298. Although other macros have parameters that are significant in an IMSplex, they can all be overridden at execution time by an execution parameter or in one of the IMS PROCLIB members.

- DATABASE

  Be sure the ACCESS parameter for shared databases is set to UP(date). The default is EX(clusive), which would not allow the databases to be shared

- APPLCTN and TRANSACT

  Each of these macros may be coded to allow only serial processing. Serial processing of a PSB or transaction can only be enforced within a single IMS, not across the IMSplex. PSBs and transactions coded as SERIAL should be examined to determine whether it is really a requirement. If so, then those transactions will have to be routed to a single system for serial processing. This is easy to do in a shared queues environment. Without shared queues, you must route the transactions either across an MSC link, or by having the user log on to the system where that transaction can be executed serially.

- MSC macros

If you are running with shared queues, and any IMS in the shared queues group has MSC links to a remote IMS, then every IMS must be generated with at least one each of the three MSC-related macros - MSPLINK, MSLINK, and MSNAME. This is true even if not all IMSs have physical MSC links to remote systems. Coding the macros includes the necessary MSC code to handle the MSC prefix found on incoming messages, and to queue the output messages on the remote queue.

If your IMSs are clones, then this is not an issue. Each IMS can be defined with the same physical and logical links even though only one may actually start those links.

- Network definitions

  If your IMSs are clones, then you have only one issue here, and that is the names of the master and secondary master NODEs and LTERMs. They must be unique for each IMS. However, beginning with IMS Version 6, each IMS can be generated with the same names and then made unique by coding overrides in DFSDCxxx.

If you decide that all your IMS systems will be clones, then you only need a single IMS system definition. Everything that needs to be unique can be overridden in the various PROCLIB members.

JCL and procedures

Most IMS address spaces are started either by an operator issuing the OS/390 S(tart) command for a started procedure, or by IMS itself issuing the OS/390 S(tart) command. It is possible, however, that some address spaces will be submitted as JOBs. For example, the IMS control region, DBRC, and DLISAS are typically started tasks, while dependent regions may be JOBs whose JCL is found in the IMS.JOBS library.

Each of the following address spaces, whether started procedures or JOBs, should be examined for execution in an IMSplex. Each will have to be tailored, to some extent, to the particular IMS image that they represent:

- IMS control region (CTL, DBC, DCC)

  Some data sets must be shared by all members of the IMSplex. Others must be unique to each IMS. Still others may either be unique or shared. Some may be dynamically allocated while others must be defined in the JCL. If they are dynamically allocated, they should not be specified in the procedure JCL.

  – Must be shared

    These libraries must be shared by all IMSs in the IMSplex:

    - Fast Path area data sets

      DEDB ADSs belong to the control region, not the DLI address space. These data sets are usually dynamically allocated from their DBRC definitions (INIT.ADS).

    - OLCSTAT

      This data set is used to keep track of the global online change status for each IMS with OLC=GLOBAL coded in DFSCQxxx. Not all IMSs in the IMSplex have to have the same OLC= value - some can be global while others are local. This data set is dynamically allocated using the data set name found in DFSCGxxx.

  – Must be unique

    Each IMS must have its own copy of these libraries. Data set naming conventions should identify the IMS to which the data set belongs:

    - DFSOLPnn and DFSOLSnn (OLDS)

      Every IMS must have its own set of log data sets. These can be dynamically allocated using the corresponding DFSMDA member in STEPLIB.

- DFSWADSn (WADS)

  Like the OLDS, these must be unique. These can be dynamically allocated using the corresponding DFSMDA member in STEPLIB.

- IMSRDS (restart data set)

  This data set must be unique for each IMS and defined in the JCL.

- QBLKS, LGMSG, SHMSG (message queue data sets - non-shared queues only)

  These data sets are not used in a shared queues environment. When running with local queues, they must be unique for each IMS and defined in the JCL.

- MODSTAT

  This data set is used to keep track of the online change status for an IMS that is running with local online change (OLC=LOCAL in DFSCGxxx).

