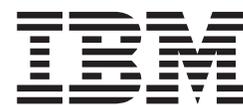


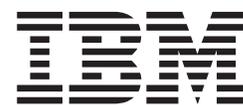
z/OS



# Parallel Sysplex Application Migration



z/OS



# Parallel Sysplex Application Migration

**Note**

Before using this information and the product it supports, be sure to read the general information under "Appendix. Notices" on page 183.

**First Edition, March 2001**

This edition applies to Version 1 Release 1 of z/OS (5694-A01), and to subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation  
Department 55JA, Mail Station P384  
2455 South Road  
Poughkeepsie, NY 12601-5400  
United States of America

FAX (United States & Canada): 1+845+432-9405

FAX (Other Countries):

Your International Access Code +1+845+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)

Internet e-mail: mhvrdfs@us.ibm.com

World Wide Web: <http://www.ibm.com/servers/eserver/zseries/zos/webqs.html>

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1994, 2001. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Figures</b> . . . . .	ix
<b>About This Publication</b> . . . . .	xi
Who Should Use This Publication . . . . .	xi
How to Use This Publication . . . . .	xii
Notes on Terminology . . . . .	xii
Notes on Product Availability . . . . .	xii
Where to Find More Information . . . . .	xiii
Related Publications in Other Libraries . . . . .	xiii
Using LookAt to look up message explanations . . . . .	xiv
Accessing licensed books on the Web . . . . .	xv

---

## Part 1. Introduction . . . . . 1

<b>Chapter 1. Sysplex Benefits for Online Transaction Processing</b> . . . . .	3
Benefits of Data Sharing in a Sysplex . . . . .	3
IMS DB Multisystem Data Sharing . . . . .	4
DB2 Multisystem Data Sharing . . . . .	4
VSAM Record Level Sharing . . . . .	5
CICS Data Sharing Facilities . . . . .	5
Benefits of Running IMS in a Sysplex Environment . . . . .	6
Running IMS TM in a Sysplex Environment. . . . .	6
Running IMS DB in a Sysplex Environment. . . . .	7
Application and Data Compatibility . . . . .	7
Benefits of Running CICS in a Sysplex Environment . . . . .	7
A CICS Complex (CICSplex) . . . . .	7
The CICS Resource Manager Regions for Sysplex Operation . . . . .	8
Application and Data Compatibility . . . . .	8
Pre-sysplex Reasons for Splitting CICS into Resource Manager Regions . . . . .	8
Extra Benefits Provided by the Sysplex . . . . .	10
CICSplex System Manager/ESA . . . . .	12
Cross-System MRO . . . . .	13
General Considerations. . . . .	13
VTAM Support for the Sysplex . . . . .	13
Security . . . . .	15
Naming Conventions. . . . .	15
<b>Chapter 2. Planning the Subsystem Configuration for a Sysplex</b> . . . . .	17
Prerequisite Hardware and Software . . . . .	17
The Target Configuration . . . . .	19
The Target Subsystems. . . . .	19
The CICS Terminal-Owning Regions . . . . .	20
The CICS Application-Owning Regions . . . . .	21
The CICS File-Owning Regions . . . . .	22
The CICS Queue-Owning Regions . . . . .	22
The IMS DBCTL Environment . . . . .	22
The DB2 Subsystems . . . . .	23
The IMS Online Environment. . . . .	23
The CICSplex SM Address Space. . . . .	23

---

## Part 2. Migrating CICS Applications . . . . . 25

<b>Chapter 3. Planning Naming Conventions for CICS and Related Subsystems</b>	27
What to Consider When Planning Naming Conventions	27
CICS	27
IMS DB	27
DB2	27
IRLM	28
CICSplex SM	28
Other Considerations	28
Designing Effective Naming Conventions	30
The Naming Convention	31
Applying the Naming Conventions for CICS	34
For CICS VTAM APPL Names (APPLIDs)	34
For SYSIDNT Names	36
For CONNECTION Names	36
For SESSIONS Definitions with and without Prefixes	39
For TERMINAL Names	42
For CICS JOB Names	43
For Data Set Names	44
Sharing CICS System Data Sets	45
CICS System Definition Data Set	45
CICS Journal Partitioned Data Set	45
SYSIN Data Set	45
Defining the Shared CICS Data Sets	45
Example of Using the Naming Convention	46
Six CPCs and Six MVS Images	46
The Terminal-Owning Regions	46
IMS DBCTL and DB2 Workloads	46
<b>Chapter 4. Planning the Terminal-Owning Regions</b>	49
Transaction Routing	49
Static and Dynamic Transaction Routing	49
Splitting CICS into Separate Terminal- and Application-Owning Regions	50
Planning Migration to Dynamic Transaction Routing	53
Transaction Affinities	53
Detecting Inter-transaction Affinities	54
Planning a Dynamic Transaction Routing Program	56
Using CICSplex SM	57
Planning for VTAM Generic Resources	58
Defining the Coupling Facility Structure	58
Defining Security Authorizations	58
Setting Trace Options	59
Defining the Generic Resource Name to CICS	59
Migrating to VTAM Generic Resources	59
Implementing VTAM Persistent LU-LU Sessions	62
Cloning the CICS Terminal-Owning Regions	62
<b>Chapter 5. Planning the Application-Owning Regions</b>	65
Achieving the Optimum Level of Processor Utilization	65
The Effect of Workload Balancing on Capacity Planning	65
CICS and Multiprocessor Capacity of an n-way CPC	65
The Ratio of AORs to TORs	67
Achieving the Required Transaction Throughput	67
Estimating the Throughput Rate	67
Achieving the Required Level of Availability	68
START Commands In a Dynamic Transaction Routing Environment	69

Start Commands that Do Not Specify TERMIID . . . . .	70
START Commands that Specify TERMIID . . . . .	70
<b>Chapter 6. Planning for VSAM Record-Level Sharing.</b> . . . . .	<b>73</b>
Concepts and use of RLS . . . . .	73
Coupling Facility Requirements for VSAM Record-level Sharing . . . . .	74
Data Set Eligibility. . . . .	74
Restrictions for Data Sets Defined with IMBED . . . . .	74
Choosing Between RLS-Mode and Non-RLS Mode . . . . .	74
Restricting Switching Between RLS Mode and Non-RLS Mode Access . . . . .	75
General Rule About Switching Opening Modes . . . . .	75
Switching Modes Exception for Read-Only Operations . . . . .	75
Resolving Retained Locks . . . . .	76
Preparing for RLS. . . . .	76
Read Integrity . . . . .	76
The LOCKED Exception Condition. . . . .	76
Defining the Coupling Facility Structures . . . . .	77
Defining the Sharing Control Data Sets . . . . .	78
Defining SMS Storage Classes . . . . .	78
Defining IGDSMSxx Parameters in SYS1.PARMLIB . . . . .	78
Defining Deadlock Time Intervals . . . . .	78
Defining recovery attributes for VSAM Data Sets . . . . .	79
Planning Migration and Coexistence . . . . .	79
Fallback Planning . . . . .	81
<b>Chapter 7. Planning for Temporary Storage Data Sharing . . . . .</b>	<b>83</b>
TS Pools and the Coupling Facility . . . . .	84
Defining Shared TS Queues . . . . .	84
The TS Data Sharing Server . . . . .	85
The Subsystem Interface . . . . .	85
Security . . . . .	86
<b>Chapter 8. Planning for Coupling Facility Data Tables . . . . .</b>	<b>87</b>
Comparison with User-maintained Data Tables . . . . .	87
Coupling Facility Data Table Models . . . . .	87
CFDT Pools and the Coupling Facility . . . . .	88
Defining a Coupling Facility Data Table . . . . .	89
The Coupling Facility Data Table Server . . . . .	89
The Subsystem Interface . . . . .	89
Security . . . . .	90
<b>Chapter 9. Planning for Named Number Counters . . . . .</b>	<b>91</b>
The Named Counter Application Programming Interfaces . . . . .	91
Named Counter Pools and the Coupling Facility. . . . .	92
Defining a Named Counter Options Table . . . . .	93
The Named Counter Server . . . . .	93
The Subsystem Interface . . . . .	93
Security . . . . .	94
<b>Chapter 10. Planning Resource-Owning Regions . . . . .</b>	<b>95</b>
Planning the File-Owning Regions . . . . .	95
Function Shipping . . . . .	95
Planning the Number of File-Owning Regions . . . . .	96
Ensuring Availability of the Data . . . . .	98
Capacity Planning Considerations . . . . .	98
Data Integrity Considerations. . . . .	99

Planning the Queue-Owning Regions . . . . .	100
Avoiding Inter-transaction Affinity Associated with CICS Queues . . . . .	101
Creating a Queue-Owning Region . . . . .	102
Data Integrity Considerations . . . . .	106
<b>Chapter 11. Planning for IMS DBCTL Multisystem Data Sharing with CICS</b>	<b>109</b>
Migrating from CICS Local DL/I to IMS DBCTL . . . . .	109
When Databases Need to be Migrated. . . . .	109
Migrating CICS Parameters to IMS DBCTL . . . . .	110
Creating Multiple DBCTL Subsystems . . . . .	111
The CICS Database Resource Adapter Startup Table . . . . .	111
Naming the DBCTL Subsystems . . . . .	111
Converting a CICS Shared-Database Program to a BMP Program . . . . .	112
Creating the Data Sharing Environment . . . . .	112
Database Recovery Control. . . . .	112
Internal Resource Lock Manager (IRLM) . . . . .	113
Defining IMS Coupling Facility Structures . . . . .	113
<b>Chapter 12. Planning for DB2 Subsystem Access from CICS Regions</b>	<b>115</b>
Creating Multiple DB2 Data Sharing Subsystems . . . . .	115
Using a Common Resource Control Table . . . . .	115
Naming the DB2 Subsystems . . . . .	115
Creating the Data Sharing Environment . . . . .	116
Defining DB2 Coupling Facility Structures. . . . .	116
Internal Resource Lock Manager (IRLM) . . . . .	116
<b>Chapter 13. Planning the Log Streams</b> . . . . .	<b>117</b>
The MVS System Logger and the CICS Log Manager . . . . .	117
Coupling Facility Requirements for the System Logger . . . . .	118
Defining the logger environment for CICS. . . . .	118
<b>Chapter 14. Planning the Resource Definitions</b> . . . . .	<b>121</b>
Defining Remote Attributes for Transaction Routing . . . . .	121
Defining Transactions for Static Transaction Routing. . . . .	121
Defining Transactions for Dynamic Transaction Routing . . . . .	123
Defining Connection and Session Definitions . . . . .	125
Links from the Terminal-Owning Regions . . . . .	125
Links from the Application-Owning Regions . . . . .	129
Links from the File-Owning and Queue-Owning Regions . . . . .	131
Cloning CICS Regions . . . . .	134
Cloning Regions of the Same Type . . . . .	134
Defining Common System Initialization Parameters . . . . .	134
Defining the Unique System Initialization Parameters . . . . .	135
Examples of SYSIN Members for Cloning Application-Owning Regions	135
<b>Chapter 15. Planning CICSplex Security</b> . . . . .	<b>137</b>
Defining Bind-Time and Link Security . . . . .	137
Defining Bind-Time Security. . . . .	137
Defining Link Security . . . . .	137
Authenticating Users in Remote MRO Regions. . . . .	138
Authenticating Users Associated with MRO Requests . . . . .	138
Authenticating Users Signing On Directly to Remote Regions . . . . .	138
Guaranteeing Equal Access to Cloned Application-Owning Regions . . . . .	139
Defining CICS Region Userids for Started Jobs . . . . .	139
MRO Link Security Considerations . . . . .	140
Bypassing Link Security Checking . . . . .	141

Authorizing the Link Userid . . . . .	141
Security in the Receiving Regions . . . . .	141
<b>Chapter 16. Planning for Workload Management . . . . .</b>	<b>143</b>
Using CICSplex SM for Workload Balancing and Workload Separation . . . . .	143
Workload Separation . . . . .	143
Workload Management . . . . .	143
Implementing Shortest-Queue Workload Balancing . . . . .	144
Providing a Dynamic Transaction Routing Program . . . . .	145
The MVS Workload Manager . . . . .	146
Preparing to Migrate to Goal-Mode Workload Management . . . . .	146
<b>Chapter 17. Planning the CICS Startup Procedures . . . . .</b>	<b>149</b>
The CICS System Initialization Parameters . . . . .	149
Using the Default System Initialization Table . . . . .	149
Defining Common System Initialization Parameters for Cloned Regions . . . . .	150
Using SYSIN for Common System Initialization Parameters . . . . .	156
Using the PARM Parameter for Unique System Initialization Parameters . . . . .	157
The CICS System Data Sets . . . . .	158
Defining the CICS Startup Procedure for Started Jobs . . . . .	158
Starting the CICS Regions . . . . .	160

---

## **Part 3. Migrating IMS Applications . . . . . 163**

<b>Chapter 18. Planning for IMS TM in a Sysplex Environment . . . . .</b>	<b>165</b>
Cloning Your IMS Subsystems. . . . .	165
What to Share between IMS Subsystems in a Parallel Sysplex. . . . .	165
Ensure Unique IMSIDs . . . . .	166
Ensure Unique Terminal Names, LU Names, and User IDs . . . . .	168
Divide Your Network . . . . .	169
Advantages of MSC . . . . .	169
Planning for MSC . . . . .	169
Workload Balancing Using MSC . . . . .	170
Flow of Data within Multiple Systems . . . . .	171
Convert Batch Jobs to BMP Programs. . . . .	172
MVS Resource Management . . . . .	173
Availability and Recovery. . . . .	174
<b>Chapter 19. Planning for IMS/ESA Version 6 in a Parallel Sysplex Environment . . . . .</b>	<b>175</b>
Parallel Sysplex Migration Requirements . . . . .	175
Migrating IMS . . . . .	175
Migrating Your Data-Sharing Environment . . . . .	175
Planning for Migration to IMS Version 6 . . . . .	176
Planning for a Shared-Queues Environment. . . . .	176
Migrating to a Shared-Queues and Shared-EMH Environment . . . . .	176
Benefits of Using Shared Queues . . . . .	176
Required Components of a Shared-Queues Environment . . . . .	177
Planning for the Common Queue Server (CQS) . . . . .	178
Planning for Using VTAM Generic Resource Groups . . . . .	178
Requirements for Using VTAM Generic Resource Groups . . . . .	179
Restrictions on Using VTAM Generic Resource Groups . . . . .	180
Planning for OSAM Database Cache Migration . . . . .	180
Planning for Shared SDEPs Migration . . . . .	180
Planning for Shared VSO DEDB Areas Migration . . . . .	180

---

<b>Part 4. Appendixes</b> . . . . .	181
<b>Appendix. Notices</b> . . . . .	183
Trademarks. . . . .	184
<b>Glossary</b> . . . . .	187
Sources of Terms and Definitions. . . . .	187
Explanation of Cross-References. . . . .	187
<b>Index</b> . . . . .	193

---

## Figures

1.	A Conceptual View of Data Sharing Subsystems in a Sysplex . . . . .	18
2.	Example of Connection Naming Between Terminal-Owning Region and Application-Owning Region . . . . .	37
3.	Example of generic connection naming. . . . .	38
4.	Allocation of SEND/RECEIVE Prefixes for Down and Up Links . . . . .	41
5.	Sample MRO Configuration . . . . .	42
6.	Allocation of SEND/RECEIVE Prefixes for Lateral Links . . . . .	42
7.	Unique CICS Data Sets Required for Each Region . . . . .	44
8.	Shared CICS Data Sets . . . . .	45
9.	The CICS Transaction Routing Facility . . . . .	49
10.	CICS Region Configurations Before and After Splitting into Separate Resource Manager Regions for Terminals and Applications . . . . .	51
11.	Alternative Initial Configuration Using Multiple Terminal-Owning Regions . . . . .	52
12.	The CICS Affinity Utility Components . . . . .	56
13.	CICS Region Configurations Before and After Splitting into Separate Terminal-Owning and Application-Owning Regions. . . . .	63
14.	A Partial View of the CICSplex Configuration Showing some of the Application-Owning Regions . . . . .	66
15.	Plan Showing Which Applications the Regions Can Process . . . . .	69
16.	Conceptual View of Parallel Sysplex with an SMSVSAM Server in Each MVS Image . . . . .	73
17.	Migration Scenario Using a Mixture of Function Shipping and RLS . . . . .	81
18.	Conceptual View of Parallel Sysplex with a TS Server in Each MVS Image . . . . .	84
19.	Conceptual View of Parallel Sysplex with a CFDT Server in Each MVS Image . . . . .	88
20.	Conceptual View of Parallel Sysplex with a Named Counter Server in Each MVS Image . . . . .	92
21.	Function Shipping . . . . .	96
22.	Function Shipping—Update . . . . .	97
23.	SYNCPPOINT Processing in an MRO Environment . . . . .	99
24.	Example of Inter-transaction Affinity Caused by Using Local Temporary Storage . . . . .	101
25.	Using Remote Queues to Avoid Inter-transaction Affinity Relating to Temporary Storage . . . . .	103
26.	Using Remote Queues to Avoid Inter-transaction Affinity Relating to Transient Data . . . . .	105
27.	Local DL/I Access Direct from an Application-Owning Region . . . . .	110
28.	DL/I Access Through Function Shipping to a File-Owning Region . . . . .	110
29.	CICS access to IMS databases through the IMS DBCTL interface . . . . .	110
30.	A Remote Transaction Resource Definition that Specifies Static Routing . . . . .	123
31.	A Remote Transaction Resource Definition that Specifies Dynamic Routing . . . . .	125
32.	Required Links to the Application-Owning Regions from the Terminal-Owning Regions. . . . .	126
33.	Resource Definitions for Links from Terminal-Owning Regions to Application-Owning Regions . . . . .	127
34.	Required Links, 6 in all, from Application-Owning Regions to the Terminal-Owning Regions and Resource-Owning Regions. . . . .	129
35.	Resource Definitions for Links from the Application-Owning Regions to Terminal-Owning Regions and Resource-Owning Regions . . . . .	130
36.	Required Links, 12 in All, from File-Owning and Queue-Owning Regions to All Application-Owning Regions . . . . .	131
37.	Resource Definitions for Links from File-Owning and Queue-Owning Regions . . . . .	132
38.	Example of Common System Initialization Parameters for a Terminal-Owning Region . . . . .	151
39.	Example of Common System Initialization Parameters for an Application-Owning Region . . . . .	153
40.	Example of Common System Initialization Parameters for a File-Owning Region . . . . .	155
41.	Example of Common System Initialization Parameters for a Queue-Owning Region . . . . .	156
42.	Defining the CICS System Initialization Parameters in the PARM Parameter . . . . .	157
43.	A Sample CICS Startup Procedure for All CICS Regions in a CICSplex . . . . .	159
44.	Procedure to Start 5 CICS Tasks, 1 for Each CICS Region . . . . .	161
45.	Multiple IMS Systems Transaction Flow . . . . .	172
46.	Components of a Shared-Queues Environment . . . . .	178



---

## About This Publication

This revised edition of the publication is about migrating user applications from a non-sysplex MVS environment to a sysplex that uses the coupling facility (a Parallel Sysplex). It is aimed primarily at users of online transaction processing systems such as CICS and IMS. It also discusses the implications for migrating IMS batch applications.

The applications described in this book are predominantly online applications that use CICS as the transaction manager, and a database manager such as IMS (DBCTL), or DB2. It also covers the use by CICS transactions of VSAM-managed files, and CICS temporary storage data, and how access to such data can be optimized in a sysplex configuration. This book also describes application migration for the IMS Transaction Manager.

In general, this book aims to emphasize what you can do to aid migration *before* establishing a Parallel Sysplex environment, such as preparing to use CICS dynamic transaction routing and migrating to IMS Database Control (DBCTL).

### Sysplex and Product Availability (“Roll-Out”)

The sysplex is a large system computing environment that is evolving. Since the introduction of the sysplex, the coupling facility technology was developed to enhance sysplex capabilities. With a coupling facility in a sysplex, the participating MVS systems can do high performance data sharing. A sysplex with a coupling facility is called a *Parallel Sysplex*.

**Note that this publication generally does not differentiate between a sysplex without a coupling facility and a sysplex with a coupling facility. When you see the term *sysplex*, understand it to mean a *Parallel Sysplex*.**

There might be changes to the implementation or availability of new products or functions. For information about availability of hardware and software sysplex support, consult your IBM marketing representative.

See “Notes on Product Availability” on page xii for additional information about phased product availability.

---

## Who Should Use This Publication

This book is for CICS or IMS system administrators who are responsible for managing CICS (either in a single-region or multiregion environment) or IMS, and who want to know about the benefits of a sysplex and how to plan for its implementation. It is also for system and application programmers who need to understand the effect of a sysplex on application development and design.

The book assumes that you are an existing CICS, IMS, or DB2 user:

- For CICS users, this book assumes you use CICS with the IMS/ESA Database Manager (IMS DB) or DB2 and are familiar with CICS/ESA, DB2, and IMS DB applications.

- For IMS users, this book assumes you use IMS TM with IMS DB or DB2 and are familiar with IMS and DB2 database and data communications applications.