- IMSMON

  The IMS monitor data set must be unique for each IMS. It can be dynamically allocated using the corresponding DFSMDA member in STEPLIB.

- MSDB data sets

  These data sets, which include the MSDB checkpoint, dump, and initialization data sets, are required only if IMS is using Fast Path main storage databases (MSDBs). Since these databases cannot be shared, these data sets must be unique.

- DFSTRA0n

  The external trace output data sets must be unique to each IMS.

– May be shared or unique

You can decide whether to share these libraries, make them unique to each IMS, or some combination of both. For example, you will probably want two dynamic allocation libraries: one for data sets that must be unique (OLDS, WADS, IMSMON) and another for data sets that must be shared (RECONs, database data sets, OLCSTAT).

- SDFSRESL

  This library (formerly, and still sometimes, called RESLIB) contains most of the IMS executable code. Each IMS system has its own nucleus module which is identified by a unique suffix. Since most of the code is loaded at initialization time, SDFSRESL is not accessed often during normal executions. It is unlikely that unique copies would be required for performance reasons. However, if one SDFSRESL is shared, all IMS systems will be at the same maintenance level and have the same defined features.

  If an installation wants to be able to run different IMS subsystems at different maintenance levels, they must have different SDFSRESLs. This is especially desirable when new maintenance levels are introduced. It might be advisable to implement the new level on one system instead of all systems simultaneously. For installations with continuous availability (24x7) requirements, this will allow them to introduce the new levels without bringing down all systems at the same time.

- Dynamic allocation libraries

  Dynamic allocation members for an IMS subsystem reside in libraries which are part of the STEPLIB concatenation. Some of the data sets defined in these members will be shared. Others must be unique. Therefore, it will probably be necessary to have two sets of dynamic allocation libraries, one common library for shared data sets, and one which is unique to each IMS for unique data sets. The STEPLIB would be similar to the following:

  ```
  //STEPLIB DD DSN=IMSPLEX1.IMSA.DYNALLOC,DISP=SHR
  //        DD DSN=IMSPLEX1.IMS.DYNALLOC,DISP=SHR
  //...
  ```

  If multiple libraries are used, then be sure to update the DFSMDA utility JCL ("//SYSLMOD DD") to put these members in the correct data set.

- Exit libraries

  Exit routines, whether written by the user or provided by IMS, must be in libraries in the //STEPLIB concatenation for the address space that invokes them. This can be the control region, the DLISAS region, or even a dependent region (for example, the data capture exit). Exit routines can reside either in their own libraries or in the IMS SDFSRESL.

Some exit routines must be common to all IMSs (for example, database randomizers and compression routines) and should share a common library. Others can be tailored for a particular IMS and must be in a library which is unique to that IMS, that is, found in the //STEPLIB for only that IMS.

The following example shows an IMS with both a common and a unique exit library. Data set names containing the IMS subsystem name are unique to that IMS subsystem:

```
//STEPLIB  DD  DSN=IMSPLEX1.IMSA.EXIT,DISP=SHR
//         DD  DSN=IMSPLEX1.EXIT,DISP=SHR
//...
```

- Online change libraries (ACBLIB, FMTLIB, MODBLKS, MATRIX)

  Whether or not these are shared will depend somewhat on whether the IMS systems are cloned, and whether the IMSs are running with OLC=GLOBAL or LOCAL. With OLC=GLOBAL, if libraries are unique, they should be identical.

  Even in a cloned environment, where the contents of all libraries are identical, many users will elect to use different data sets to guard against the possibility of a single data set error causing all IMSs to fail.

► DLISAS address space

This address space is started by the IMS control region using the DLINM execution parameter (usually in DFSPBxxx) on the OS/390 S(tart) command. The address space needs access to the IMS PROCLIB, the ACB libraries, and the database data sets. Each IMS provides its own IMSID on the internally issued START command.