---

## How to Use This Publication

For the CICS information, this book is intended to be read sequentially, and in conjunction with other books in related libraries (MVS, CICS/ESA, IMS, and so on). When the other libraries contain more detailed information, there are references to the relevant books.

For the IMS information, you should read relevant portions of “Part 1. Introduction” on page 1 and all of “Part 3. Migrating IMS Applications” on page 163. This book is intended to be read in conjunction with other books in related libraries (MVS, CICS/ESA, IMS, and so on). When the other libraries contain more detailed information, there are references to the relevant books.

See also “Related Publications in Other Libraries” on page xiii for details of books from related libraries.

## Notes on Terminology

Note the following regarding terminology in this publication:

- When you see the term **sysplex**, understand it to mean a Parallel Sysplex.
- When you see the term **MVS** in this book, understand it to mean the element of either OS/390 or z/OS.
- The term **system** is one that is used heavily in many publications to mean many different things. Often the reader must attempt to discern its exact meaning from the context in which it is used. This publication uses the term **system** to mean an **MVS system**. Here are some important definitions:
  - MVS system**. An MVS image together with its associated hardware, which collectively are often referred to simply as a system, or MVS system.
  - MVS image**. A single occurrence of the MVS operating system that has the ability to process work.
- In this publication, the licensed program DB2 for OS/390 is referred to as **DB2**.
- In this publication, the element of DFSMS of OS/390 or z/OS is referred to as **DFSMS**.
- The Customer Information Control System is referred to as **CICS**, an element of the CICS Transaction Server for OS/390 licensed program. CICS Transaction Server also contains CICSplex System Manager (CICSplex SM), the CICS Clients for workstation platforms, and other elements,
- The Information Management System Database Manager is referred to as **IMS DB**, and IMS Transaction Manager is referred to as **IMS TM**.

Some terms and concepts that may be new to you are introduced in this book. Where necessary, these are explained either in the text where they are first used, or in footnotes. There is also a small glossary at the back of the book.

## Notes on Product Availability

The term “follow-on phase” is used when discussing function planned to be available in future phases of the roll-out of sysplex support across products. If “follow-on phase” is not specified, assume that the function is available or announced. Note that the information about follow-on phases represents IBM’s intent, and is subject to change or withdrawal.

---

## Where to Find More Information

This publication is part of a library of sysplex planning publications. IBM recommends that you read these publications before you begin planning your sysplex. Here is the complete list of publications in the sysplex planning library, along with their order numbers:

Table 1. Sysplex Library Books

Publication Title	Order Number	Description
<i>z/OS Parallel Sysplex Overview</i>	SA22-7661	Use this publication as an overview to the sysplex and coupling facility data sharing. It describes highlights and the value of the sysplex to your business.
<i>z/OS Parallel Sysplex Application Migration</i>	SA22-7662	Use this publication to understand planning considerations for the parallel processing of online applications that run on Customer Information Control System (CICS)/ESA and Information Management System (IMS)/ESA software in the sysplex.

For OS/390 or z/OS, see *z/OS MVS Setting Up a Sysplex* (SA22-7625) for information about installing and managing MVS systems in a sysplex.

## Related Publications in Other Libraries

A number of publications in other product libraries are referenced in this publication.

### CICS

*CICS Transaction Server for OS/390 Installation Guide*, GC34-5697  
*CICS System Definition Guide*, SC34-5725  
*CICS Application Programming Reference*, SC34-5703  
*CICS Application Programming Guide*, SC34-5702  
*CICS Resource Definition Guide*, SC34-5722  
*CICS Shared Data Tables Guide*, SC34-5723  
*CICS Transaction Server for OS/390 Migration Guide*, GC34-5699  
*CICS Operations and Utilities Guide*, SC34-5717  
*CICS Intercommunication Guide*, SC34-5712  
*CICS Recovery and Restart Guide*, SC34-5721  
*CICS Performance Guide*, SC34-5718  
*CICS IMS Database Control Guide*, SC34-5711  
*CICS RACF Security Guide*, SC34-5720  
*CICS/ESA Dynamic Transaction Routing in a CICSplex*, SC33-1012  
*CICS Customization Guide*, SC34-5706

### IMS

*IMS/ESA Administration Guide: System*  
*IMS/ESA Installation Volume 1: Installation and Verification*  
*IMS/ESA Installation Volume 2: System Definition and Tailoring*  
*IMS/ESA Operations Guide*  
*IMS/ESA Utilities Reference: System*, SC26-8770

### DB2

*DB2 Data Sharing: Planning and Administration*  
*DB2 Installation Guide*

## VTAM

*VTAM Network Implementation Guide* , SC31-6494  
*VTAM Operation* , SC31-6495

## OS/390 or z/OS

The OS/390 or z/OS product contains MVS, DFSMS, and VTAM as elements. The titles and order numbers for the MVS, DFSMS, and VTAM elements referenced in this publication are for z/OS and include the following:

*z/OS DFSMSdfp Storage Administration Reference*, SC26-7402  
*z/OS DFSMS Access Method Services*, SC26-7394  
*z/OS MVS Planning: Workload Management*, GC28-1761.  
*z/OS MVS Setting Up a Sysplex*, GC28-1779  
*z/OS Communications Server: SNA Network Implementation Guide*, SC31-6548  
*z/OS Communications Server: SNA Operation*, SC31-6549

## Processor Resource/Systems Manager (PR/SM)

*PR/SM Planning Guide* , GA22-7123  
*PR/SM Planning Guide (S/390 processors only)*, GA22-7236

For information about other products discussed in this publication, see the individual product libraries.

---

## Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages and system abends.

Using LookAt to find information is faster than a conventional search because LookAt goes directly to the explanation.

LookAt can be accessed from the Internet or from a TSO command line.

You can use LookAt on the Internet at:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>

To use LookAt as a TSO command, LookAt must be installed on your host system. You can obtain the LookAt code for TSO from the LookAt Web site by clicking on **News and Help** or from the *z/OS Collection*, SK3T-4269.

To find a message explanation from a TSO command line, simply enter: **lookat** *message-id* as in the following example:

```
lookat iec192i
```

This results in direct access to the message explanation for message IEC192I.

To find a message explanation from the LookAt Web site, simply enter the message ID. You can select the release if needed.

**Note:** Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS MVS Routing and Descriptor Codes*. For such messages, LookAt prompts you to choose which book to open.

---

## Accessing licensed books on the Web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link Web site at:

<http://www.ibm.com/servers/resourceLink>

Licensed books are available only to customers with a z/OS license. Access to these books requires an IBM Resource Link Web userid and password, and a key code. With your z/OS order you received a memo that includes this key code.

To obtain your IBM Resource Link Web userid and password log on to:

<http://www.ibm.com/servers/resourceLink>

To register for access to the z/OS licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **User Profiles** located on the left-hand navigation bar.
3. Click on **Access Profile**.
4. Click on **Request Access to Licensed books**.
5. Supply your key code where requested and click on the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed. After your request is processed you will receive an e-mail confirmation.

**Note:** You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

To access the licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **Library**.
3. Click on **zSeries**.
4. Click on **Software**.
5. Click on **z/OS**.
6. Access the licensed book by selecting the appropriate element.



---

## Part 1. Introduction

This provides a general introduction to the migration of CICS and IMS online applications to a Sysplex environment.

## Introduction

---

## Chapter 1. Sysplex Benefits for Online Transaction Processing

This chapter explains why a sysplex provides an ideal environment for online transaction processing. In particular, it discusses why the sysplex with data sharing through use of the coupling facility is so well suited to a transaction processing subsystem such as CICS or IMS/ESA Transaction Manager.

The transaction workload of a transaction processing system is typically composed of a large number of independent, relatively simple, processing requests from a large set of requestors (often a terminal network). Theoretically there is a high degree of parallelism in the workload that can be executed on a highly parallel architecture relatively easily.

**Challenges to Running Transactions in Parallel:** There are complicating factors that make running transactions in parallel less straightforward, such as:

- The parallel systems need to look, from the outside, like one system.
- The transactions often are trying to update or access the same data.
- The pattern of reference to the data often varies radically over time.
- Today's systems need to be available for work without interruption, even when some components fail.
- The parallel systems should be able to execute existing programs without modification.

**How the Sysplex Meets the Challenge:** The features of the sysplex, exploited by IBM subsystems and products, are designed to produce a system that meets all of these challenges. The sysplex enables parallel transaction processing, which provides:

- High availability
- High capacity
- Workload balancing
- Application compatibility
- Ease of use
- The capability for nondisruptive growth as your needs grow.

**Management of Data and Transactions:** Online transaction processing involves the management and sharing of data and the management of transactions that use that data. In this overview chapter, you will read about the following:

- "Benefits of Data Sharing in a Sysplex"
- "Benefits of Running IMS in a Sysplex Environment" on page 6
- "Benefits of Running CICS in a Sysplex Environment" on page 7
- "General Considerations" on page 13

---

### Benefits of Data Sharing in a Sysplex

The sysplex data sharing capabilities are exploited by a number of IBM products:

#### **IMS DB**

DL/I data sharing is available to both IMS and CICS online applications, and also IMS batch applications.

#### **DATABASE 2**

DB2 data sharing is available to both IMS and CICS online applications, and also DB2 batch and TSO/E applications.

## DFSMS

VSAM record-level data sharing is available to CICS online applications.

## CICS

Temporary storage data sharing, coupling facility data tables, and named number counter sharing are available to CICS online applications.

## IMS DB Multisystem Data Sharing

The sysplex coupling facility provides support for high performance multisystem data sharing, which is exploited by the database management component of IMS/ESA. This is used by CICS application-owning regions that need to access databases managed by IMS Database Control (IMS DBCTL) and by IMS TM dependent regions that need access to databases managed by IMS DB.

IMS data sharing ensures that all CICS-DL/I or IMS TM transactions have equal access to the databases. You are no longer restricted by the number of MVS images running IMS; therefore, the total processing capacity that can be applied to access a given database is increased considerably. Note that IMS multisystem data sharing is only available with IMS DB or DBCTL—it is not available for CICS local-DL/I applications.

For more information migrating to a multisystem data sharing environment, see “Chapter 11. Planning for IMS DBCTL Multisystem Data Sharing with CICS” on page 109

## DB2 Multisystem Data Sharing

DB2 Version 4 uses the sysplex to provide multisystem data sharing, which can provide many benefits, including more processing capacity, higher availability, and more ways to configure systems to meet the needs of various user groups. And DB2 can take advantage of these benefits while continuing to serve as your enterprise data server.

More information about DB2 data sharing is in *DB2 for MVS/ESA Version 4 Data Sharing: Planning and Administration*. See “Chapter 12. Planning for DB2 Subsystem Access from CICS Regions” on page 115 for more information about migrating CICS applications to a DB2 multisystem data sharing environment.

**Increased Capacity:** As more mission-critical transactions and queries use DB2, it becomes increasingly important to extend the processing capacity of your system. To avoid the cost of managing data copies, and to avoid having to rewrite applications to access distributed data, you can add DB2s to a data sharing group, and thereby have concurrent read-write access to the same set of data. There is a single DB2 catalog and directory used by all members of the group, providing applications with a single-system image. Because the existing SQL interfaces remain unchanged, your investment in people and skills is protected.

You can continue to add DB2s to the group without affecting the work currently in progress. Data is not “bound” to a particular DB2, so you can balance the workload without having to repartition data.

**Higher Availability:** Having multiple DB2s capable of handling the transaction workload provides an opportunity for greater availability, especially when you perform maintenance, by allowing users on one system to be moved to another system. If you use CICSplex SM, unplanned outages can be less disruptive because transactions can automatically be routed away from the failed system. DB2

can also take advantage of MVS automatic restart management to get the DB2 subsystem up again as quickly as possible.

**Configuration Flexibility:** Each DB2 subsystem in the group can be configured to meet the needs of different types of user groups. For example, you might configure some DB2s for high-priority transaction processing and others for TSO/E or batch users. And all users can still have access to the same set of data.

**The DB2 Server:** The DB2 data sharing group is a single large server capable of handling many thousands of client connections. There are options for how these clients can connect to the data sharing group and how to balance requests. Applications do not require changes to begin accessing a data sharing group. See “How DB2 Uses VTAM Generic Resources” on page 14 for more information.

## VSAM Record Level Sharing

DFSMS 1.3 uses the sysplex to provide multisystem VSAM record-level sharing (RLS), removing the need for CICS file-owning regions (FORs). DFSMS supports a new data sharing subsystem (SMSVSAM) that provides CICS applications running in multiple CICS regions equal access to VSAM data.

Data integrity is provided by global locking at the record level, not by control interval as defined by the SHAREOPTIONS parameter (which is ignored for data sets accessed in RLS mode).

**High Availability:** With VSAM record-level sharing, multiple CICS application-owning regions (AORs) running in one or more MVS images within the sysplex all have access to the same data. The sysplex supports more than one SMSVSAM subsystem, ensuring that there is no single point of failure as in the case of a CICS file-owning region. If there is loss of access through one subsystem, the CICS dynamic transaction routing facility can route work to those AORs that continue to have access to data through another SMSVSAM.

See “Chapter 6. Planning for VSAM Record-Level Sharing” on page 73 for more information.

## CICS Data Sharing Facilities

CICS provides three data sharing facilities for use by application programs:

- Temporary storage data sharing
- Coupling facility data tables
- Named number counters.

### Temporary Storage Data Sharing

CICS temporary storage (TS) data sharing provides multiregion access to non-recoverable temporary storage queues. This facility allows CICS applications to have equal access to non-recoverable TS queues from multiple CICS regions running on any MVS image within the sysplex.

Enabling TS queues to be accessed by any CICS region within the sysplex:

- Improves workload management for CICS transactions
- Simplifies the migration of existing applications to the sysplex environment.

See “Chapter 7. Planning for Temporary Storage Data Sharing” on page 83 for more information.

## **Coupling Facility Data Tables**

CICS coupling facility data tables provides multiregion access to file data sharing without the need for a file-owning region. The data is held in tables, equivalent to user-maintained data tables, in a coupling facility list structure. This facility allows user application programs to have equal access to working data, with full update capability, through the CICS file control API.

Enabling data tables to be stored in the coupling facility and accessed by any CICS region within the sysplex:

- Improves workload management for CICS transactions
- Provides a sysplex-wide shared data tables support, with update integrity.

See “Chapter 8. Planning for Coupling Facility Data Tables” on page 87 for more information.

## **Named Number Counters**

This facility maintains a sequence of numbers as a named counter in the coupling facility, and allows CICS applications and batch program to have access to a named counter through a named counter server. The named counter APIs enable multiple CICS regions running on any MVS image within the sysplex, and batch programs, to obtain sequence numbers that are unique throughout the sysplex

See “Chapter 9. Planning for Named Number Counters” on page 91 for more information.

---

## **Benefits of Running IMS in a Sysplex Environment**

IMS has provided a database and transaction processing environment for more than a quarter of a century. Over the years, IMS DB and IMS TM have been extensively enhanced to meet the needs of customer business applications, which have continued to grow and place increasing demands upon IMS services. To meet these demands, and to satisfy customer needs, IMS is continually evolving to exploit the capabilities of modern computer processors and communication systems.

IMS/ESA Version 6 continues in this tradition. Some of the latest steps in the IMS evolution are designed to exploit the Sysplex environment.

See “Part 3. Migrating IMS Applications” on page 163.

## **Running IMS TM in a Sysplex Environment**

IMS TM has always run in a multiprocessor environment. It is perfectly suited to the Sysplex environment because most customers will not have to make any changes in the way they administer or operate IMS.

However, there are some things you should do to migrate your IMS TM systems more easily from a non-Sysplex environment to a Sysplex environment. See “Chapter 18. Planning for IMS TM in a Sysplex Environment” on page 165. With IMS Version 6, you can share message queues across the Sysplex. Therefore, the total processing capacity that can be applied to access a given transaction is increased considerably. IMS TM ensures that all programs have equal access to the shared message queues. See “Chapter 19. Planning for IMS/ESA Version 6 in a Parallel Sysplex Environment” on page 175.

## Running IMS DB in a Sysplex Environment

IMS DB uses the Sysplex coupling facility to support high-performance, multisystem data sharing in the IMS DB/DC environment.

IMS DB has had data sharing for some time, but was limited to two MVS images. Now, by using the coupling facility, IMS DB no longer has this limitation. Therefore, the total processing capacity that can be applied to access a given database is increased considerably. IMS DB ensures that all transactions have equal access to the shared databases.

## Application and Data Compatibility

There is no need to change the structure of your IMS applications, or application programs, or data, to migrate to a data sharing Sysplex environment.

---

## Benefits of Running CICS in a Sysplex Environment

CICS has been providing a transaction processing environment for more than a quarter of a century. Throughout this period, IBM's customers have been designing and developing their own CICS online business applications. At the same time, a huge industry has developed around CICS, with hundreds of independent software vendors producing CICS applications packages, for a wide range of industries and organizations.

Over the years, CICS has been extensively enhanced to meet the needs of these business applications, which continue to grow and place increasing demands upon CICS services. To meet these demands, and to satisfy customer needs, CICS has been continually evolving to exploit the capabilities of modern computer processors and communication systems.

CICS continues in this tradition—many of the latest steps in its evolution are designed to exploit a sysplex environment.

## A CICS Complex (CICSplex)

Although, originally, a full-function CICS ran in a single address space (region) within the MVS environment, nowadays most CICS users are more accustomed to running multiple, interconnected, CICS regions. Using the CICS multiregion operation (MRO) intercommunication facility, users can combine CICS regions into a complex of subsystems.

Using multiregion operation, you can separate CICS functions into individual regions, the different types of CICS regions being classified as resource managers. With the latest enhancements to MRO, these CICS resource manager regions can reside in one or more MVS images (see “Cross-System MRO” on page 13 for details).

A set of interconnected CICS regions acting as resource managers, and combining to provide a set of coherent services for a customer's business needs, is often referred to as a CICSplex. In its simplest form, a CICSplex operates within a single MVS image, and uses CICS multiregion operation facilities. Within a sysplex, a CICSplex can be configured across all the MVS images in the sysplex (see Figure 1 on page 18 for an illustration of a CICSplex spread across an MVS sysplex.)

## The CICS Resource Manager Regions for Sysplex Operation

Most CICS customers are familiar with the CICS resource manager regions known as terminal-owning regions (TORs), application-owning regions (AORs), and file-owning regions (FORs). However, to migrate CICS applications to a sysplex successfully, you may need to extend and change the use of this resource manager principle even further in your installation.

For example, combined regions such as a terminal- and application-owning region, or an application- and file-owning region, are not recommended in a sysplex, and you should plan to split these functions into separate regions. Creating unique terminal and application regions is discussed in “Chapter 4. Planning the Terminal-Owning Regions” on page 49 and “Chapter 5. Planning the Application-Owning Regions” on page 65.

You should also investigate how your applications use the CICS temporary storage and transient data facilities. If, after investigation, it's necessary, create queue-owning regions for the management of CICS temporary storage and transient data queues. The reasons are discussed in “Planning the Queue-Owning Regions” on page 100.

In addition to the CICS-based resource managers, CICS also supports access to other (non-CICS) data managers through its resource manager interface (RMI). The two IBM-supplied resource managers that use the CICS RMI are DATABASE 2\* (DB2) and IMS Database Control (DBCTL) (DBCTL being the strategic interface to IMS databases in preference to the CICS local DL/I interface). If your applications require access to DB2 or IMS databases, you must plan how to access these databases in a sysplex.

All the traditional benefits of splitting CICS into several resource manager regions and linking them into an MRO-based CICSplex are valid in a sysplex. With the introduction of the coupling facility data sharing sysplex, there are additional benefits from operating an MRO CICSplex across multiple MVS images in the sysplex.

## Application and Data Compatibility

There is no requirement to change the structure of your applications, or the application programs, or data, to migrate to a data sharing sysplex environment.

The MRO configurations discussed in this book assume that you can migrate your CICS applications and their data, without change, to a sysplex environment.

You may want to change some of the names you use in the various system and resource definitions, for the reasons discussed in “Chapter 3. Planning Naming Conventions for CICS and Related Subsystems” on page 27, but such changes are optional, not mandatory. Other similar recommendations are discussed in the chapters that discuss the different types of CICS region.

## Pre-sysplex Reasons for Splitting CICS into Resource Manager Regions

There are several reasons for separating CICS functions into a number of resource manager regions, and linking them into an MRO-based CICSplex. These are:

- Multiprocessor performance
- Workload separation
- Virtual storage constraint relief

- Availability
- Faster restart
- Managing resources between shifts.

### **Multiprocessor Performance**

In general, CICS runs user application programs under a single task control block (TCB). Running multiple application-owning regions, driven from one or more terminal-owning regions, enables the multiprocessor capacity of n-way central processor complexes (CPCs) to be more fully utilized than by channelling all CICS work through one region. Also, creating a number of application-owning regions that are all capable of running the same workload enables the terminal-owning regions to perform workload management across the CICSplex.

### **Workload Separation**

Assigning applications to separate application-owning regions might be desirable for a number of reasons:

#### **Departmental Separation**

Some user departments might require their own CICS region for accounting requirements, for security purposes, or other reasons.

#### **Time-sharing**

Applications that are heavy users of central processor resource (processor-bound) should be run in a separate region, to avoid any impact on the response times for other applications.

### **Virtual Storage Constraint Relief**

Although nowadays most CICS code and control blocks reside above 16MB, a CICS region is still limited by the amount of virtual storage available below 16MB boundary for user transactions. It is possible to relieve the storage constraint by splitting a CICS region into two or more separate regions, each of which has access to shared resources as necessary.

### **Availability**

Creating multiple application-owning regions can increase CICS availability by reducing or minimizing the impact of a CICS failure. If one region fails, users whose transactions are running in another region are unaffected. Availability is further increased if more than one region is capable of running the same transactions. See "Chapter 5. Planning the Application-Owning Regions" on page 65 for more information about replicating (cloning) CICS regions.

### **Faster Restart**

Restart after a CICS region failure is generally faster because there is less recovery to do for a single-resource region than there is for a stand-alone CICS region. For example, if an application-owning region fails, and the files and databases are owned by other CICS regions, the resource owning regions perform any needed recovery operations associated with transactions that are in-flight at the time of the failure. They do this independently as soon as the failure occurs, without waiting for the application-owning region to restart. This means the application-owning region does not have to do any file or database back-out recovery operations (which would otherwise have to be performed serially) during an emergency restart.

Furthermore, the application-owning region does not have to perform any terminal control recovery operations for logged-on terminals, because this is a function of the terminal-owning region. It also means that the only users affected by the failure are those whose transactions were running in the application-owning region at the time of the failure. Although these users experience the effects of their transactions abending, they remain connected to CICS.

## Managing Resources Between Shifts

The division of operating hours into different shifts (prime shift, evening shift, and so on) may also provide a reason for splitting CICS into separate regions. Although some regions, including a terminal-owning region, may need to span many shifts, some application-owning regions may be needed only for one shift (the prime shift, say) after which they can be shut down and their released resources made available for other tasks, such as an evening's batch work.

## Extra Benefits Provided by the Sysplex

Installing an MRO-CICSplex across an MVS sysplex, supported by CICS dynamic transaction routing, can give you:

- Improved performance
- Improved availability
- Simplified systems management.

If you are unfamiliar with CICS dynamic transaction routing, see the *CICS Intercommunication Guide*.

### Improved Performance

Improvements in performance can be achieved in the form of increased throughput and improved response times.

#### Increased throughput

An increase in throughput can be achieved by cloning application-owning regions, and spreading the workload across the cloned application-owning regions, which themselves can be spread across the sysplex.

Application-owning region clones are application-owning regions that are identical in every respect except for identifiers such as SYSIDNTs and APPLIDs, and therefore each application-owning region is capable of running the same transaction workload. Using workload balancing techniques in the dynamic transaction routing program to optimize the use of the available application-owning regions, can give you:

- An increase in capacity for single applications, by exploiting multiple CPCs within the sysplex
- An improvement in central processor (CP) utilization.

You cannot achieve the same improvements for a single application using static transaction routing, because the application-owning region for any given set of transaction ids is always fixed. Using dynamic transaction routing, however, you can have multiple application-owning regions that are capable of processing the transactions for an application, and you can balance the incoming workload according to some load-balancing algorithm. For example, the dynamic transaction routing program could use the "shortest-queue" technique to select the application-owning region with the lowest number (shortest queue) of current transactions.

#### Improved response times

An improvement in response times can be achieved by giving priority to some transactions in preference to others, and sending high-priority tasks to the application-owning region with the shortest queue. This form of workload balancing can:

- Improve response times where response time is critical, possibly at the expense of throughput
- Achieve more consistent response times

- Help you to manage the workload within the targets set by service level agreements.

## **Improved Availability**

Improved availability can be achieved in the form of both high availability and continuous operations.

### **High availability**

Dynamic transaction routing can help you to provide high availability by enabling you to clone application-owning regions (and terminal-owning regions). Initializing clones of application-owning regions in different MVS images allows the terminal-owning region to route transactions to another application-owning region in the event of either a CICS or an MVS failure.

Also, with multiple application-owning regions available, a dynamic transaction routing program in any terminal-owning region can bypass an application-owning region that is giving poor response times, or that may have a problem.

Faster restart is possible for an application-owning region when the workload is spread across a greater number of regions, minimizing the impact of any single failure. All these factors contribute to:

- Reducing the number users affected by any single failure
- Reducing the length of outages seen by an end user
- Reducing the incidence of queueing between CICS regions, caused by too high a demand for the available number of sessions.

### **Continuous operations**

Dynamic transaction routing helps you to achieve continuous availability by allowing you to:

- Quiesce an application-owning region and direct transactions to an alternate application-owning region
- Remove an application-owning region when it has been quiesced
- Stop and restart an application-owning region to apply service changes to the application-owning region when no transactions are active
- Add an application-owning region clone and notify the dynamic transaction routing program that a new target application-owning region is available.

The capability thus provided by dynamic transaction routing enables you to add and remove application-owning regions, and apply changes to application-owning regions, in a way that is transparent to end users. The kind of changes you might want to make to an application-owning region, while still maintaining service to end users, could be to introduce modified CICS tables, or to apply service to CICS code.

The *CICS Customization Guide* gives all the programming interface information you need to write a dynamic transaction routing program, together with some processing considerations. Alternatively, you can use the dynamic transaction routing program provided as part of the workload management component of CICSplex System Manager/ESA.

## **Simplified Systems Management**

Dynamic transaction routing simplifies systems management by helping you to manage departmental workloads, and facilitates the support of different versions of CICS and cut-over from test to production.

### **Managing departmental workloads**

Dynamic transaction routing enables you to separate workloads easily for financial (departmental accounting), security, or company policy reasons.

You can achieve separate workloads by:

- Cloning application-owning regions
- Performing workload separation in a dynamic transaction routing program, routing transactions to subsets of the cloned application-owning regions based on:
  - Transaction identifier (TRANSID), which allows workload separation by application
  - User identifier (USERID), which allows workload separation by user (even though different groups of users may be using the same application)
  - LU (or terminal) name (NETNAME or TERMID), which allows workload separation by geographical location (for example, different branch offices).

Separate workloads managed in these ways still allow you to retain the benefits of cloned application-owning regions, with equivalent regions providing simplified systems management.

### **Supporting different versions and cut-over to production**

Dynamic transaction routing enables you to offer different versions of an application to different users. It also allows you to cut over to a new version of an application while retaining the ability to back off the new version in the event of problems arising. These benefits of simplified systems management are achieved by:

- Cloning application-owning regions
- Performing workload separation in a dynamic transaction routing program, routing transactions to subsets of the cloned application-owning regions based on criteria such as user identifier and LU name.

Increasing the number of CICS regions by cloning terminal-owning and application-owning regions might be seen as causing an increase in the number of actions required to perform a given task against a CICS resource. For example, if you need to disable a transaction or discard a resource definition, you must do this in all the clones in which these resources reside. However, by using the single system image capabilities of CICSplex SM you can avoid any increase in the number of actions, and probably reduce them. See “CICSplex System Manager/ESA”.

## **CICSplex System Manager/ESA**

CICSplex System Manager/ESA (CICSplex SM) is a system management tool that enables you to manage the multiple CICS regions in a CICSplex as a single-system image.

The provision of a single-system image enables you to operate at the logical rather than the physical level, without regard to either the scale or location of CICS resources. That is, you can operate the CICSplex though it is a single CICS region. CICSplex SM provides a general-purpose interface to a variety of CICS system management functions that allow operators to manage multiple CICS regions, within a sysplex, from a single point of control.

The main functions provided by CICSplex SM are:

- Operating functions through a general-purpose operator interface

- Online access to CICS monitoring and statistics information across the CICSplex.
- Workload management functions that support both workload balancing and workload separation
- Real-time analysis functions that support external notification of unusual or otherwise interesting conditions.

The role of CICSplex SM in workload management, and how it interacts with MVS workload management, is covered in “Chapter 16. Planning for Workload Management” on page 143.

## Cross-System MRO

MRO provides communication between two or more CICS regions using CICS internal facilities and protocols. CICS supports MRO links between CICS regions that reside in the same MVS image, or in different MVS images.

The MRO access methods you can specify for all MRO links are IRC, which uses the CICS interregion program, and XM, which uses MVS cross-memory services.

CICS uses MVS sysplex services to support MRO links between regions that reside in different MVS images. This MRO access method, called XCF/MRO, enables you to configure your CICSplexes across multiple MVS images without incurring the performance penalty inherent in CICS intersystem communication (ISC). CICS uses XCF/MRO automatically when it detects that its MRO partners are in different MVS images.

---

## General Considerations

Other IBM products that use the coupling facility to provide benefits for applications using the sysplex are:

- VTAM
- RACF or the SureWay Security Server element of OS/390 or z/OS

## VTAM Support for the Sysplex

VTAM provides two functions that are of particular benefit to subsystems operating in a sysplex environment:

1. Generic resources
2. Persistent LU-LU sessions.

The distributed data facility of DB2 Version 4 and IMS Version 6 can make use of generic resources, and CICS support both of these VTAM functions. For CICS, you can use both functions together in the same CICS terminal-owning region. However, you could decide to use VTAM generic resources without persistent sessions. You should consider the benefits of each facility, how they interact, and the effect on your terminal users. See “Implementing VTAM Persistent LU-LU Sessions” on page 62.

### VTAM Generic Resources

The VTAM generic resources function enables several VTAM applications to be known by a single name—the generic resource name. At session start, the generic resource name is used in the logon request. Using the generic resource name, VTAM selects one of the real VTAM application names, registered as members of the generic resource group, to be the target of the session.

For the generic resources function to operate, each VTAM application must register itself to VTAM under its generic resource name. Registration is performed automatically by CICS when it is ready to receive VTAM logon requests.

CICS uses the VTAM generic resource name for authenticating user signon in all the terminal-owning regions, removing the need to define and maintain separate APPL resource class profiles for each terminal-owning region.

**Note:** The generic resources function is provided by a sysplex using advanced peer-to-peer networking (APPN). At least one VTAM in the sysplex must be an APPN network node, with the other VTAMs being APPN end nodes. Each VTAM must be connected to the coupling facility and be part of the same sysplex.

See “Chapter 4. Planning the Terminal-Owning Regions” on page 49 for more information about CICS support for VTAM generic resources, and see “Planning for Using VTAM Generic Resource Groups” on page 178 about IMS support for VTAM generic resources.

### **How DB2 Uses VTAM Generic Resources**

If you have OS/390, z/OS, or VTAM Version 4 Release 2 or later on the server, you can use VTAM's support for generic resources and have a generic 8-character name to represent a group of VTAM LU names. For example, generic name LUGROUP might represent three DB2 subsystems in the group whose real LU names are LUDB2A, LUDB2B, and LUDB2C.

Clients can make a simple change to their communications directories to refer to the DB2 data sharing group by the generic name. VTAM selects the real DB2 LU name to be used by the requester. VTAM makes this choice based on the number of active DDF sessions or the result of a user-written VTAM or MVS workload manager exit routine.

After a requester is associated with a particular LU in the data sharing group, all future requests from that requester are sent to the same member of the data sharing group until all connections between the two LUs are terminated.

For more information about using generic resources with DB2, see *DB2 for MVS/ESA Version 4 Data Sharing: Planning and Administration*.

### **VTAM Persistent LU-LU Sessions**

Persistent session support improves the availability of CICS. It exploits VTAM persistent LU–LU session support to provide restart-in-place of a failed CICS without the need for network flows to re-bind CICS terminal sessions.

CICS support of VTAM persistent LU-LU sessions retains LU–LU sessions for all VTAM supported sessions (except LU0 pipeline and LU6.1 sessions). After the failure of a CICS terminal-owning region, VTAM can determine from information passed by CICS the period of time over which the sessions should be retained. If a failed CICS is restarted within this time, it can use the retained sessions immediately without needing network flows to re-bind them.

See the *CICS Intercommunication Guide* for more information about running CICS with VTAM's persistent session support.

## Security

You can run CICS or IMS/ESA with RACF 1.9.2 as the external security manager. However, to obtain maximum value from the sysplex environment, you should consider the benefits of migrating to RACF 2.1, OS/390, or z/OS.

RACF 2.1 or the SureWay Security Server element of OS/390 or z/OS exploits the sysplex and provides the following benefits for IMS TM Version 5 and CICS/ESA 4.1 regions running in a sysplex:

- Faster retrieval of security profiles.

RACF 2.1 or the Security Server uses the MVS virtual lookaside facility (VLF) to cache control blocks associated with signed-on users, which gives a significant performance improvement in the reuse of the cached information.

- For CICS, there is no need to use the CEMT or EXEC CICS PERFORM SECURITY REBUILD command when adding new or changed security profiles to a running CICS region.

General resource class profiles are now held in storage common to all address spaces, which are refreshed by the RACF security administrator.

- Enhanced security for started tasks with JOB support (provided as a PTF for APAR OW02190).

This allows you to run CICS or IMS regions as started tasks using the same procedure, but with each region running under the authority of its own userid. RACF or the Security Server provides the STARTED general resource class with the STDATA segment to support started jobs.

---

## Naming Conventions

In the sysplex environment, where you are likely to have large numbers of subsystems installed, meaningful naming conventions are very important. The more subsystems you have to deal with, the more important it is to have good naming conventions.

Naming conventions are particularly important where you have multiple instances of a subsystem, such as CICS and IMS, many of which are connected through communication links. Even when subsystems are not directly connected, use a common naming convention for all subsystems that coexist within a sysplex. Wherever possible, design your conventions with as wide a scope as possible (for example, global scope within an entire corporate network).

See “Part 2. Migrating CICS Applications” on page 25 for the naming conventions used in the CICS examples used in this book.



---

## Chapter 2. Planning the Subsystem Configuration for a Sysplex

In the previous chapter, we gave some reasons why a sysplex environment provides an ideal environment for online applications. This chapter describes the kind of target configuration to which you should be planning to migrate, so that your transaction processing subsystems can fully exploit a sysplex environment.

The target configuration is designed to support the following types of workload, all of which can fully exploit the data sharing capability of the sysplex:

- CICS-DL/I applications

These are applications where the principal database is managed by DBCTL—it doesn't mean they don't also use other forms of data. The applications may also use a DB2 database, and VSAM data that is accessed either through an SMSVSAM server or through a CICS file-owning region.

- CICS-DB2 applications

These are applications where the principal database is managed by DB2—it doesn't mean they don't use also other forms of data. The applications may also use a DBCTL database, and VSAM data that is accessed either through an SMSVSAM server or through a CICS file-owning region.

- CICS-VSAM applications

These are applications where the main application data is in one or more VSAM data sets—it doesn't mean they don't use also other forms of data. The applications may also use a DBCTL database and DB2 database.

- IMS TM applications with either IMS DB or DB2 as the database manager.

The CICSplex shown in the target configuration is designed to support CICS-VSAM applications that access VSAM data through an SMSVSAM server. It is also possible to use a CICS file-owning region in this configuration where this is considered more appropriate.

Most CICS transactions use the CICS temporary storage facility. The sysplex configuration illustrated in Figure 1 on page 18 includes CICS temporary storage servers for use with non-recoverable TS queues. Where these are not appropriate, perhaps because your TS queues are defined as recoverable, you can add queue-owning regions (QORs). These provide an alternative method of shared access to the queues, to avoid potential intertransaction affinity problems. (For an explanation of CICS intertransaction affinities, see the *CICS Application Programming Guide*.)

---

### Prerequisite Hardware and Software

To implement the data sharing configuration illustrated in Figure 1 on page 18 to support IMS, DB2, and VSAM RLS data sharing, you need the prerequisite hardware and software.

For information about DB2 data sharing, see the DATABASE 2 Version 4 publication, *DB2 for MVS/ESA Version 4 Data Sharing: Planning and Administration*.

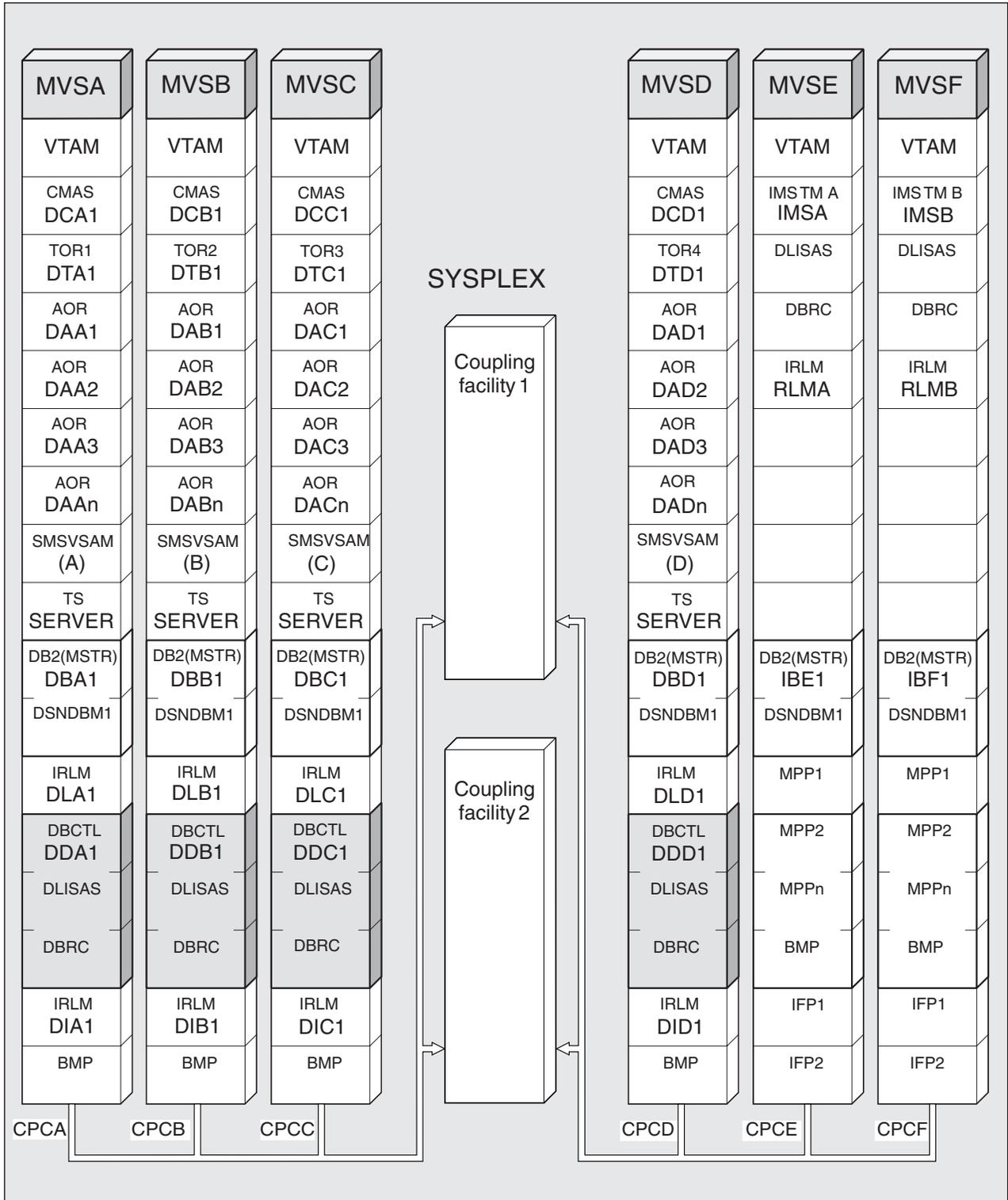


Figure 1. A Conceptual View of Data Sharing Subsystems in a Sysplex

**Notes to Figure 1:**

1. This configuration consists of a sysplex based on a S/390 9672 Parallel Transaction Server with 6 CPCs, each CPC running one MVS image, plus 2 CPCs used as coupling facilities. All the CPCs running under MVS have high-speed links to both coupling facilities.

2. There are 4 CICS terminal-owning regions assigned to the first four MVS images. To ensure maximum flexibility, all terminals in the network should have access to all application-owning regions (AORs), regardless of the terminal-owning region to which they log on. This means defining MRO connections between each TOR and every application-owning region. The links between TORs and AORs that reside in different images are XCF/MRO links.
3. The named AORs are labeled using the naming convention described in “Chapter 3. Planning Naming Conventions for CICS and Related Subsystems” on page 27 (DAA1–DAA3, DAB1–DAB3, DAC1–DAC3, and DAD1–DAD3, with additional AORS, as required, shown as DAA $n$  through DAD $n$ ).
4. All the application-owning regions have equal access to DBCTL, DB2, and VSAM data through connections with the DBCTL, DB2, SMSVSAM, and TS server subsystems that reside in the same MVS image.
5. All the CICS regions, regardless of type or special purpose, are considered to form one CICSplex.

---

## The Target Configuration

The remainder of this chapter describes a target data sharing configuration that is recommended for a Parallel Sysplex. The main subsystem components of this target configuration are described in “The Target Subsystems”.

The target configuration is designed with the following main objectives:

### Availability

To provide maximum availability of online applications by cloning as many of the sysplex components as possible. Currently, these include:

- The MVS images
- The VTAM nodes
- The transaction processing subsystems—CICS and IMS TM
- The DBCTL and DB2 subsystems, providing DL/I and DB2 multisystem data sharing.