Although it might be possible to use the same procedure for all copies, it is recommended that each IMS environment have its own unique DLISAS procedure.

– ACBLIB

  These JCL allocated data sets must be the same as those used by the control region. If they are concatenated, the concatenation order must be the same.

– Full function database data sets (non-HALDB)

  These data sets can be dynamically allocated using the corresponding DFSMDA member in STEPLIB.

– HALDB data sets

  These data sets are dynamically allocated using the DBRC definitions.

► DBRC address space

This address space is started by the IMS control region using the DBRCNM execution parameter (usually in DFSPBxxx) on the OS/390 S(tart) command. Each IMS provides its own IMSID on the internally issued start command. Each IMS provides its own IMSID on the internally issued start command.

– RECONs

  All DBRC address spaces in the data sharing group must use the same set of RECONs. They may be dynamically allocated using the corresponding DFSMDA members in STEPLIB.

– JCLPDS

  This is the source of skeletal JCL for the "GENJCL" command. It is defined in the DBRC procedure by the "//JCLPDS DD" statement. The skeletal JCL for GENJCL.ARCHIVE includes a STEPLIB pointing to a SDFSRESL library. However, since it is only used to find a library with the Archive utility program (DFSUARC0), it probably does not matter which SDFSRESL is identified. And, since the data sets have DSNs which include .%SSID. qualifiers, this library could be shared by all DBRCs:

```
//JCLPDS DD DSN=IMSPLEX1.JCLOUT,DISP=SHR
```

– JCLOUT

This JCL allocated data set is used to hold the output of a GENJCL command. If the DD statement specifies the internal reader, the job is submitted as soon as the JCL is generated. If it specifies a data set, then the job is placed in that data set for later submission (for example, from a TSO ISPF session). When the data set is used, it must be unique for each IMS subsystem (each DBRC instance).

```
//JCLOUT DD SYSOUT=(A,INTRDR)
- or -
//JCLOUT DD DSN=IMSPLEX1.IMSA.JCLOUT,DISP=SHR
```

– Automatic RECON loss notification (ARLN)

IMS Version 8 supports automatic RECON loss notification (ARLN) when running with the Common Service Layer. To enable this function, you must identify the IMSplex which this DBRC is to join. Note that, the first time this is done, the RECON header will be updated with the IMSplex name and can only be used within that IMSplex. ARLN can be enable by:

- Coding an IMSPLEX= parameter in the DBRC JCL
- Coding an DBRC SCI Registration Exit (DSPSCIX0) and include that exit in STEPLIB

The exit is recommended since it can be used for all DBRC instances (online, batch, and utility) without having to change any JCL.

► Dependent regions

The region types include MPP, IFP, BMP, JMP, and JBP regions. There are two planning considerations for dependent regions.

– Starting dependent regions

They are typically started as JOBs using the IMS command:

```
/START REGION <member-name>
```

where <member-name> is a member of a library defined on the //IEFRDER DD statement of the IMSRDR procedure:

```
//IEFRDER DD DSN=IMSPLEX1.JOBS,DISP=SHR
```

This library would contain JCL for each dependent region that you want to start, although a single member can contain the JCL for multiple dependent region jobs to be started with a single command. With a single IMS environment, the IMSID can be coded in the JCL. With multiple IMSs, you have choices.

- Use a different IMSRDR procedure for each IMS. The default is IMSRDR but can be changed for each IMS by coding the PRDR parameter in DFSPBxxx (or on the IMSCTF macro). Each procedure should define a different JOBS library where the dependent region JCL can be found:

```
//IEFRDER DD DSN=IMSPLEX1.IMSA.JOBS
```

- Use the same IMSRDR procedure and JOBS library and start the regions using the LOCAL keyword on the command:

```
/START REGION IMSMSG01 LOCAL
```

However, since these JOBS have job names, and job names must be unique within an OS/390 environment, to use the same JOBS library when multiple IMSs are executing on the same OS/390, you would have to change the job name when the start command is issued:

```
/START REGION IMSMSG01 JOBNAME AMSG01 LOCAL
```

Since this can be confusing, and cumbersome, it is probably easier to use different IMSRDR procedures and different JOBS libraries for each IMS subsystem. Using different JOBS libraries also allows you to tailor your dependent region configuration for each IMS environment.