### Capacity

To provide a growth capability that lets you add additional capacity without disrupting production work. The parallel transaction server-based sysplex provides this kind of non-disruptive growth capability, whereby you can add a new CPC or extra frames.

### System management

To provide better system management of multiple MVS images, with MVS clones making for easier installation and administration of additional MVS images.

The configuration shown in Figure 1 on page 18 is not symmetrical; that is, the MVS images and their subsystems are not exact clones. In the case of the CICS and IMS online subsystems, these are segregated as shown for illustrative purposes only. In a sysplex, you should aim for as much symmetry as possible, especially in a sysplex that is based entirely on a parallel transaction server. For example, there are no capacity reasons why you should not install a CICS terminal-owning region on every MVS image in the sysplex.

## The Target Subsystems

The target configuration shown in Figure 1 on page 18 comprises the following elements:

- One terminal-owning region in each of the MVS images MVSA through MVSD (TOR1 through TOR4)
- A minimum of 12 application-owning regions allocated across the 4 MVS images, MVSA—MVSD.

Although the transaction processing workload that runs on our Parallel Sysplex configuration is assumed to be a mixture of DBCTL-, DB2-, and VSAM-based applications, all the application-owning regions are capable of running all transactions. That is, they are clones of one another, and any workload separation is controlled by workload management policies and the CICS dynamic routing mechanism. For this reason, all the application-owning regions require a connection to all the data sharing subsystems (DBCTL, DB2, and SMSVSAM) in their respective MVS images. Note that only DBCTL and DB2 need connections to be defined and started. CICS registers automatically with SMSVSAM if the subsystem is present and RLS=YES is specified as a system initialization parameter. CICS also connects automatically to a temporary storage server as required.

- There are 4 DBCTL environments, allocated across 4 MVS images (MVSA—MVSD) to support the CICS-DL/I workload processed by the application-owning regions. Each DBCTL consists of a database control (DBCTL) address space, a database recovery control (DBRC) address space, and a DL/I separate address space (DLISAS).
- There are 4 DB2 subsystems, allocated across 4 MVS images (MVSA—MVSD) to support the CICS-DB2 workload processed by the application-owning regions.
- There are 4 SMSVSAM subsystems, allocated across 4 MVS images (MVSA—MVSD) to support the CICS-VSAM workload processed by the application-owning regions.
- There are 4 temporary storage subsystems, allocated across 4 MVS images (MVSA—MVSD) to support access to remote temporary storage queues that reside in shared TS pools.
- There are 8 IMS internal resource lock managers (IRLMs), one for each DBCTL and one for each DB2. (SMSVSAM takes care of its own lock management and does not use a separate address space.)

## The CICS Terminal-Owning Regions

With the VTAM generic resources function, all the terminal-owning regions in the CICSplex can be represented by one generic APPL name, and appear as one to terminal users. This means that, regardless of which application the users want, they logon to only a single CICS APPLID. VTAM generic resources resolves the generic name to the specific APPLID of one of the terminal-owning regions. Thus the CICSplex appears as a single system to the end user.

These terminal-owning regions are identical in every respect except in their external identifiers. This means that:

- They have different specific APPLIDs to identify them to VTAM and their partner MRO regions
- They each have a unique local name specified on the SYSIDNT system initialization parameter
- They each have a unique CICS monitoring subsystem identifier for RMF performance reporting, specified on the MNSUBSYS system initialization parameter.

Generally, apart from the identifiers listed above, you should try and make your terminal-owning regions identical clones, defined with identical resources—the same system initialization parameters, the same group-list of CSD definitions, and so on.

Exact cloning may not be possible if you have some resources that can be defined to only one region. For example, if your network needs to support predefined auto-connected CICS terminals, you have to decide which region such resources should be allocated to, and specify them accordingly. In this situation you cannot use exactly the same GRPLIST system initialization parameter to initialize all your terminal-owning regions. However, the GRPLIST system initialization parameter allows you to specify up to four group list names, which can include a generic symbol, making it easier to handle variations in CICS startup group lists.

The reasons for having multiple terminal-owning regions are as follows:

**For continuous availability**

You need to ensure that you have enough terminal-owning regions to provide continuous availability of the CICSplex.

Fewer users are impacted by the failure of one terminal-owning region. If a terminal-owning region fails, the users connected to other terminal-owning regions are unaffected, while the users of the failed region can re-logon immediately, using the VTAM generic resource name, without waiting for the failed terminal-owning region to restart.

**For performance**

To service a number of application-owning regions requires many MRO send and receive sessions. It is better to allocate the required sessions across a number of terminal-owning regions than try to load them all on to just one or two.

In the target configuration, we have tried to balance the number of subsystems and CICS regions to fully exploit MVS images running on multiprocessor CPCs. For example, if the CICSplex is running on a parallel transaction server that comprises 6-way CPCs, one terminal-owning region supporting 3 or 4 application-owning regions should be able to fully exploit the capacity of each CPC. The total number of CICS regions will depend on the average path length and number of DBCTL, DB2, or VSAM database accesses per transaction. For more information, see “Chapter 5. Planning the Application-Owning Regions” on page 65.

**For faster restart**

In the event of a terminal-owning region failure, not only are fewer terminal users affected when their sessions are allocated across multiple terminal-owning regions, but restarting the failed terminal-owning region is faster because of the smaller number of sessions to be recovered.

For more information about creating the terminal-owning regions, see “Chapter 4. Planning the Terminal-Owning Regions” on page 49.

## The CICS Application-Owning Regions

The application-owning regions are defined as sets, with each set comprising identical regions (AOR clones). Each set of clones should be capable of handling one or more different applications. Workload balancing and availability is achieved by the terminal-owning regions dynamically routing the incoming transactions to the best candidate application-owning region within a cloned set.

If you have split your CICS regions into separate regions based on application, the data sharing, workload balancing environment of the sysplex enables you to collapse regions together again. If your reason for splitting applications into separate regions is to provide some form of storage protection between applications, transaction isolation makes this no longer necessary.

For more information about creating the application-owning regions see “Chapter 5. Planning the Application-Owning Regions” on page 65.

## The CICS File-Owning Regions

There is no file-owning region (FOR) shown in our configuration, which assumes that VSAM data sets are accessed in RLS mode to achieve sysplex-wide data sharing.

You might need a file-owning region, however, to provide shared access to those data sets for which RLS is not suitable or not supported (for example, as in the case of BDAM data sets). If files are defined as remote files owned by an FOR, the AORs function-ship remote file requests to the FOR. This method of data sharing by function-shipping does not offer the same high level of availability as RLS, because a data set can be accessed through one FOR only, which becomes a single point of failure.

See “Planning the File-Owning Regions” on page 95 for information about creating a file-owning region.

## The CICS Queue-Owning Regions

Our CICSplex configuration shown in Figure 1 on page 18 does not include a queue-owning region (QOR). The configuration assumes that CICS temporary storage queues that need to be shared to avoid intertransaction affinities are held in TS pools and accessed through a CICS TS server.

If you have any recoverable TS queues (which are not supported by CICS temporary storage data sharing servers), that must be accessible through all the AORs, the solution is to include a queue-owning region in the CICSplex. Defining queues to the application-owning regions as “remote” queues, accessed through a queue-owning region, ensures that they are accessible by any application-owning region via function-shipping requests.

See “Planning the Queue-Owning Regions” on page 100 for more information about including a queue-owning region in your CICSplex.

## The IMS DBCTL Environment

To exploit IMS data sharing, our target configuration includes one IMS DBCTL environment in each MVS (MVSA—MVSD) that has CICS application-owning regions, with each application-owning region connected to the DBCTL in its MVS image.

Throughout this book, we assume that all the DBCTL subsystems are installed in MVS images running on a parallel transaction server.

See “Chapter 11. Planning for IMS DBCTL Multisystem Data Sharing with CICS” on page 109 for more information about DBCTL and data sharing in a sysplex environment.

## The DB2 Subsystems

To exploit DB2 data sharing, our target configuration includes one DB2 subsystem in each MVS that has CICS application-owning regions (MVSA—MVSD), with each application-owning region connected to the DB2 in its MVS image.

See “Chapter 12. Planning for DB2 Subsystem Access from CICS Regions” on page 115 for more information about handling DB2-related applications in a sysplex environment.

## The IMS Online Environment

Our configuration could also show an IMS online environment in one or more of the MVS images. These IMS subsystems can benefit from the data sharing capability of the sysplex and from the VTAM generic resources function (new in IMS/ESA Version 6).

Transactions that access IMS data entry databases (DEDBs) with sequential dependent segments or the virtual storage option (VSO) can run in only one IMS subsystem, that is, cannot participate in data sharing.

See “Chapter 18. Planning for IMS TM in a Sysplex Environment” on page 165 and “Chapter 19. Planning for IMS/ESA Version 6 in a Parallel Sysplex Environment” on page 175 for more information about IMS online systems in a Sysplex environment.

## The CICSplex SM Address Space

Our configuration shows a CICSplex SM address space (CMAS) in each MVS image that runs CICS regions (MVSA—MVSD). Together they provide a single-system image of the CICSplex shown in Figure 1 on page 18.

A CMAS is the hub of any CICSplex SM configuration, being responsible for the work involved in managing and reporting on CICS regions and their resources. A CICSplex can be managed by one or more CMAS—in our case we have installed one CMAS in each relevant MVS image.

The CMAS implements the monitoring, real-time analysis, workload management, and operational functions of CICSplex SM, and maintains configuration information about the CICS regions for which it is responsible.



---

## Part 2. Migrating CICS Applications

This deals with migration for CICS applications. It discusses the following topics:

- “Chapter 3. Planning Naming Conventions for CICS and Related Subsystems” on page 27
- “Chapter 4. Planning the Terminal-Owning Regions” on page 49
- “Chapter 5. Planning the Application-Owning Regions” on page 65
- “Chapter 6. Planning for VSAM Record-Level Sharing” on page 73
- “Chapter 7. Planning for Temporary Storage Data Sharing” on page 83
- “Chapter 8. Planning for Coupling Facility Data Tables” on page 87
- “Chapter 9. Planning for Named Number Counters” on page 91
- “Chapter 10. Planning Resource-Owning Regions” on page 95
- “Chapter 11. Planning for IMS DBCTL Multisystem Data Sharing with CICS” on page 109
- “Chapter 12. Planning for DB2 Subsystem Access from CICS Regions” on page 115
- “Chapter 13. Planning the Log Streams” on page 117
- “Chapter 14. Planning the Resource Definitions” on page 121
- “Chapter 15. Planning CICSplex Security” on page 137
- “Chapter 16. Planning for Workload Management” on page 143
- “Chapter 17. Planning the CICS Startup Procedures” on page 149

## Migrating CICS Applications

---

## Chapter 3. Planning Naming Conventions for CICS and Related Subsystems

The naming conventions described in this chapter are designed to support the following subsystems:

- CICS
- DB2
- IMS DB
- IRLM
- CICSplex SM

When you define a relationship between the subsystem components in an MVS image, use the same naming convention for all of them.

---

### What to Consider When Planning Naming Conventions

To plan your naming conventions, you need to consider the following points.

#### CICS

The naming convention described in this chapter is of particular benefit for those environments with large numbers of CICS regions using CICS multiregion operation (MRO).

Although such conventions may offer less immediate benefit for installations with a small number of CICS regions, standard, structured naming conventions are nevertheless encouraged for use in any installation using CICS and associated database subsystems.

They are considered essential for a CICSplex in a sysplex environment.

Most CICS users are operating multiple CICS regions within an MRO CICSplex, and the naming conventions used must therefore be suitable for CICSplex operations. They should also allow for more than one CICSplex in each sysplex.

#### IMS DB

In the Parallel Sysplex, IMS DBCTL, an environment of IMS DB, provides DL/I data sharing support for CICS.

A CICS region connects to a specific DBCTL: if CICS were to fail, that CICS region must always be reconnected to the same specific DBCTL when it is restarted. This is because any logical units-of-work that are **in doubt** at the time of the failure can be resolved only by reconnecting the restarted CICS region to the same specific DBCTL region. (See “Data Integrity Considerations” on page 99 for information about situations when data integrity may be in doubt.)

The naming convention should therefore support the relationship of specific CICS regions with a particular IMS DBCTL and should support multiple IMS DBCTLs.

#### DB2

For DB2 data sharing, consider the relationship between DB2 and its associated subsystems, such as IMS, CICS, and IRLM. In addition, all data objects (such as tables, table spaces and indexes) in a data sharing group must have unique names.

Ideally, your chosen naming convention should reflect the relationship between DB2 and its connected subsystems.

## IRLM

In a Parallel Sysplex environment, the database managers use the lock services of IRLM to ensure data integrity.

Each type of database manager within an MVS image requires its own IRLM, and the naming convention should show the relationship between the database managers and the IRLMs that support them. IMS and DB2 cannot share the same IRLM. In our target configuration shown in Figure 1 on page 18, the DB2 and DBCTL subsystems running in MVSA—MVSD each have their own IRLM.

Each IMS data sharing group needs at least one IRLM per MVS. If you have more than one DBCTL in the same MVS image, they can share the same IRLM, which is the recommended option.

## CICSplex SM

The CICSplex SM address space (CMAS) is the hub of any CICSplex SM configuration, and is responsible for most of the work involved in managing and reporting on its associated CICS regions and their resources. A CICSplex managed by CICSplex SM has at least one CMAS, and in our target configuration (shown in Figure 1 on page 18) we have installed one CMAS in each MVS image to manage the CICS regions in that MVS. These CMASs have the same recovery considerations as other subsystems, such as DBCTL and DB2, that are connected to CICS regions. As in the case of the database managers, your chosen naming convention should reflect the relationship between the CMASs and their CICS regions.

## Other Considerations

- Cloning
- CICS connectivity
- Workload management
- Recovery
- Full use of the character set
- Four-character naming restriction.

### Cloning

The naming convention you choose must support the replicating (or **cloning**) of MVS images and the subsystems, such as CICS and IMS, that run in them.

Cloning CICS terminal-owning regions and application-owning regions is a way of achieving MP exploitation and maximum availability, and makes it possible to balance a workload across a CICSplex when using dynamic transaction routing. Multiple terminal-owning regions and application-owning regions enable terminal users to log on to any terminal-owning region and to receive the same service from each one; they also enable any application-owning region in a set of identical application-owning regions to run any transaction defined to that set.

Cloning whole MVS images, as well as CICS regions and IMS subsystems, provides similar benefits for availability and workload balancing.

### CICS Connectivity

For maximum availability, all terminal-owning regions should have connectivity to all the application-owning regions. This allows terminal users to run transactions in any

application-owning region from any terminal-owning region, regardless of which MVS image the terminal-owning regions or application-owning regions reside in. MRO links between regions residing in different MVS images are supported by the cross-system MRO (XCF/MRO) access method.

The naming convention must be capable of supporting this multiplicity of connection definitions.

You can also use IMS Intersystem Communication (ISC) to connect other subsystems, such as IMS, to your CICS subsystems.

## Workload Management

The naming convention must also be capable of supporting methods of workload management.

In CICS, there are two aspects to workload management—*workload separation* and *workload balancing*—and it is assumed that both may be in operation within a CICSplex.

**Workload separation:** Although it is possible for all CICS application-owning regions to run all applications, you may want to separate some workloads—perhaps for reasons of security, or perhaps to create individual application-owning regions for special purposes. This is the concept of workload separation.

**Workload balancing:** Dynamic transaction routing, which is more flexible than static routing, requires the terminal-owning region to decide where to route an incoming transaction, based on performance considerations. This is the concept of workload balancing.

Workload balancing can be defined as the ability to handle a continuously varying workload efficiently. In a CICSplex managed by CICSplex SM, CICS, CICSplex SM, and MVS all cooperate to provide goal-oriented workload balancing for the CICS workload. See “Chapter 16. Planning for Workload Management” on page 143 for more information about goal-oriented workload balancing for CICS.

## Recovery

The naming conventions described here are designed for those subsystems that are connected in some way and have an operational relationship, such as CICS with IMS DBCTL or DB2, and IMS with IRLM or DB2.

MVS provides automatic restart management for failed subsystems, components, and applications. This facility plays an important part in the availability of key MVS components and subsystems, which in turn affects the availability of data. When a subsystem such as CICS, DB2, or IMS DB fails, it might be holding locks that prevent other applications from accessing the data they need. MVS automatic restart management quickly restarts the failed subsystem; the subsystem can then resume processing and release locks, making data available again to other applications. The naming convention, therefore, must enable automatic restart management to identify relationships between subsystems. For this purpose, our naming convention uses one character to identify the recovery **group** to which subsystems belong. See the use of the letter G in the naming convention template, described in 33.

There may be cases when there is no relationship—for example, terminal-owning regions have none with other subsystems such as IMS DBCTL. However, even for terminal-owning regions you might want to associate them with the other CICS regions in the same MVS image.

Also, application-owning regions do not have a restart relationship with each other, but can be related through use of the same DB2 or DBCTL.

### Full Use of the Character Set

All subsystem names can be created from the characters 0 to 9, A to Z, \$, @, and #. Blanks are allowed for trailing characters. Lowercase characters are treated as uppercase.

### Four-Character Naming Restriction

MVS subsystem names are restricted to 4 characters. This restriction also applies to system names in CICS, such as SYSIDNT and CONNECTION names (TERMIDs are also restricted to 4 characters). This significantly restricts the possible combinations for naming.

Because some subsystem names are defined at installation time, you need to give very early consideration to the names you are going to choose—changing names later can be difficult. Table 2 describes how the various CICS, DB2, IMS DBCTL, IMS, and IRLM systems are defined, and how the names are used between the subsystems.

Table 2. Application Subsystem Names

Subsystem	Where defined initially	Used by
CICS	In a CONNECTION resource definition and on the SYSIDNT system initialization parameter. <b>Note:</b> This system name is not used as the formal MVS subsystem name in the IEFSSNxx PARMLIB member. CICS is always defined in the subsystem name (SSN) table as "CICS".	Not required, either by CICS or other subsystems, but is recommended for MRO connection definitions. See "Chapter 14. Planning the Resource Definitions" on page 121 for examples of such use.
DB2	In the IEFSSNxx PARMLIB member; and in the DB2 resource control table.	CICS, when specified on the CICS DB2 connection command to override the DB2 name specified in the DB2 resource control table.
IMS DB	In the IMS system generation macros.	CICS, when using IMS DBCTL. Specify the IMS DBCTL name either in the database resource adapter (DRA) table or in the IMS connection command override. Also used by IMS TM.
IRLM	In the IEFSSNxx PARMLIB member; during IMS system definition; in the DB2 resource control table.	IMS online systems, IMS DBCTL, and DB2 subsystems.

---

## Designing Effective Naming Conventions

Naming conventions should be designed with clearly defined objectives in mind, such as those that follow.

### **Be clear and meaningful**

Ideally the naming conventions should clearly indicate:

- The type of subsystem, such as the type of CICS region, the type of database manager, and so on
- Whether the subsystem has an association with any other subsystem(s) such as a CICS region with its IMS DBCTL
- How many subsystems there are of that particular type, such as the number of CICS application-owning regions
- The CICSplex that the CICS region is a member of
- The other regions to which a CICS region is connected.

A good, meaningful, naming convention should make it possible to determine, where applicable, all of the above from the various subsystem names.

### **Aid problem determination**

Naming conventions should aid the problem determination task. For example, if a transaction abends, it is helpful to be able to derive useful information from the SYSIDNT, such as whether it is a terminal-owning region or an application-owning region; which MVS image the transaction is running on; and so on. (The naming convention described in 31 does not include a specific element to indicate the MVS image, but the recovery group code can be taken to indicate the preferred MVS image.)

### **Aid replication of system definitions**

The use of cloning requires naming conventions that enable you to clone CICS regions, and IMS DBCTL and DB2 subsystems, with the minimum of effort (that is, the minimum number of changes to resource definitions, system initialization parameters, and so on).

### **Be simple**

Keep the names as simple as possible so that they are easily identified.

### **Aid automation**

Naming conventions should satisfy automation requirements. For example, if the CICS application-owning regions AOR1, AOR2, and AOR3 are currently active and more CICS capacity is required, automatic operations products can easily determine that the next CICS system to start should be AOR4.

### **Aid command abbreviation**

Naming conventions should allow the use of a “wildcard” character, such as an asterisk (\*). For example, when our suggested naming convention is used, issuing the CICS CEMT INQUIRE CONNECTION(D\*) command in a CICS region returns all the connections from that region to other regions in the same CICSplex.

### **Allow for growth**

If you are running multiple CICS systems in a sysplex, where it is essential to have the ability to add new regions to enable effective workload balancing, it is important that the naming conventions you adopt allows for growth.

---

## **The Naming Convention**

The naming convention discussed here is designed for 4-character names, and is based on the following 4-character template:

C T G I

**Note:** For those CICS names that can be up to 8 characters long, the general recommendation is that the letters CICS are used for the first 4 characters, particularly for production regions. The *C* symbol (see below) is designed to identify any logical collection of subsystems, such as an entire CICSplex and its associated subsystems; it can be used to distinguish test, maintenance, and other (non-production) subsystems.

For some subsystems, such as CICS, the structure is hierarchical with *C* as the highest level in the structure and *I* as the lowest. DB2 is an exception, as explained in the description of the template letter G in 33.

The CTGI template has the following meaning:

**C** identifies a logical **collection** of subsystems, such as a **CICSplex** and any associated subsystems such as IMS DBCTL and DB2.

This is a high-level qualifier that enables multiple subsystems to be logically grouped together. Some possible reasons for doing this are:

- **Company separation**—where you are running more than one company's transactions within a sysplex environment, but with each company using its own set of subsystems, such as a CICSplex and associated IMS DBCTL, DB2, and IRLMs. The application workload could also be separated at the CICSplex level on a different basis, such as by department, or by geographical location. For example, if a regional data center has only one large CICSplex, the initial character can be used to denote the geographical region, such as D for the Dallas CICSplex.

- **Replication of specific business applications**—for example, an insurance application. The application could then be installed in different CICSplexes at the same, or different, regional data centers, accessible via APPL names that are distinguished by a unique CICSplex identifier.

This would enable data centers to be merged on one sysplex at a single location, where the CICSplexes would continue to be identified by the unique CICSplex identifier.

- **Separation of test and production regions.**

**T** identifies the **type** of region, or subsystem, where:

**A** indicates an application-owning region (AOR).

**B** indicates a DB2 subsystem.

**C** indicates a control-owning region (COR), such as a CICSplex SM CICS managing address space (CMAS).

**D** indicates a CICS database-owning region (DOR), or an IMS DBCTL subsystem.

A CICS local DL/I region should be used only until the databases can be migrated to IMS DBCTL.

**F** indicates a file-owning region (FOR).

**I** indicates an IRLM that supports an IMS TM or IMS DBCTL subsystem.

**L (or J)** indicates an IRLM that supports a DB2 subsystem.

**P** indicates a printer-owning region (POR).

- Q** indicates a queue-owning region (QOR) for temporary storage, transient data, or both.
- R** indicates a general resource-owning region (ROR).
- T** indicates a terminal-owning region (TOR).

**G** identifies a **recovery group**, or a DB2 subsystem that is a member of a DB2 data sharing group.

This part of the naming structure enables you to group those subsystems that have some intersystem relationship with each other. It is relevant to recovery operations, where subsystems in a recovery group have some recovery and restart considerations in common.

Affinity groups and independent groups are characterized as follows:

- Affinity groups
  - These are groups of subsystems that are associated with each other in some way; for example, a group of CICS application-owning regions that have an affinity with a particular database manager subsystem such that they always need to connect to the same subsystem, such as IMS DBCTL or DB2. When one region in a recovery group is moved to a new MVS, they must all be moved.
- Independent groups
  - The regions in such groups have no affinities with each other, apart from being the same type of region; for example, terminal-owning regions. In this case, you could choose 0 (zero) as the group character, or one of the #, \$, or @ symbols.

For example, if a Dallas CICSplex supports five terminal-owning regions, they can be identified under our naming convention as DT01 through DT05. The numbers 1 through 5 identify the cloned terminal-owning regions, and the 0 indicates there is no group relationship, either between the terminal-owning regions or between the terminal-owning regions and any other subsystem.

**Notes:**

1. For terminal-owning regions to be true clones, they should all have identical links to all the application-owning regions.
2. Although we have used here the example of terminal-owning regions as being regions that may have no special recovery relationships, system management considerations may dictate otherwise. For example, if you use automatic recovery procedures, you might want to assign to the terminal-owning regions the same recovery group letter that you assign to the other CICS regions that reside on the same MVS, as we have done in our target configuration, shown in Figure 1 on page 18.

**I** identifies **iterations** or, for DB2, it's the group identifier.

This part of the naming structure allows regions within groups to be named iteratively. For example, using the lexical sequence 1–9, and then A–Z, it is possible to have 35 instances of a particular type of region in a group. If more than 35 are required, 2 or more different “G” group characters could be used, which could logically have the same meaning.

You can use 1 when there is only one instance of a subsystem, which may be the case if you have only one file-owning region or only one queue-owning region.

Alternatively, you could use the characters #, \$, @, or 0 to denote that there are no iterations within the group.

---

## Applying the Naming Conventions for CICS

Using the CTGI template described above, it is possible to construct the principal names used by all the subsystems in a sysplex.

### For CICS VTAM APPL Names (APPLIDs)

You need to consider the definition of VTAM names to several subsystems—to CICS, to VTAM, and to RACF, as follows:

- You can specify two forms of the VTAM APPL name to CICS:
  - The VTAM APPL name that is unique to one specific CICS region (VTAM application).  
This is specified on the APPLID system initialization parameter, as in previous releases of CICS.
  - The VTAM generic resource name that applies to a group of CICS regions, such as an entire CICSplex.  
This is specified on the GRNAME system initialization parameter.
- In VTAM, you specify the application program major nodes (APPLs). If you use generic naming for your APPL definitions for CICS, you avoid the need to create a specific APPL definition for each CICS region.
- In RACF, you can define APPL resource class profiles for APPLIDs, to control which users can access a VTAM application (CICS region).

You should consider all these related requirements when planning your naming strategy for CICS APPLIDs.

#### Specific CICS APPLIDs

You can use the CTGI template, together with some suitable prefix letters, to create the CICS APPLID.

Thus, with CICS as the prefix, the APPLID is of the form:  
CICSCTGI,

In all the examples shown in this book we have used the letters CICS as the prefix. You might want to use prefix characters that conform to some other convention, such as network naming standards. However, we recommend that you use the same first four characters for all the regions in the CICSplex.

You can see several examples of APPLIDs based on this convention later in this chapter.

Applying the naming convention should ensure that APPLIDs are unique within a VTAM network.

#### VTAM Generic Resource Names

The VTAM generic resource name is a name by which a group of CICS terminal-owning regions in a CICSplex are known to VTAM. You specify the name on the CICS GRNAME system initialization parameter, and CICS registers itself to VTAM as a member of the generic resource.

The generic name enables terminal end users to log on to a CICSplex as if it was a single VTAM application. As each logon request to a CICS generic resource name

is received, VTAM selects the specific APPLID of one of the CICS terminal-owning regions that registered as a member of the generic resource.

The recommended form for GRNAME operands is the same as for specific APPLIDs: use *CICS* as the first 4 characters followed by the identifying character for the CICSplex (for example, CICS<sub>D</sub> for the Dallas CICSplex). You are also recommended to pad the name to the full 8 characters with a generic symbol, as follows:

```
GRNAME=CICSD###
```

You should apply the VTAM generic name to only the CICS terminal-owning regions in a CICSplex.

### VTAM APPL Definitions

When defining a VTAM application program major node (APPL), you normally specify the full APPL name on the APPL definition statement.

Instead of defining a large number of specific APPL names, VTAM allows you to use a “wildcard” symbol in the APPL name to denote a generic definition that can be shared by a number of CICS regions. When you use this facility, VTAM uses the generic VTAM APPL definition whenever the generic name matches the appropriately masked APPLID specified on the CICS OPEN VTAM ACB request.

For example, you could use the following definition for the set of terminal-owning regions in the Dallas CICSplex:

```
*****
* Generic APPL Definition for All CICS Regions in the DALLAS CICSplex
*****
CICSD*  APPL  AUTH=(ACQ,VPACE,PASS),VPACING=0,EAS=5000,          X
        PARSESS=YES,SONSCIP=YES
*****
```

In the example, the definition would be used by VTAM for all the CICS terminal-owning regions that log on to VTAM with specific APPLIDs that begin with the letters CICS<sub>D</sub>.

Note that the name you use for the VTAM generic APPL definition does not have to relate in any way to the generic resource name you specify on the CICS GRNAME system initialization parameter. However, it is recommended that you use the VTAM generic APPL name as the basis of the generic resource name you specify on the GRNAME parameter. For example, if CICS<sub>D</sub>\* is defined on a VTAM APPL definition, use CICS<sub>D</sub> as the first five characters of the GRNAME parameter for the CICS terminal-owning regions, then pad this to 8 characters with the # symbol to give CICS<sub>D</sub>### as the full generic resource name.

**For Application-Owning-Regions:** If you also need to define APPL definitions for your application-owning regions, create a separate generic APPL for these. This is because you need to specify a low value on the EAS parameter for the application-owning regions compared with that of the terminal-owning regions. The following is an example of an APPL definition. You could use it for the application-owning regions in the same CICSplex as the terminal-owning regions in the previous example.

```
*****
* Generic APPL Definition For All CICS Regions in the DALLAS CICSplex
*****
```

```
CICSDA*  APPL  AUTH=(ACQ,VPACE,PASS),VPACING=0,EAS=10,          X
          PARSESS=YES,SONSCIP=YES
*****
```

### RACF APPL profiles

In terminal-owning regions that specify a VTAM generic resource name, the VTAM APPL check that CICS makes during signon processing (to verify that the user is authorized to sign on to a CICS region) is made using the VTAM generic resource name. Therefore, for VTAM generic resources, you must create a RACF profile in the APPL resource class using the generic resource name specified on the GRNAME system initialization parameter as the profile name.

For example, if the VTAM generic resource name is CICSD###, the RDEFINE command to create the APPL profile is as follows:

```
*****
*          APPL Generic Resources Profile for the DALLAS CICSplex
*****
RDEFINE  APPL CICSD###  UACC(NONE)  NOTIFY(userid)
*****
```

See the *CICS RACF Security Guide* for more information about controlling access to the VTAM APPL of a CICS region.

### For SYSIDNT Names

The SYSIDNT system initialization parameter defines the private name by which a CICS region identifies itself. For this name, use the characters defined by the CTGI template.

Using the CTGI template enables you to create SYSIDNTs that are unique within a sysplex, and also unique across sysplexes if a CICSplex spans more than one sysplex.

**Note:** Unique SYSIDNT names are not required nor enforced by CICS, because the SYSIDNT is known only to the CICS region in which it is defined.

### For CONNECTION Names

To define a link between two CICS regions requires two CONNECTION definitions, one connection definition being installed in each linked CICS region. For example, to define the link between a terminal-owning region and an application-owning region, you must:

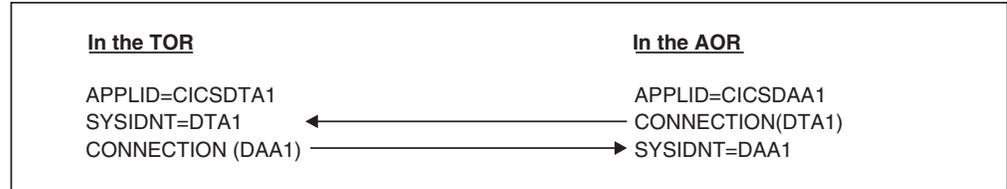
- Define and install a CONNECTION definition in the terminal-owning region that defines its link with the application-owning region, and
- Define and install a CONNECTION definition in the application-owning region that defines its link with the terminal-owning region.

Provided the SYSIDNTs are unique within a CICSplex, you can use the SYSIDNTs as the connection names. Use the SYSIDNT defined for a CICS region as the name of the CONNECTION definition installed in its partner region.

An example based on the CTGI convention is shown in Figure 2 on page 37, where:

- The CICSplex is D (for Dallas).
- The region types are T and A for the terminal-owning region and application-owning region, respectively.
- Each region is the first, or only, member in its group, identified by the numeral 1.

**Note:** Although a terminal-owning region does not usually have any affinity with any particular subsystem (such as IMS DBCTL or the CICS application-owning regions), it can be useful to consider it as being in the same group as the other CICS regions in the same MVS image. Hence our example uses the group code A, the same as the application-owning region.



*Figure 2. Example of Connection Naming Between Terminal-Owning Region and Application-Owning Region*

Connection names should be unique within a CICSplex.

Using the CTGI template enables you to create CONNECTION names that are unique within a sysplex, and also unique across sysplexes if a CICSplex spans more than one sysplex.

**Note:** A connection name installed in a CICS region cannot be the same as the SYSIDNT of that CICS region.

### Generic Connection Names

For situations where complete business applications are operated in more than one CICSplex (perhaps running in different data centers), you can use a generic character in the connection name. Using this modification to the standard naming convention, you can assign a generic character as the CICSplex identifier to represent a number of identical CICSplexes, instead of a specific character for each instance of the CICSplex. In this way you can reduce the number of CICS resource definitions you need to maintain.

It is recommended that you use a character from the set of #, @ or \$ symbols. This avoids using an alphabetic character that might conflict with a chosen alphabetic sequence.

Using a generic connection name enables CICS remote resource definitions to be portable across CICSplexes. For example, a remote file resource definition can refer to one of these generic connection names. You can then install this file resource definition in as many regions, in as many CICSplexes, as you want, thus reducing the number of file resource definitions you need to maintain. This technique is suitable for all resources that can be remote: files, programs, temporary storage and transient data queues. However, it does not apply to remote transactions because the use of dynamic transaction routing means that you do not need to specify a remote system name.

The technique can also be useful with application programs that specify a remote system name on an EXEC CICS command. The same program could be installed in more than one CICSplex, without the need to maintain separate versions that specify explicit SYSIDs.

The connection definition itself cannot be cloned, because it contains the NETNAME of the system that the connection refers to, and the NETNAMEs should be unique. However, you can define connection definitions with the same name but

different NETNAMEs, provided they either reside in different CSDs or in different groups within the same CSD. In general, you should use the same CSDs for all the CICS regions in a CICSplex, only using a different CSD for a different CICSplex within the same MVS sysplex.

The use of generic connection names is illustrated in Figure 3—see also the notes that follow the figure.

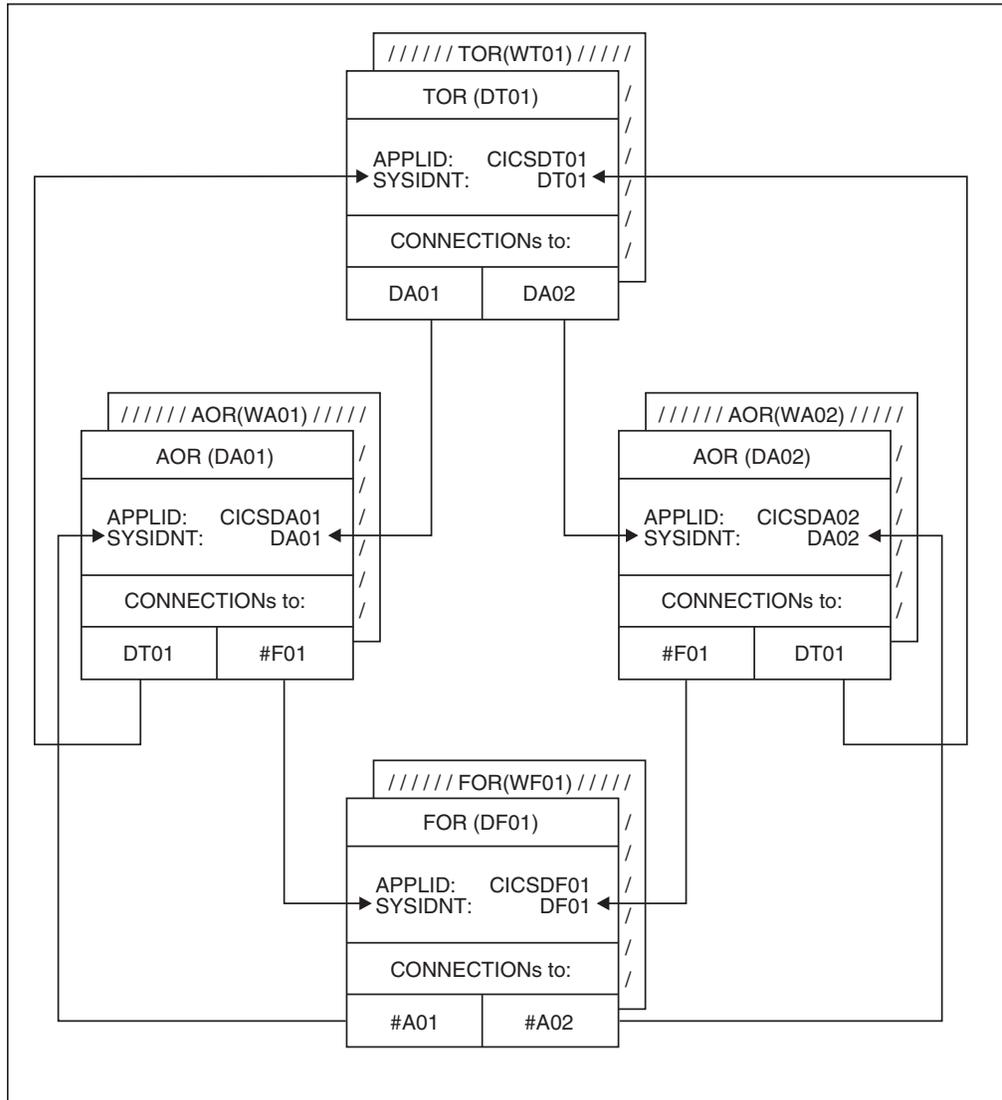


Figure 3. Example of generic connection naming

**Notes:**

1. In Figure 3, the CICSplexes are identified as D (for Dallas) and W (for Washington) in all except the CONNECTION names for the AOR–FOR links.
2. The connection definition names for the links between the application-owning regions and the file-owning regions can be common for sets of application-owning regions and common for sets of file-owning regions, in every respect except for the NETNAME.
3. Remote file definitions in the application-owning regions in both CICSplexes need to specify only #F01 as the REMOTESYSTEM name.

## For SESSIONS Definitions with and without Prefixes

CICS regions communicate over sessions carried on a connection linking the regions. The number of sessions available on any connection is defined in the sessions resource definition associated with the MRO connection. CICS generates the 4-character terminal identifiers it requires for each send and receive session, each of which is represented by an entry in the terminal control table (TCT).

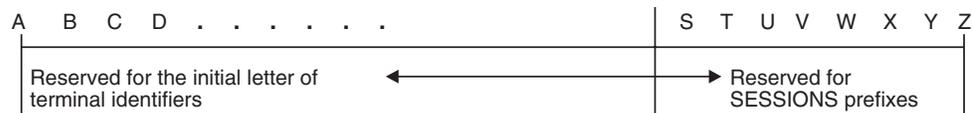
In releases earlier than CICS for MVS/ESA 4.1, CICS uses 2-character send and receive prefixes, together with the number of send and receive sessions for a link, to create the TCTTE identifier (termid) for each session. This can limit the choice of characters available for these session identifiers, because you can use only two characters (for the prefixes), the remainder of the termid being a number that CICS increments to the specified number of sessions.

In CICS 4.1 and later, this limitation does not apply because CICS generates the whole of the termid for the MRO sessions. The generated termids comprise a 3-character identifier, prefixed by a > symbol for the send sessions and a < symbol for the receive sessions. The 3-character identifiers begin with AAA for the first connection, and continue in ascending sequence until the number of session entries reaches the limit set by the send and receive counts. The sequencing continues for the second and subsequent connections. Therefore, after CICS 4.1, a naming convention for sessions is unnecessary, although you can continue to specify a prefix if you want to.

Because you can create a CICSplex that mixes earlier releases of CICS with CICS 4.1 and later, the following naming convention for sessions is suggested for use with the earlier releases, and for those users who continue to use the old prefix method.

### Defining Sessions With Predefined Prefixes

Because all the terminal identifiers in a TCT must be unique, it is important that you choose session terminal prefixes that do not conflict with terminal identifiers chosen for real terminals (either explicitly defined ones, or identifiers selected by the terminal autoinstall program). One possible solution is to reserve letters at the end of the alphabet for the first character of session prefixes, and ensure these are never used for the initial character of other types of terminal. For example:



The second character of the send/receive session prefix represents the iteration number for cases where terminal-owning regions have links with multiple application-owning regions, and application-owning regions have links with multiple terminal-owning regions, and so on.

In our examples of the naming convention we have reserved the letters S, T, U, V, W, X, Y, and Z. They are used for the initial prefix characters for all sessions between terminal-owning regions, application-owning regions, file-owning regions, and queue-owning regions; allocated to send and receive sessions as shown in Figure 4 on page 41 and Figure 6 on page 42. For ease of illustration, we have used the following terms to describe the links between regions:

#### Down links

These describe the links:

- From terminal-owning regions to application-owning regions

- From application-owning regions to file-owning regions, queue-owning regions, and resource-owning regions,

These *down links* are assigned the send and receive session prefix letters Y and Z.

### **Up links**

These describe the links:

- From application-owning regions to terminal-owning regions
- From file-owning regions, queue-owning regions, and resource-owning regions to application-owning regions

These *up links* are assigned the send and receive session prefix letters W and X.

### **Right links and left links**

These terms are used to describe links between application-owning regions. Where application-owning regions communicate with each other, for purposes such as linking to remote application programs using distributed program link commands, links are referred to as lateral *right links* and *left links*:

- *Right links* are assigned the letters U and V.
- *Left links* are assigned the letters S and T.