– Program libraries

Another consideration is the program libraries used by the dependent regions. Since it is likely that a program is independent of the IMS on which it executes, these libraries could be shared by all dependent regions in all IMSs. However, the same performance concerns with multiple dependent regions sharing the same program libraries on a single IMS apply even more so with multiple IMS environments. But if multiple copies of program libraries are used, then updates become a factor. Therefore, it is important that, if program libraries are unique, that program updates to these libraries be carefully coordinated across the IMSplex.

PROCLIB members

There are several new PROCLIB members when running in an IMSplex and several new or changed parameters required in existing members. Each IMS must be started with an RGSUF=xxx parameter either in the IMS procedure JCL or on the S IMS command. RGSUF identifies the DFSPBxxx member where other parameters can be found, and where other PROCLIB members are identified. Because of this, there can be a single IMS PROCLIB for the IMSplex containing multiple members tailored to each IMS.

The DFSPBxxx member defines all other PROCLIB members for IMS. It is identified by the RGSUF=xxx execution parameter on the IMS control region JCL or start command. It is the only execution parameter that is required on the JCL or start command. All others can be specified or overridden in this member, or other members pointed to by this member.

Parameters identifying other members are the following:

VSPEC	DFSVSMxx
DC	DFSDCxxx
FDRMBR	DFSFDRxx
SHAREDQ	DFSSQxxx
CSLG	DFSCQxxx

Each of these PROCLIB members are either unique to an IMSplex environment, have parameters that may differ in an IMSplex, or parameters which apply only to an IMSplex environment. They are identified and described in detail in *IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations,* SG24-6929.

Exits

Special attention should be given to exits which are dependent either on the IMS on which they are running, or which may have some dependencies on the IMSplex environment. Most (but not all) of these are related to the communications environment. For example, the Signon Exit (DFSSGNX0) in an ETO environment with shared queues may need to verify that generated LTERM names are unique within the IMSplex. Some of the exits to pay particular attention to are:

- Initialization Exit (DFSINTX0)
- Logon Exit (DFSLGNX0)
- Logoff Exit (DFSLGFX0)
- Signon Exit (DFSSGNX0)

- Signoff Exit (DFSSGFX0)
- Conversational Abnormal Termination Exit (DFSCONE0)
- Output Creation Exit (DFSINSX0)
- DBRC SCI Registration Exit (DSPSCIX0)
- Build Security Environment Exit (DFSBSEX0)
- Fast Path Input Edit/Routing Exit (DBFHAGU0)
- TM and MSC Message Routing and Control User Exit (DFSMSCE0)
- Queue Space Notification Exit (DFSQSPC0)
- BPE exits (you name them)

7.3.2 Cutover to operational status

It is likely that implementation of any one of the IMSplex functions will require a cold start of IMS. Certainly, since you are moving from a single IMS environment to a multiple IMS environment, new IMSs must be cold started. Planning for the cutover to IMSplex must be carefully coordinated with operations *and with the end-users*. They must know how to disconnect from the single environment and reconnect to the IMSplex environment. For example, for VTAM users, the end-user may switch from logging on to an IMS APPLID to an IMS generic resource name (GRSNAME). For TCP/IP users coming in through IMS Connect, they must know what IP addresses to use, or what DNS names to use.