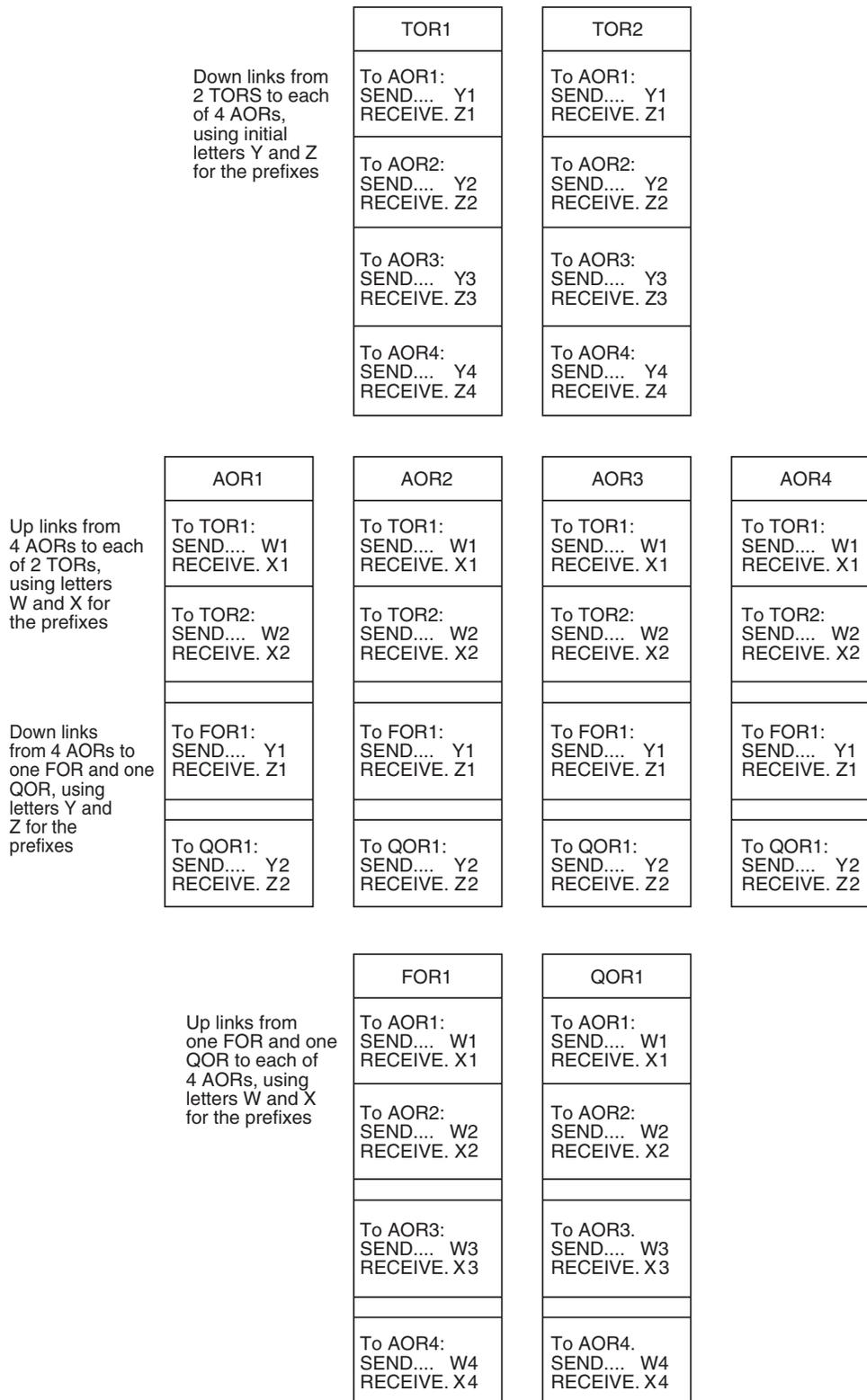


Figure 4. Allocation of SEND/RECEIVE Prefixes for Down and Up Links

The MRO configuration using the SEND/RECEIVE prefix conventions described in Figure 4 would give the region-to-region connectivity shown in Figure 5 on page 42.

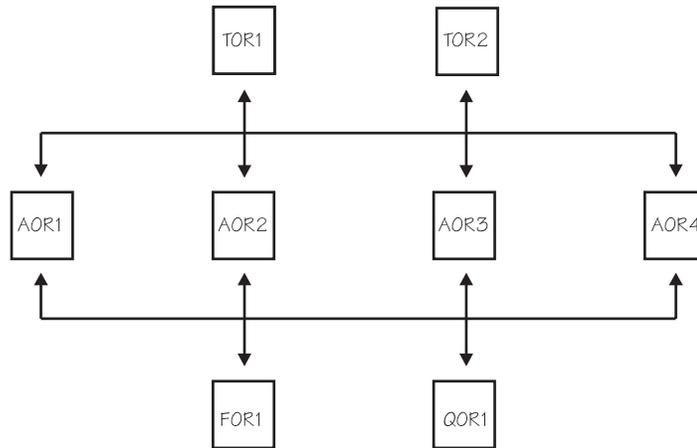


Figure 5. Sample MRO Configuration

Figure 6 illustrates the use of the letters S, T, U, and V for the lateral right and left links between AORs.

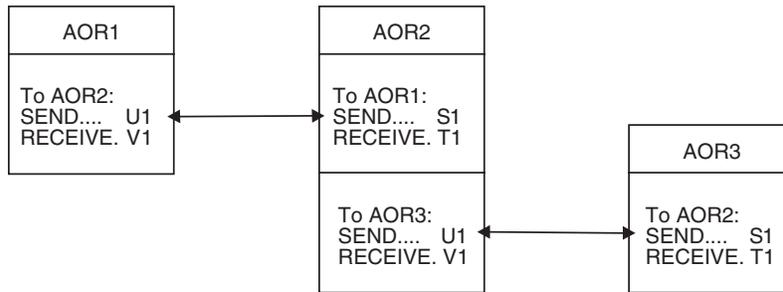


Figure 6. Allocation of SEND/RECEIVE Prefixes for Lateral Links

## For TERMINAL Names

For a CICSplex environment with multiple terminal-owning regions, you are strongly recommended to make the 4-character CICS terminal identifiers (termids) unique across the whole CICSplex. In CICSplex with a single terminal-owning region, termids need to be unique only within the terminal-owning region.

In a CICSplex with multiple terminal-owning regions, however, problems could arise over naming temporary storage queues if the termid were used as part of the queue name (which is not uncommon). For example, if two terminals connected to different terminal-owning regions are assigned the same termid, there could be a conflict over temporary storage queue names. This is particularly important in a sysplex where one of the CICS regions is a queue-owning region (QOR) managing all the temporary storage queues for a CICSplex. If two terminals try to access the same queue instead of different queues, the results are unpredictable.

The following techniques will help to ensure unique terminal naming:

1. Ensure that NETNAMEs are unique. Ensure also that their characters are arranged in such a way that the CICS autoinstall exit can extract a part of the netname to construct a 4-character termid that remains unique within the CICSplex.
2. Divide the full range of terminal names into discrete subsets, and assign a different subset to each terminal-owning region in the CICSplex. For example, assign to TOR1 a pool of terminal names in the range AAAA to DZZZ, and assign to TOR2 a pool of terminal names in the range EAAA to HZZZ. The autoinstall user program can then choose an unused termid from the appropriate pool, and no other terminal-owning region can have chosen the same name.

However, this second solution can present problems for transactions initiated by EXEC CICS START commands that specify delay interval and a TERMID.

## For CICS JOB Names

You can start a CICS region either as a batch job or as a started task. You may need to modify the naming convention for each method.

### Batch Jobs

Use the CICS region APPLID for the job name. This can make it easy to identify all CICS systems in the sysplex. For example, the SDSF DA panel (“Display active users of the system”) shows the jobname as the first field in the display of active users.

You could build one common procedure for all CICS regions, or alternatively create one for each type of region (designated by the letter T in the naming convention), using CICS as the first four characters of the procedure name. If you are using symbolic parameters for values such as data set qualifiers and region size, and a suffix or similar qualifier for a SYSIN data set, your procedures can be called by a variety of jobs or job steps. See the CICS-supplied sample procedure, DFHSTART, for an example.

### Started Tasks with JOB support

Use one common procedure for all CICS regions or alternatively create one for each type of region (designated by the letter T in the C T G I template), using CICS as the first four characters of the procedure name. On the START command, use the CICS APPLID on the JOBNAME parameter. This makes for ease of identification when you are using facilities such as SDSF.

For example, if a procedure contains the following statements and parameters:

```
//CICSTASK EXEC PGM=DFHSIP,REGION=150M,
//          PARM='START=&START,SYSIDNT=&SYSIDNT,SYSIN'
//* The SYSIN data set containing system initialization parameters
//SYSIN    DD DISP=SHR,DSN=&DSINDEX..CICSD###.SYSIN(CICS&CLONE)
//*
//* The temporary storage data set
//DFHTEMP  DD DISP=SHR,DSN=&DSINDEX..CNTL.CICS&SYSIDNT..DFHTEMP
```

the command for a terminal-owning region that has an APPLID of CICSDTA1 could be:

```
START CICSTASK,JOBNAME=CICSDTA1,DSINDEX=CICS410,SYSIDNT=DTA1,CLONE=DT##,START=AUTO
```

We recommend that you use the JOBNAME parameter on the START command. This enables you to use the JOB support for started tasks and started jobs.

For information about issuing multiple START commands, see “Cloning CICS Regions” on page 134.

## For Data Set Names

The *CICS System Definition Guide* recommends a naming convention for CICS data sets, in which the APPLID is used as one of the qualifiers. Our naming convention supports this recommendation.

The following are the symbolic parameters, for use in data set names, that we use in our CICS startup procedure shown in “Chapter 17. Planning the CICS Startup Procedures” on page 149:

- INDEX1** Represents the high-level data set qualifier. In our startup procedure shown in Figure 43 on page 159 we use the CICS release identifier in the form CICS410.
- SYSIDNT** Represents the local name for each CICS region, and we also use this to represent the last four characters of the second-level data set qualifier.

For example:

```
//DFHGCD DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHGCD
```

Using the C T G I convention, assigning the value DTA1 to &SYSIDNT identifies this data set as belonging to a CICS region with an APPLID of CICSDTA1 (a terminal-owning region in a CICSplex identified by the letter D for Dallas).

If you have to apply the naming convention to all the CICS system data sets, the DD statements you might need for the CICS startup procedure are shown in Figure 7.

```
/* The global catalog data set
//DFHGCD DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHGCD
/* The local catalog data set
//DFHLCD DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHLCD
/* The temporary storage data set
//DFHTEMP DD DISP=SHR,DSN=&INDEX1..CNTL.CICS&SYSIDNT..DFHTEMP
/* The transient data intrapartition data set
//DFHINTRA DD DISP=SHR,DSN=&INDEX1..CNTL.CICS&SYSIDNT..DFHINTRA
/* The restart data set
//DFHRSD DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHRSD
/* The journal archive control data set
//DFHJACD DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHJACD
/* The XRF control data set
//DFHXRTL DD DISP=SHR,DSN=&INDEX1..CNTL.CICS&SYSIDNT..DFHXRTL
/* The XRF message data set
//DFHXMSG DD DISP=SHR,DSN=&INDEX1..CNTL.CICS&SYSIDNT..DFHXMSG
/* The transaction dump data sets
//DFHDMPA DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHDMPA
//DFHDMPB DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHDMPB
/* The auxiliary trace data sets
//DFHAUXT DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHAUXT
//DFHBXT DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHBXT
/* The system log data sets
//DFHJ01A DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHJ01A
//DFHJ01B DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHJ01B
//DFHJ01X DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHJ01X
```

Figure 7. Unique CICS Data Sets Required for Each Region

---

## Sharing CICS System Data Sets

For the common data sets that can be shared between CICS regions, you do not need the second-level data set qualifier to identify the CICS regions to which the data sets belong. Typically, there are three data sets in this category:

- The CICS system definition (CSD) data set
- The journal partitioned data set, DFHJPDS
- The SYSIN data set.

### CICS System Definition Data Set

The CSD data set holds the CICS resource definitions. Where multiple regions are part of the same CICSplex, we recommend that all regions share the data set. Most CICS regions in a CICSplex require the same resource definitions (for example, for transactions), and it is good practice to share the data set, and so minimize the overhead involved in maintaining definitions. See Figure 8 for an example of a data definition (DD) statement for the CSD.

As an alternative to having a DD statement for the CSD in the startup procedure, you can specify the data set name and disposition option using the CSDDSN and CSDDISP system initialization parameters. For example, if you specify CSDDSN=CICS410.CICSD###.DFHCSD and CSDDISP=SHR in the SYSIN data set that contains the system initialization parameters for each region, CICS file control uses dynamic allocation to allocate the data set. This is the recommended method for defining the CSD to CICS.

### CICS Journal Partitioned Data Set

The CICS journal data set, DFHJPDS, holds the skeleton JCL for automatic journal archiving. Not only can this data set be shared between all the regions in the CICSplex, but the member containing the archive job can also be shared.

### SYSIN Data Set

The CICS SYSIN data set holds the system initialization parameters that are common to cloned CICS regions. Using a permanent SYSIN data set, you can use the CICS default system initialization table for all regions, and define those parameters that are common to clones in a SYSIN data set member.

### Defining the Shared CICS Data Sets

Shared data sets can be identified within the sysplex by means of the DSINDEX high-level qualifier, and a second-level qualifier that identifies the CICSplex, as shown in Figure 8. The SYSIN data set in this example also shows the use of a symbolic parameter, for the member name that contains the clone-specific system initialization parameters.

```
/* The system definition data set
//DFHCSD DD DISP=SHR,DSN=&INDEX1..CICSD###.DFHCSD
/* The journal partitioned data set
//DFHJPDS DD DISP=SHR,DSN=&INDEX1..CICSD###.DFHJPDS
/* The SYSIN partitioned data set
//SYSIN DD DISP=SHR,DSN=&INDEX1..CICSD###.SYSIN(CICS&CLONE.)
```

Figure 8. Shared CICS Data Sets

The data set naming conventions described here are an important factor when cloning CICS regions. See “Cloning CICS Regions” on page 134 for more information.

---

## Example of Using the Naming Convention

The target configuration shown in Figure 1 on page 18 provides an example of the naming convention applied to a CICSplex that runs in four of the six MVS images in the sysplex. In this example, we use the letter D to identify the Dallas CICSplex. The characteristics of the configuration are described under the next few headings.

## Six CPCs and Six MVS Images

Each CPC shown in the target configuration runs a single MVS image, with system identifiers MVSx where x is one of the letters A—F.

- MVSA, MVSB, MVSC, and MVSD are clones, with each MVS image running the same set of CICS regions.
- MVSE and MVSF are clone systems supporting the IMS TM subsystems.

## The Terminal-Owning Regions

All the terminal-owning regions are clones, with each terminal-owning region able to route transactions to any of 12 cloned application-owning regions.

Using the naming convention the TORs are named DTA1, DTB1, DTC1, and DTD1.

Although the terminal-owning regions do not have an affiliation with another subsystem in the same sense as the application-owning regions that are connected to IMS DBCTL and DB2, they are assigned the same group identifier as the application-owning regions that are in the same MVS image. This is useful when you are using automatic restart procedures to restart all the CICS regions on another MVS image.

## IMS DBCTL and DB2 Workloads

The total workload for the CICSplex shown in the target configuration ( Figure 1 on page 18) needs access to the following types of data:

- DL/I data in DBCTL-managed databases
- DB2 data in DB2-managed databases
- VSAM data in data sets accessed through SMSVSAM servers
- CICS shared temporary storage queues through a TS server.

### The Application-Owning Regions

The application-owning regions are allocated equally across the four MVS images MVSA, MVSB, MVSC, and MVSD, and all are identical clones. They are all capable of handling CICS-DL/I or CICS-DB2 transactions. Using the C T G I convention, their names all begin with DA (for Dallas application-owning regions), followed by a group character identifying them with their respective MVS image, and the iteration numbers 1, 2 and 3. Thus the SYSIDs are DAA1 through DAA3; DAB1 through DAB3; DAC1 through DAC3; and DAD1 through DAD3.

All these application-owning regions have an affiliation with the IMS DBCTL in their respective MVS images:

- The application-owning regions in MVSA are affiliated with the IMS DBCTL named DDA1.
- The application-owning regions in MVSB are affiliated with the IMS DBCTL named DDB1.

- The application-owning regions in MVSC are affiliated with the IMS DBCTL named DDC1.
- The application-owning regions in MVSD are affiliated with the IMS DBCTL named DDD1.

The application-owning regions also have an affiliation with the DB2 subsystems in their respective MVS images:

- The application-owning regions in MVSA are affiliated with the DB2 named DBA1.
- The application-owning regions in MVSB are affiliated with the DB2 named DBB1.
- The application-owning regions in MVSC are affiliated with the DB2 named DBC1.
- The application-owning regions in MVSD are affiliated with the DB2 named DBD1.

**Note:** CICS regions do not have the same close affiliation to an SMSVSAM, or temporary storage, server as they do with DB2 and DBCTL, because they do not connect to a specific named instance of these servers. CICS connects automatically to the SMAVSAM or TS server that is active in its MVS image at the time of CICS initialization. In the event of any kind of failure, it doesn't matter to which SMSVSAM CICS reconnects from the point of view of recovery and transaction backout. Backout considerations do not apply shared TS pool queues managed by a TS server, because such queues are not recoverable.

### **Connections**

For information about how to define the links for this configuration, using the minimum number of CONNECTION and SESSIONS definitions, see "Chapter 14. Planning the Resource Definitions" on page 121.



---

## Chapter 4. Planning the Terminal-Owning Regions

In “Chapter 2. Planning the Subsystem Configuration for a Sysplex” on page 17, we proposed a target configuration for a CICSplex in a sysplex environment. This chapter discusses in more detail the planning considerations for the terminal-owning regions, covering the following topics:

- Transaction routing, and changing from static to dynamic transaction routing
- Splitting combined terminal-owning/application-owning regions into separate terminal-owning regions and application-owning regions
- Planning migration to dynamic transaction routing, including the detection of inter-transaction affinities and considerations for a dynamic transaction routing program
- Implementing VTAM generic resources for multiple terminal-owning regions
- Creating multiple terminal-owning regions
- Implementing VTAM persistent LU-LU sessions.

---

### Transaction Routing

The CICS transaction routing facility allows terminals or logical units connected to one CICS region to initiate, and communicate with, transactions in another CICS region. This means that you can distribute terminals and transactions around your CICS regions and still have the ability to run any transaction from any terminal.

Figure 9 shows a terminal connected to one CICS region running a user transaction in another. Communication between the terminal attached to the terminal-owning region and the user transaction is handled by a CICS-supplied transaction called the relay transaction.

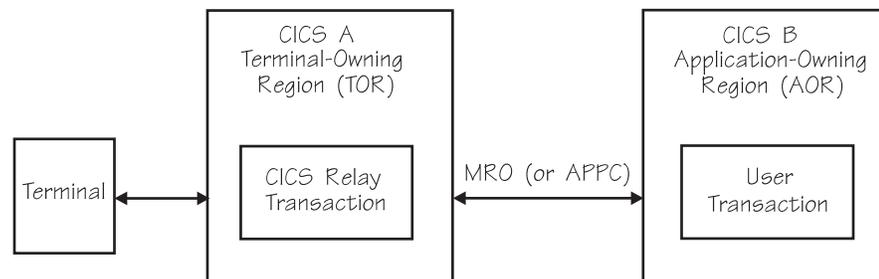


Figure 9. The CICS Transaction Routing Facility

### Static and Dynamic Transaction Routing

When you define transactions to CICS, you can define them as local or remote transactions. Local transactions always run in the terminal-owning region; that is, in the CICS region to which the terminal initiating the transaction is directly logged on. Remote transactions are routed to another CICS region connected to the terminal-owning region by MRO links (or routed to other CICS regions that are connected by CICS intersystem communication (ISC) links).

CICS supports two forms of transaction routing for remote transactions:

## 1. Static transaction routing

The transaction resource definition is predefined with the name of a remote CICS region to which it is to be routed for execution. Transactions defined in this way are always routed to the specified remote CICS region. With this method, you can change the remote destination of the specified transaction only by manually altering and reinstalling the transaction resource definition.

## 2. Dynamic transaction routing

You do not need a transaction resource definition in the terminal-owning region to specify transaction routing (unless the transaction is defined to be invoked by means of a PF or PA key—see note). When a transaction is invoked at a user's terminal, CICS searches the table of installed transaction definitions for the name of the transaction. If CICS does not find an entry in the table, CICS automatically calls the dynamic transaction routing program to determine whether it should be routed to a remote CICS region. This is the recommended method for using dynamic transaction routing.

**Note:** If you have applications in which terminal users invoke transactions by special task request functions, as defined by the TASKREQ attribute of the transaction resource definition, you must define specific transaction resource definitions for them. The elimination of the need for transaction definitions applies only to those transactions that are invoked by a traid.

Transactions attached in the terminal-owning region without their own unique transaction definition are exempt from transaction attach security checks in the terminal-owning region (unless they are run locally in the terminal-owning region). See “Chapter 15. Planning CICSplex Security” on page 137 for details.

Alternatively, you can use transaction resource definitions for each individual transaction, specify the DYNAMIC(YES) attribute. For transactions defined with DYNAMIC(YES), CICS invokes the dynamic transaction routing program to determine where to route the transaction.

For more information, see “Defining Remote Attributes for Transaction Routing” on page 121.

You can specify both a remote system name and dynamic transaction routing. In this case, CICS still invokes the dynamic transaction routing program, allowing the defined remote system name to be dynamically changed.

Using dynamic transaction routing across a CICSplex, compared with static routing, can improve performance and improve availability for terminal end users. However, in the section that follows you will see that we recommend you use static transaction routing before migrating to the dynamic method. Planning and implementing dynamic transaction routing is discussed further in 53.

---

## Splitting CICS into Separate Terminal- and Application-Owning Regions

If you are operating with combined terminal-owning/application-owning regions (TOR/AORs) you are recommended to split them into separate terminal- and application-owning regions. For example, if your combined CICS regions are organized to support different applications as shown in Figure 10 on page 51, you should reconfigure them as illustrated.

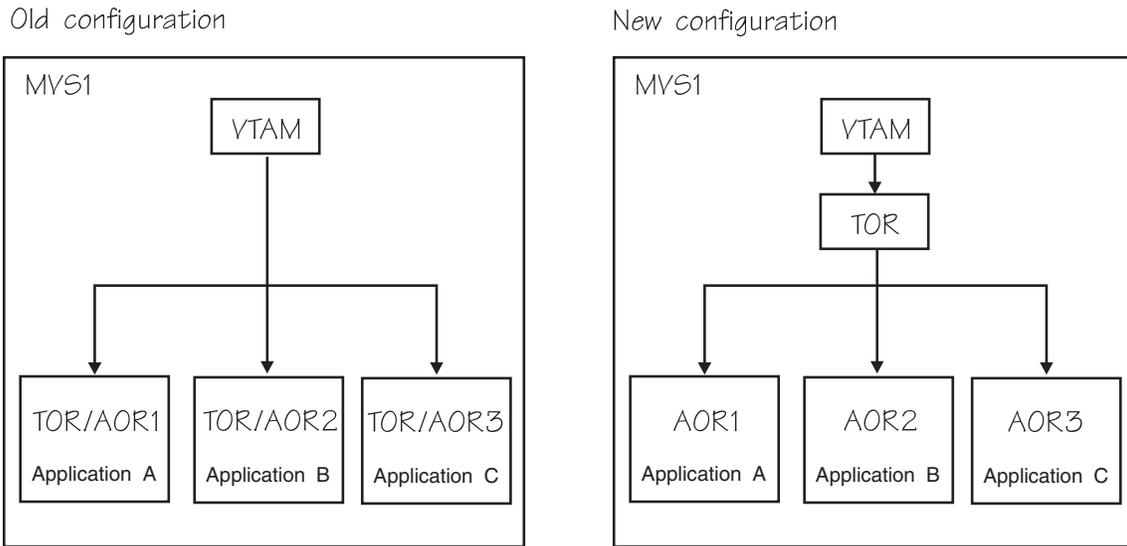


Figure 10. CICS Region Configurations Before and After Splitting into Separate Resource Manager Regions for Terminals and Applications

Splitting the existing regions into separate terminal- and application-owning regions is a first step, using static routing definitions to route the transactions to the appropriate application-owning region. Initially, there is no need to clone these regions—each application-owning region should be capable of handling its application workload as well as, and probably better than, before when they were combined with the terminal-owning regions. Test the new terminal-owning region—AOR configuration with static transaction routing, and later switch to using a dynamic transaction routing program. Finally, you can replicate the single terminal-owning region to create as many as you need to provide continuous availability.

**Alternative Using Multiple Terminal-Owning Regions:** If your terminal users are already using different APPLIDs, an alternative approach is to create multiple terminal-owning regions to support the application-owning regions you create, creating one terminal-owning region for each application-owning region (see Figure 11 on page 52).

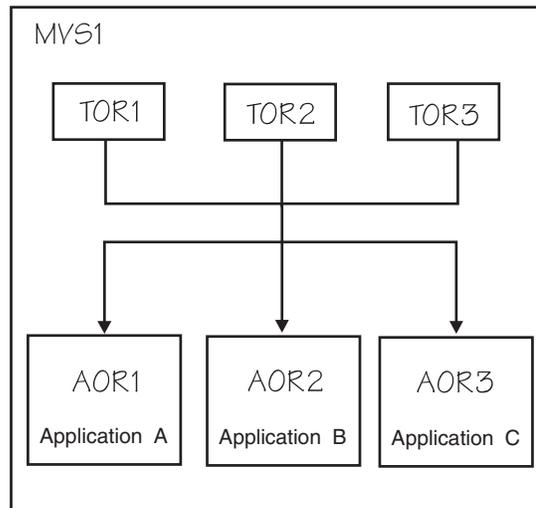


Figure 11. Alternative Initial Configuration Using Multiple Terminal-Owning Regions

In summary, the tasks in this stage are as follows:

**1. Create a new region for terminal control only**

Set up a new CICS terminal-owning region with a new VTAM APPL name, with all the unique CICS system data sets required for a new CICS region, but sharing the old CICS system definition (CSD) data set with the application-owning regions.

**2. Remove the old APPLIDs**

It is not absolutely necessary to have VTAM APPL names for the application-owning regions and you can remove these from your VTAM definitions if you so choose. Terminal end users should be enabled to log on only to the terminal-owning region. However, you may choose to allow some special users (system programmers, for example) to log on to the application-owning regions directly for problem determination purposes.

**3. Modify VTAM front-end LOGON panels**

Modify the VTAM front-end logon panel for the CICS applications, replacing the old combined region options with the terminal-owning region only. This could involve changes to Netview Access Services panels, or VTAM unformatted system services (USS) tables, or SNA Application Monitor (SAMON), or to whatever method you currently use to present application selections to network users.

Alternatively, use multiple terminal-owning regions where the APPLIDs are the same as those for the old combined regions.

**4. Set up transaction routing definitions**

Alter your existing transaction definitions so that the terminal-owning region knows which application-owning region to route them to. Use the same definition in both the terminal-owning region and application-owning region. For example, the following ALTER command adds the minimum remote system attribute necessary for static transaction routing for transaction id AC20 in group DFH\$CTXT:

```
ALTER TRANSACTION(AC20) GROUP(DFH$CTXT) REMOTESYSTEM(AOR1)
```

You can enter this command online using the CEDA transaction, or in a batch job using the CICS system definition utility program, DFHCSDUP. This transaction definition works as both a remote definition for the terminal-owning region, and a local definition for the application-owning region.

#### 5. Define the MRO links

Define the CONNECTION/SESSIONS pairs for the terminal-owning region with the three application-owning regions.

#### 6. Test the configurations of separate terminal- and application-owning regions

When you've completed the resource definitions, bring up all four regions and test that:

1. The MRO connections work
2. The transactions work in the transaction routing environment.

**Note:** Before going into production with separate terminal-owning regions and application-owning regions, you should review how your resource definitions are grouped. For example, you should avoid installing resource definitions in a terminal-owning region that are unnecessary when the transactions that use them are being routed to an application-owning region.

---

## Planning Migration to Dynamic Transaction Routing

After the splitting of the combined terminal- and application-owning regions and the subsequent testing, the next step is to switch from static to dynamic transaction routing. This is necessary to exploit workload balancing when you clone the application-owning regions, to enable you to spread the CICS application workload over more than one application-owning region

There are two main tasks you have to consider when planning to use dynamic transaction routing. These are:

- **The suitability of the transactions within each application for dynamic transaction routing.** You must determine whether any of the transactions are subject to constraints that could inhibit them from being routed dynamically. One factor that can constrain an otherwise free choice of application-owning region is the use of particular CICS programming techniques that transactions use to pass data one to another. See "Transaction Affinities" for information about the kinds of affinity that can constrain dynamic transaction routing.
- **The provision of a dynamic transaction routing program.** You can either use the dynamic transaction routing program provided by CICSplex SM, or write your own, in which case you must ensure it has the necessary logic to:
  - Determine which remote systems are capable of processing the incoming transactions
  - Manage transaction affinities, where they exist
  - Do workload balancing.

See "Planning a Dynamic Transaction Routing Program" on page 56 for information about the requirements for a dynamic transaction routing program.

## Transaction Affinities

There are many different CICS application programming techniques that you can use to enable CICS transactions to pass data from one to another. Some of these techniques require that the transactions exchanging data must run in the same

CICS region, which imposes restrictions on the dynamic routing of transactions. If transactions exchange data in ways that impose such restrictions, there is said to be an affinity between them.

Basically, there are two types of affinity:

- Inter-transaction affinity
- Transaction-system affinity

### **Inter-transaction Affinity**

Two or more CICS transactions are said to have affinity when they pass information between them, or synchronize their activities, using techniques that require the transactions to execute in the same CICS region. This type of affinity is inter-transaction affinity, where a set of transactions share a common resource and/or coordinate their processing. Inter-transaction affinity, which imposes restrictions on the dynamic routing of transactions, can occur in the following circumstances:

- One transaction terminates, leaving “state data” in a place that a second transaction can access only by running in the same CICS region as the first transaction.
- One transaction creates data that a second transaction accesses while the first transaction is still running. For this to work safely, the first transaction usually waits on some event, which the second transaction posts when it has read the data created by the first transaction. This synchronization technique requires that both transactions are routed to the same CICS region.

### **Transaction-System Affinity**

There is another type of transaction affinity that is not an affinity between transactions themselves. It is an affinity between a transaction and a particular CICS region, where the transaction interrogates or changes the properties of that CICS region—this is called transaction-system affinity.

Transactions with affinity to a particular system, rather than another transaction, are not eligible for dynamic transaction routing. In general, they are transactions that use INQUIRE and SET commands, or have some dependency on global user exit programs, which also have an affinity with a particular CICS region.

For a detailed discussion of transaction affinities, see the *CICS Application Programming Guide*.

## **Detecting Inter-transaction Affinities**

To help you detect instances of inter-transaction affinity, CICS supplies an affinity utility, which is available as a program offering for use with the following releases of CICS:

- CICS for MVS/ESA Version 4 Release 1
- CICS/ESA Version 3 Release 3
- CICS/ESA Version 3 Release 2.1
- CICS/MVS Version 2 Release 1.2

For all these releases of CICS, the utility is supplied as a separate program product.

The affinity utility is supplied as part of the CICS base in the CICS Transaction Server for OS/390.

The affinity utility is available on the earlier releases so that users of these releases can plan how to remove, or how to manage, the affinities before migrating to the workload management environment provided by CICS and MVS.

The CICS affinity utility is designed to help you plan your migration to dynamic transaction routing by detecting potential causes of inter-transaction affinity. You can use it to detect programs containing EXEC CICS commands that may cause transaction affinity.

The affinity utility consists of the following components:

1. **The affinity load module scanner**

This is a batch utility that scans a load module library to detect programs in the library that issue EXEC CICS commands that may cause transaction affinity.

Although there is no interface between this component and the others in the affinity utility, you can use the report to verify, and modify where necessary, the basic affinity groups generated by the affinity reporter. This is indicated by the broken line between the scanner report and the basic affinity transaction groups shown in Figure 12 on page 56.

2. **The affinity detector**

This detects transaction affinities in an operating CICS region (that is, in real time) by intercepting the EXEC CICS commands that cause transaction affinity, and storing details of the affinities in a data space. It consists of two CICS-supplied transactions and some global user exit programs.

3. **The affinity reporter**

This is a batch utility that takes the affinity data collected by the affinity detector, and generates a report for use by system programmers, and sets of basic affinity transaction groups. A basic affinity transaction group is a set of transaction identifiers that share a particular resource that creates an inter-transaction affinity.

4. **The affinity group builder**

This is a batch utility that takes the basic affinity transaction groups produced by the affinity reporter, and combines them to produce **affinity transaction groups** in a form suitable for batch input to CICSplex SM. An affinity transaction group is a set of transaction identifiers that have an inter-transaction affinity with one another. CICSplex SM uses the transaction groups to construct affinity tables for use by its dynamic transaction routing program.

The affinity utility components are illustrated in Figure 12 on page 56.

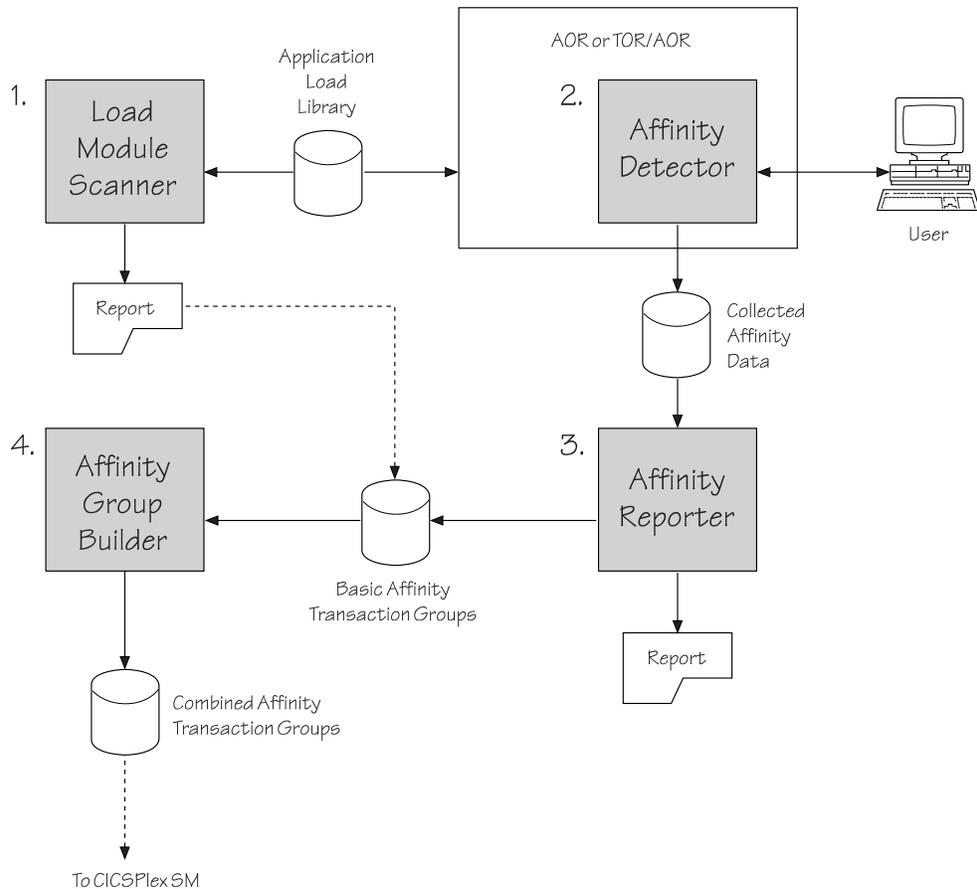


Figure 12. The CICS Affinity Utility Components

The affinity utility detector determines the affinities that apply to a single CICS region; that is, a single application-owning region or a single terminal/application-owning combined region. It can be run against production CICS regions, and is also useful in a test environment, to detect possible affinities introduced by new or changed application suites or packages.

**Notes:**

1. The affinity utility cannot guarantee to find every example of inter-transaction affinity in a given CICS region. It is intended primarily as a migration aid that will help you to detect affinities. To ensure that you detect as many potential affinities as possible, you should use the affinity utility against all parts of your workload, including rarely used transactions and unusual situations.

## Planning a Dynamic Transaction Routing Program

This section covers the main points that you must consider when developing your dynamic transaction routing environment. These are:

- Defining transaction routing tables
- Identifying inter-transaction affinities
- Maintaining the status of application-owning regions
- Monitoring the performance of application-owning regions

### Defining Transaction Routing Tables

Transaction routing tables specify which application-owning regions are capable of processing the transactions.

Your dynamic transaction routing program must be able to identify all the candidate application-owning regions for each transaction that is passed to it for routing, and then select the best one to meet the transaction's service-level agreement (SLA) objectives.

At the simplest level you must provide some way of mapping the transaction identifiers (tranids) to the available application-owning regions. You can achieve this directly by building tables of tranids with supporting SYSIDs. An indirect method is to group collections of transactions into defined workloads, and then map the workload names to the application-owning regions.

The tables of workload information can be held on DASD and loaded ready for use at system initialization.

### **Identifying Inter-transaction Affinities**

Inter-transaction affinities constrain the choice of application-owning region when making the routing decision.

If any of your transactions have affinities with one another, your dynamic transaction routing program must be capable of managing this, either through information built into the routing tables, or in some other way.

### **Maintaining the Status of Application-Owning Regions**

If your dynamic transaction routing program is to maintain effective workload balancing, it must know the status of the various application-owning regions. It must also know whether they are available for transaction routing.

### **Monitoring the Performance of Application-Owning Regions**

You must provide some facility for continually monitoring the performance of the available application-owning regions. Relative performance data for the available application-owning regions is needed to enable the dynamic transaction routing program to select the remote region that is best able to meet a transaction's SLA objectives.

## **Using CICSplex SM**

As an alternative to writing your own dynamic transaction routing program and all the other functions that you need to support it, you can choose to use the workload management services of CICSplex SM or another workload management product that provides equivalent function.

CICSplex SM provides the facilities you need for defining and controlling CICS workloads.

You can use CICSplex SM to control dynamic transaction routing, where the routing can be based on:

- User, terminal, and affinity attributes associated with the transactions
- The transaction identifiers.

In addition to all the support functions needed for dynamic transaction routing, CICSplex SM provides a dynamic transaction routing program that includes:

- The ability to recognize and handle inter-transaction affinities.
- Workload balancing based on either a "join-shortest-queue" algorithm or a "goal-oriented" algorithm.
- Workload separation, based on the requirements of users, terminals or the transactions.

- An ABEND compensation facility, whereby CICSplex SM attempts to avoid application-owning regions that have a not-insignificant probability of abending the transaction—that is, where similar transactions have terminated abnormally in the recent past.

If necessary you can add more function by customizing the CICSplex SM dynamic transaction routing program.

---

## Planning for VTAM Generic Resources

When you have created a single terminal-owning region to handle the flow of work to the separate application-owning regions, as described under “Splitting CICS into Separate Terminal- and Application-Owning Regions” on page 50, you need only one VTAM APPLID for all the applications supported by the application-owning regions. To ensure your CICSplex can provide continuous availability, the next step is to create multiple terminal-owning regions, all with links to all the application-owning regions. However, to enable these terminal-owning regions to appear as if they are a single entity to the network of CICS users, you should use the VTAM generic resources function.

Establishing the VTAM generic resource function for your CICSplex involves the following steps:

- Defining the coupling facility structure
- Defining security authorizations
- Setting trace options
- Defining the generic resource name to CICS
- Migrating to VTAM generic resources.

**Note:** The VTAM generic resources function and the CICS extended recovery facility (XRF) are mutually exclusive.

## Defining the Coupling Facility Structure

You must set up in the coupling facility policy the required generic resource structure, ISTGENERIC, using a coupling facility resource management (CFRM) policy. Plan the size of the generic resource structure according to the number of CICS specific APPLIDs it has to support, following the instructions given in the VTAM library. Also see *z/OS MVS Setting Up a Sysplex*.

## Defining Security Authorizations

If you are operating with an external security manager, such as RACF, authorize all the CICS terminal-owning regions to register with the generic resource name you plan to use. To authorize CICS terminal-owning regions to access a VTAM generic resource, you must define a VTAMAPPL profile with the generic resource name as the VTAMAPPL profile name. Authorize each CICS terminal-owning region with READ access to the VTAMAPPL profile.

For example, if the generic resource name **CICSD###** is used by all the terminal-owning regions shown in Figure 1 on page 18, which have run under the CICS region userids of CICSDTA1, CICSDTB1, CICSBTC1, and CICSDTD1, you can define the profile and authorizations as follows:

```
RDEFINE VTAMAPPL CICSD### UACC(NONE)

PERMIT CICSD### CLASS(VTAMAPPL) ID(CICSDTA1, CICSDTB1, CICSBTC1, CICSDTD1)
ACCESS(READ)
```

## Setting Trace Options

Review your VTAM trace options, and consider whether you also want to trace coupling facility events. See the *VTAM Network Implementation Guide*, SC31-6494, and the *VTAM Resource Definition Reference*, SC31-6498 for information about using VTAM trace.

## Defining the Generic Resource Name to CICS

Define the generic resource name (CICSD### in our example) using the GRNAME system initialization parameter, to all the *terminal-owning regions* that are members of that generic resources name. This enables a CICS terminal-owning region to register its APPLID as a member of the VTAM generic resource specified on the GRNAME system initialization parameter.

When any of the terminal-owning regions are next started, the terminal users can log on either by the generic resource name or by the specific APPLID. However, LU6.2 devices *must* log on using the generic resource name.

Do *not* specify this parameter to any other CICS regions, such as application-owning regions, file-owning regions, and so on. You must also ensure that the XRF parameter specifies XRF=NO.

## Migrating to VTAM Generic Resources

When planning your migration to VTAM generic resources, you should bear in mind the following rules governing CICS use of the VTAM generic resources function:

- Generic resource names must be unique in the network
- A generic resource name cannot be the same as an APPLID in the network
- A CICS region can have only one generic resource name and only one APPLID

Also, you must have an APPN environment within the sysplex (see “VTAM Generic Resources” on page 13).