Abbreviations and acronyms

ACB	application control block
AOI	automated operator interface
AOP	automated operator program
AOR	application owning region (CICS)
APAR	authorized program analysis report
APPC	advanced program to program communication
ARLN	automatic RECON loss notification
ARM	Automatic Restart Manager
ATM	auto teller machine
BALG	balancing group
BMP	batch message program
BPE	Base Primitive Environment
BTAM	basic telecommunication access method
CCB	conversation control block
CDS	couple data set
CEC	central electronic complex
CF	Coupling Facility
CFCC	Coupling Facility Control Code
CFIA	component failure impact analysis
CFRM	Coupling Facility resource management
CI	control interval
CICS	Customer Information Control System
CNT	communication name table
CPU	central processing unit
CQS	Common Queue Server
CSA	common system area
CSL	Common Service Layer
CTC	channel-to-channel
DASD	direct access storage device
DB/DC	database/data communications
DB2	Database 2™
DBA	database administrator
DBCTL	database control
DBD	database description
DBDS	database data set
DBRC	data base recovery control
DEDB	data entry database
DL/I	Data Language/I
DLISAS	DL/I separate address space
DRA	database resource adapter
DVIPA	dynamic virtual IP addressing
EJB	Enterprise JavaBean
EMC	event monitor control
E-MCS	extended multiple consoles support
EMEA	Europe, Middle East and Africa
EMH	expedited message handler
ESCON	Enterprise System Connection
ESDS	entry sequenced data set (VSAM)
ESS	external subsystem
ETO	Extended Terminal Option
FDBR	Fast Database Recovery
FIFO	first in, first out
FTP	File Transfer Protocol
HALDB	High Availability Large Database
HDAM	hierarchic direct access method
HIDAM	hierarchic indexed direct access method
HSA	hardware system area
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
I/O	input/output
IBM	International Business Machines Corporation
IFP	IMS Fast Path program
IMS	Information Management System
IRLM	Internal Resource Lock Manager
ISC	intersystem communication
ISPF	Interactive Systems Productivity Facility
ITSO	International Technical Support Organization
JBP	Java batch processing region
JCL	job control language
JES	job entry subsystem (JES2 or JES3)
KSDS	key sequenced data set
LAN	local area network
LCV	local cache vector

LEC	list entry control
LNV	list notification vector
LOGR	System Logger
LPAR	logical partition
LTERM	logical terminal
LU	logical unit
LU2	Logical Unit 2
MCS	multiple consoles support
MNPS	multi-node persistent sessions
MPP	message processing program
MSC	multiple systems coupling
MSDB	Main Storage Data Base
MVS	Multiple Virtual System
ODBA	open database access
OLDS	online log data set
OM	Operations Manager
ORS	Online Recovery Service
OSAM	overflow sequential access method
OTMA	open transaction manager access
OTMA C/I	OTMA callable interface
PCB	program communication block
PHDAM	partitioned HDAM (HALDB)
PHIDAM	partitioned HIDAM (HALDB)
PPT	program properties table
PSB	program specification block
RACF	Resource Access Control Facility
RBA	relative byte address
RCNT	remote communication name table
RDS	restart data set
RM	Resource Manager
RMF	Resource Measurement Facility
RNR	Rapid Network Recovery
RSMB	remote scheduler message block
RSR	Remote Site Recovery
SAF	System Authorization Facility
SCI	Structured Call Interface
SFM	sysplex failure management
SLDS	system log data set
SMB	scheduler message block
SMU	security maintenance utility
SNA	Systems Network Architecture
SNPS	single node persistent sessions
SPA	scratch pad area
SPC	sysplex processing code
SPOC	single point of control
SRDS	structure recovery data set
SRM	status recovery mode
STM	sysplex terminal management
STSN	set and test sequence numbers
SVL	Silicon Valley Laboratory
TCB	task control block
TCP/IP	Transmission Control Protocol/Internet Protocol
TCT	transaction class table
TOD	time-of-day
TSO	Time Sharing Option
UOW	unit of work
UOWID	unit of work ID
VIPA	virtual IP addressing
VSAM	virtual storage access method
VSO	Virtual Storage Option (DEDB VSO)
VTAM	virtual telecommunication access method
WADS	write ahead data set
WAN	wide area network
WLM	Workload Manager
WWW	World Wide Web
XCF	cross-system coupling facility
XES	cross-system extended services
XRF	extended recovery facility

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 190:

- *IMS Primer,* SG24-5352
- *IMS/ESA V6 Parallel Sysplex Migration Planning Guide for IMS TM and DBCTL,* SG24-5461
- *IMS Version 7 High Availability Large Database Guide,* SG24-5751
- *IMS Version 7 Release Guide,* SG24-5753
- *A DBA's View of IMS Online Recovery Service,* SG24-6112
- *IMS Version 7 Performance Monitoring and Tuning Update,* SG24-6404
- *IMS e-business Connectors: A Guide to IMS Connectivity,* SG24-6514
- *IMS Version 7 Java Update,* SG24-6536
- *Ensuring IMS Data Integrity Using IMS Tools,* SG24-6533
- *IMS Installation and Maintenance Processes, SG24-6574*
- *IMS Version 8 Implementation Guide - A Technical Introduction of the New Features,* SG24-6594
- *IMS DataPropagator Implementation Guide,* SG24-6838
- *Using IMS Data Management Tools for Fast Path Databases,* SG24-6866
- *Everything You Wanted to Know about the System Logger,* SG24-6898
- *IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology,* SG24-6908
- *IMS in the Parallel Sysplex, Volume III: Operations and Implementation,* SG24-6929
- *The Complete IMS HALDB Guide, All You Need to Know to Manage HALDBs,* SG24-6945

Other resources

These publications are also relevant as further information sources:

- *IMS Version 8: Base Primitive Environment Guide and Reference*, SC27-1290
- *IMS Version 8: Command Reference,* SC27-1291
- *IMS Version 8: Release Planning Guide,* GC27-1305
- *Parallel Sysplex Overview: Introducing Data Sharing and Parallelism in a Sysplex,* SA22-7661
- *z/OS MVS Programming: Sysplex Services Guide,* SA22-7617
- *z/OS MVS Programming: Sysplex Services Reference,* SA22-7618
- *z/OS MVS Setting Up a Sysplex,* SA22-7625
- *OS/390 V1R3.0 OS/390 System Commands*, GC28-1781
- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Guid*e, SC26-4930
- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Referenc*e, SC26-4929
- *IMS Connect Guide and Reference*, SC27-0946

Referenced Web sites

These Web sites are also relevant as further information sources:

- IMS Web site:

  `http://www.ibm.com/ims`
- CF sizer tool:

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

  `http://ehone.ibm.com/public/applications/publications/cgibin/pbi.cgi`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

IMS in the Parallel Sysplex Volume II: Planning the IMSplex

(0.2"spine)
0.17"<->0.473"
90<->249 pages

IMS in the Parallel Sysplex
Volume II: Planning the IMSplex

Parallel Sysplex migration planning for IMS updated

Plan the migration to IMS data sharing and shared queues

Plan the usage of IMS Version 8 IMSplex features

This IBM Redbook is the second volume of a series of redbooks titled *IMS in the Parallel Sysplex*. These redbooks describe how IMS exploits the Parallel Sysplex functions and how to plan for, implement, and operate IMS systems working together in a Parallel Sysplex.

When migrating an IMS system from a single, non-sharing environment to one which invokes some or all of these services, or even when incorporating additional function into an existing IMSplex (for example, upgrading a data sharing system to also use shared queues), the migration process must be carefully planned. Many decisions and compromises must be made, perhaps even some application or database changes. There will also be changes to system definition and to operational procedures.

This book addresses the development of the migration plan and identifies some of the steps and considerations you might encounter when developing the plan. The result of this exercise is not to *perform* any of the implementation tasks but to *identify* those tasks which must be done and to *create a plan* for accomplishing them. The other volumes in this series are:

-*IMS in the Parallel Sysplex, Volume I: Reviewing the IMSplex Technology*, SG24-6908
-*IMS in the Parallel Sysplex, Volume III: IMSplex Implementation and Operations*, SG24-6929

SG24-6928-00 ISBN 0738453447