There are some restrictions on the use of generic resources by certain types of devices:

- Devices using message protection cannot logon using the generic resource name. They must use the APPLID and therefore cannot take advantage of session balancing
- LU6 connections *must* logon using the generic resource name. They *cannot* logon using an APPLID if the APPLID is a member of a generic resource name.
- If an LU6.2 connection is bound at synclevel 2 to a specific member of a generic resource name, it is reconnected to that specific APPLID every time it is re-bound (the VTAM generic resources function ensures that this requirement is met). If, for some reason, the specific APPLID is not available, connection to the generic resource as a whole is denied.
- If an LU6.1 connection is bound to a specific member of a generic resource, it is reconnected to that specific APPLID every time it is re-bound (the VTAM generic resources function ensures that this requirement is met). If for some reason the APPLID is not available, connection to the generic resource as a whole is denied.

In addition, certain configurations are prohibited by the following restrictions:

- You cannot route transactions from a member of a generic resource across an ISC connection.

- A remote LU6 partner cannot be accessed from more than one member of a generic resource name.
- You cannot use ISC to connect members of the same generic resource name. If a region (for example, an AOR) must connect to more than one member of a generic resource, it must connect to them using MRO: it cannot use ISC.

Note that, in this context, a terminal-owning region is any CICS region that owns terminals and is a candidate to be a member of the generic resource name. Thus a combined TOR/AOR is considered to be a terminal-owning region for the purposes of this discussion

If you have no LU6 connections to your terminal-owning region you could choose a new name for the generic resources and retain your old APPLID. Non-LU6 terminals can logon by either APPLID or generic resource name, hence they would not be affected by the introduction of the generic resource name. You could then gradually migrate the terminals to using the generic resource name.

However, if you have LU6 terminals in your network you will probably want to migrate to generic resources without requiring all your LU6 network partners to change their logon procedures. A solution to this is to use the APPLID of your existing terminal-owning region as the new generic resource name. Since this requires you to choose a new APPLID, it is also necessary to change the CONNECTION definitions of MRO-connected application-owning regions and RACF profiles that specify the old APPLID. Note, however, that you do not need to change the APPL profile to which the users are authorized—CICS passes the GRNAME to RACF as the APPL name during signon validation, and the old APPLID is now the GRNAME.

The recommended migration steps are:-

- Configure your CICSplex with a single terminal-owning region
- Set the generic resource name to be the current APPLID of that terminal-owning region
- Change the current APPLID to a new value (consider using the recommended naming convention)
- Change CONNECTION definitions in MRO partners to use the new APPLID for the terminal-owning region (again, this is a suitable time to change to an appropriate naming convention).
- Change RACF profiles that specify the old APPLID.
- Restart the CICSplex.

At this point:

- Non-LU6 terminals can logon using the old name (without being aware that they are now using VTAM generic resources). They will, of course, be connected to the same TOR as before because there is only one in the generic resources set
- LU6 connections logon using the old name (thereby conforming to the rule that they must connect by generic resource name)
- Devices using message protection must change to use the new APPLID before the existing terminal-owning region is cloned. Up to that point they are rebound to the only TOR.
- Install new cloned terminal-owning regions with the same generic resource name and the same connectivity to the set of AORs
- Autoinstalled non-LU6 terminals now start to exploit session balancing

- Autoinstalled LU6.2 synclevel 1 terminals start to exploit session balancing
- Existing LU6.1 and LU6.2 synclevel 2 terminals continue to be connected to the original terminal-owning region (by generic resource name)
- Special considerations apply to non-autoinstalled terminals and LU6 connections used for outbound requests.

### **Special Considerations for Non-Autoinstalled Terminals and Connections**

If an LU is predefined to a specific terminal-owning region, and the LU initiates the connection, the generic resources function cannot be allowed to choose any terminal-owning region in the generic resources. The connection must be made to the terminal-owning region that has the definition. This requirement means that you must install the VTAM generic resources resolution exit program, ISTEEXCGR, to enforce selection of the correct APPLID (for the terminal-owning region).

Note that this is not necessary if the connection is always initiated by the terminal-owning region.

### **Special Considerations for Outbound LU6 Connections**

As already stated, a remote LU6 partner cannot be accessed from more than one member of a generic resources.

A problem may arise when the LU6 partner is to be used as the target for function shipping or distributed transaction processing (DTP) requests from a terminal-owning region.

There is no problem if application-owning regions function-ship, or use DTP, or even transaction route to a remote LU. The restriction does not apply because the application-owning regions are not members of a generic resource. However, if a terminal-owning region in a generic resource needs to function ship or participate in DTP to a remote LU, the restriction implies that no other terminal-owning region in the generic resources can access the remote LU directly. The CICSplex must be configured so that all access to the remote LU is via that terminal-owning region. Thus other CICS regions in the CICSplex would need to daisy-chain their requests to that terminal-owning region via MRO links.

In addition, as previously explained, if it is possible that the remote LU initiates the connection, it is necessary to ensure that the connection is made to the correct terminal-owning region. Since the remote LU must use the generic resource name, you must provide a VTAM generic resources resolution exit program to select the APPLID of the correct terminal-owning region for that remote LU.

One option is to choose one terminal-owning region to act as a **network hub** for connections to all LU6 partners that are targets of outbound requests. This hub owns all such connections, which are almost certainly predefined, because they are referenced by existing applications or resource definitions in the CICSplex. All applications running in application-owning regions or other terminal-owning regions must daisy-chain their requests for services from the remote LUs through the hub.

The network hub can be a member of the generic resource name, in which case it is necessary to install a VTAM generic resources resolution exit program to direct incoming binds from the LU6 partners to the network hub terminal-owning region.

A simpler option is to have a network hub that is not a member of the generic resource name. This avoids the need for the VTAM generic resources resolution exit program, but requires that all the predefined LU6 partners that may initiate

connections to the CICSplex logon using the APPLID of the network hub terminal-owning region. This is the recommended option, unless it is not possible to change the logon name used by existing LU6 partners.

---

## Implementing VTAM Persistent LU-LU Sessions

To enable this support in each terminal-owning region, specify a time interval on the CICS system initialization parameter, PSDINT, and ensure that the XRF parameter specifies XRF=NO. You must also review the RECOVOPTION and RECOVNOTIFY attributes on your TYPETERM definitions, and set appropriate options for the actions you want CICS to take when recovering sessions. For example, you might want to run automatically your sign-on transaction to prompt users to sign on again.

**Note:** Before implementing VTAM persistent session support in your CICS terminal-owning regions, you should weigh up the relative merits of both persistent sessions and VTAM generic resources.

With multiple terminal-owning regions in your CICSplex, you may consider that VTAM generic resources are essential in order to represent the CICSplex as a single entity to the terminal network. Also, in the event of a terminal-owning region failing, users of a failed terminal-owning region can quickly logon again to the generic resources name, and be connected to another terminal-owning region.

On the other hand, if you implement persistent sessions, VTAM retains sessions either until the failed terminal-owning region is restarted and the session recovered, or the timeout interval expires, or the user breaks the session. This avoids users having to logon, although they do have to sign on again and restart any transactions that were in-flight at the time of the failure.

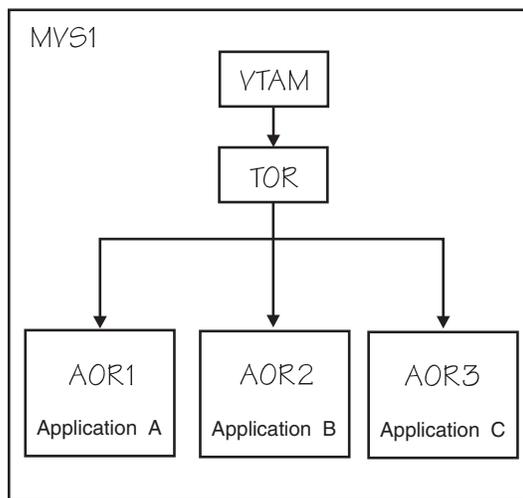
If you are using both VTAM generic resources and VTAM persistent sessions and a terminal-owning region fails, VTAM retains the terminal sessions in a recovery pending state until the TOR is restarted and recovers its sessions. During this time, the user perception is that CICS is “hanging”, and whatever is on the screen at the time of failure remains until persistent sessions recovery is complete. In some circumstances users may find it quicker to break the session and logon to another terminal-owning region using the generic resources function.

---

## Cloning the CICS Terminal-Owning Regions

In the first stage of splitting the combined TOR/AORs, as shown in Figure 10 on page 51, we recommended creating a single terminal-owning region with connections to each of the application-owning regions. The next stage is to create multiple terminal-owning regions, as shown in Figure 13 on page 63.

Single TOR configuration



Multiple TOR configuration

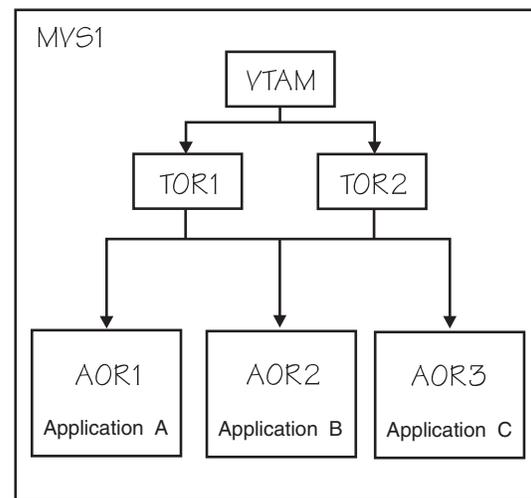


Figure 13. CICS Region Configurations Before and After Splitting into Separate Terminal-Owning and Application-Owning Regions

For the purposes of illustration, we have shown both terminal-owning regions residing in the same MVS image, but clearly in a sysplex environment it makes more sense to start each terminal-owning region in a separate MVS. Initially you can test TOR2 in the same MVS as TOR1. Later you can migrate it to another MVS, ideally when cloning the application-owning regions across the sysplex on other MVS images. The steps are as follows:

#### 1. Create another terminal-owning region

Set up a new CICS terminal-owning region with a new VTAM APPL name. This region should have all the unique CICS system data sets required for a new CICS region, but should share the same CSD as the other terminal-owning region and the application-owning regions.

**Note:** If you have predefined terminals, you may need to modify your CSD groups to ensure they are installed in the appropriate region. Predefined terminal definitions are not necessarily capable of being shared, and may have to be installed in one terminal-owning region only. For example, you must plan for predefined terminal that are defined with AUTOCONNECT(YES), because these must be installed in one region only.

Also, the autoinstall user-replaceable module in each terminal-owning region must be able to generate termids that are unique across the CICSplex.

#### 2. Use the same startup group list as the first terminal-owning region

The second and subsequent terminal-owning regions, to be true clones, must be virtually identical, and have equal access to the application-owning regions. This means they must have the same CONNECTION and SESSIONS definitions. They must also have the transaction-routing capability. The only

attributes of the clone terminal-owning regions that are different are their identifiers: APPLIDs, SYSIDNTs, and MNSUBSYS.

See “Chapter 14. Planning the Resource Definitions” on page 121 for details of how CSD groups should be defined and named for common use.

**3. Enable the VTAM generic resources function**

Start VTAM and ensure the generic resource is the only CICS VTAM application available for all users of Application A, Application B, and Application C.

**4. Test the multiple terminal-owning region configuration**

When you’ve completed the resource definitions, bring up all five regions and test that:

1. The MRO connections work
2. The transactions work with dynamic transaction routing.

---

## Chapter 5. Planning the Application-Owning Regions

This chapter discusses some of the factors that you should take into account when planning the application-owning regions. You will need to be clear about your prime objectives. For most users, these will include:

- Achieving the optimum level of central processor (CP) utilization for each CPC
- Achieving the required throughput in terms of the number of transactions per hour, per shift, or per day
- Achieving the required level of availability
- Special considerations for START commands in a dynamic transaction routing environment.

---

### Achieving the Optimum Level of Processor Utilization

There are two main factors you need to consider when planning to optimize the use of central processor resources.

1. The effect of workload balancing on capacity planning
2. The limited way in which a single CICS region exploits the multiprocessor capacity of an n-way CPC.

### The Effect of Workload Balancing on Capacity Planning

Without the benefits of dynamic transaction routing and workload management facilities, you would probably have to aim for a much lower level of CP utilization, to ensure that you had sufficient capacity to allow for peaks in workload.

In a sysplex environment, using dynamic transaction routing with full workload balancing support, you don't need to restrict workload to a particular level of utilization on any given CPC in order to allow for peak loads. With the benefit of workload balancing, any terminal-owning region within the sysplex can route work away from busy application-owning regions to those that are best able to handle the work.

This means that you should be able to run each CPC in a System/390 microprocessor cluster at high levels of utilization. However, you should include some additional capacity to provide cover in the event of a hardware failure. This reflects one of the strengths of the sysplex—with cloned systems, you would need the capacity of only one system to act as standby for all the others in the sysplex.

### CICS and Multiprocessor Capacity of an n-way CPC

In general, CICS runs user application programs under a single TCB—known as the CICS **quasi-reentrant TCB**. (CICS also uses other TCBs; for example, it uses the resource-owning TCB, for opening and closing data sets and for loading programs. However, the average time spent on these activities in an application-owning region for each transaction is very small and therefore can be discounted.)

For a single CICS region you can choose to exploit the multiprocessor capacity of a CPC by subtasking its VSAM activity. This is done by using the (optional) CICS concurrent TCB, available when you specify the SUBTSKS system initialization parameter. This means that a single CICS region running transactions that issue VSAM file requests can utilize up to two CPs concurrently. The degree of

concurrent processing depends on the proportion of VSAM activity in relation to the rest of the transaction, but it typically means that a single CICS region could “drive” up to one-and-a-half CPs.

See the *CICS Performance Guide* for details of CICS subtasking using the concurrent TCB.

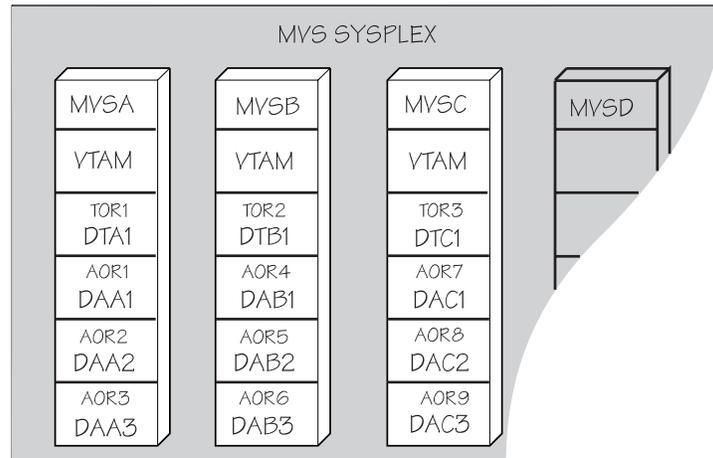


Figure 14. A Partial View of the CICSplex Configuration Showing some of the Application-Owning Regions

### Allowing for Other Work

For ease of illustration in the various examples given in this book, the target configuration that is shown in Figure 1 on page 18 shows only 3 explicitly named application-owning regions in each of the MVS images assigned for CICS transactions. To achieve full utilization of the CPCs in our System/390 microprocessor cluster we might have to allocate more application-owning regions (shown by DAA $n$  through DAD $n$  in the configuration diagram) but this would depend on the nature of the transactions. For example, when planning this aspect of the migration you should take into account the work done by:

- The VTAM address space
- Workload balancing and transaction routing from the terminal-owning regions to the application-owning regions
- Processing the DL/I, DB2, and VSAM RLS requests issued by CICS transactions in the sysplex data sharing environment
- Processing any CICS file control requests that are function-shipped to a file-owning region (if there is one)
- Processing any temporary storage and transient data requests that are passed to a TS server or function-shipped to a queue-owning region (if there is one)
- Any additional activity in the CPC, such as that required for any IMS batch regions (possibly converted from CICS shared database regions).

It is not necessary to calculate precisely the number of CPs that the application-owning regions could drive. It is sufficient to derive a rough estimate and err on the high side when calculating the number of application-owning regions required to fully utilize the number of CPs in the CPC. There is no significant performance impact from over-configuring the number of AOR.s.

## The Ratio of AORs to TORs

Assuming that a terminal-owning region performs only transaction routing functions, which are only a fraction of the work done by an application-owning region, one terminal-owning region should be able to serve between 10 and 20 application-owning regions. For example, if the dynamic transaction routing path length in a terminal-owning region is about 60 000 instructions, and the average size of your transactions is about 1 million instructions, a single terminal-owning region could drive about 16 application-owning regions. However, to assign only one terminal-owning region in our CICSplex would create a single point of failure, which must be avoided to ensure high availability to end users.

As a general rule, we recommend that you install one terminal-owning region in each MVS image, and as many application-owning regions as needed to fully utilize all the CPs available to that MVS image. For convenience and ease of illustration, we have shown only 3 explicitly named application-owning regions in our target configuration in Figure 1 on page 18.

---

## Achieving the Required Transaction Throughput

To determine whether you have enough application-owning regions to meet service level agreements, you need to estimate the capacity of the planned application-owning region configuration needed for the anticipated workload. If the number of application-owning regions is too low to handle the anticipated workload, you will need to allocate more. If you are unable to allocate more application-owning regions in the existing MVS images, because the CPCs are already at their optimum processor utilization level, you will need more MVS images to support the additional application-owning regions.

## Estimating the Throughput Rate

For the purpose of this discussion, let us assume that:

- Each individual CPC running our target CICSplex configuration can process work at the rate of 50 million instructions per second (50 mips).
- The optimum utilization aimed for is almost 100%, and half of this is for the three named application-owning regions.
- The average transaction path length is about 1 million instructions in the application region.

These assumptions would indicate that the 3 named application-owning regions we have illustrated in each MVS in our target configuration can process a maximum of 25 transactions per second, based on the following calculation:

$$((\text{MIPS} \div 2) \div \text{path length (in millions)}) = ((50 \div 2) \div 1) = 25$$

In a CICSplex in the sysplex environment it should be possible to drive the 3 named application-owning regions at this rate more or less continuously. See Table 3 for the DL/I transaction throughput across the CICSplex using the DL/I application-owning regions.

*Table 3. Potential Throughput at 25 Transactions per Second per CPC*

<b>Number of Application-Owning Regions</b>	<b>Transactions In One Hour</b>	<b>Transactions In Twelve Hours</b>
Three (in one MVS image)	90 000	1 080 000
Nine (in 3 MVS images)	270 000	3 240 000

If the anticipated number of transactions is greater or less than the numbers shown in Table 3 on page 67 we would have to change the number of application-owning regions accordingly. Also, if the peak load is ever greater than 270 000 per hour, we would also have either to increase the resources to match, or to allow response times to rise.

---

## Achieving the Required Level of Availability

When you have decided on the number of application-owning regions you need, how should these be allocated across the sysplex? Planning the allocation of the application-owning regions across the sysplex is essential to ensure the right level of availability. Before you can decide how to allocate the regions, however, you have to decide the extent to which you are going to clone them, and which regions should be cloned.

Clones are regions that are identical in every respect except for identifiers such as SYSIDNTs, and APPLIDs; therefore each application-owning region is capable of running the same transaction workload.

Should all the application-owning regions be the same, or should they be grouped in cloned subsets? You should consider the following when planning your cloning strategy:

- Cloning means creating a number of identical CICS regions, and in this case a number of identical application-owning regions to support workload balancing across the CICSplex.
- Creating multiple application-owning regions can increase CICS availability by reducing or minimizing the impact of a CICS failure. If one region fails, users whose transactions are running in another region are unaffected.
- Availability is further increased if more than one region is capable of running the same transactions.
- An increase in throughput can be achieved by cloning application-owning regions, and spreading the workload across the cloned application-owning regions, which themselves can be spread across the sysplex.
- Cloning application-owning regions in different MVS images protects your applications against an MVS failure. If an MVS image fails, clones of the application-owning regions in the failed MVS, which are running in another MVS, can continue processing the same transactions.

Although you might want to separate workloads to some extent, you should nevertheless still consider making all the application-owning regions the same. This is because it is easier to create identical CICS regions that are all capable of handling the same workload, rather than tailoring each region to suit a particular workload. With identical regions you control which transactions are routed to the application-owning regions through the dynamic transaction routing mechanism. This can be illustrated by the following scenario, which is illustrated in Figure 15 on page 69.

- Assume a configuration of 12 DL/I application-owning regions running in 3 MVS images, and designed to handle 3 main CICS-DL/I applications, identified as Appl\_A, Appl\_B, and Appl\_C. (This is similar to our target configuration shown in Figure 1 on page 18, but with four application-owning regions instead of three for the purpose of this example.)
- Appl\_C must always run in different application-owning regions from Appl\_A and Appl\_B, but Appl\_A and Appl\_B can run together in the same CICS regions.

- The ratio of transaction workload for the 3 applications is roughly 2:2:1 for Appl\_A, Appl\_B, and Appl\_C respectively.
- All the application-owning regions are identical, except for their APPLIDs and SYSIDNTs, which means they all have the same resource definitions installed, and are *capable* of running all three applications.
- Although the 12 application-owning regions are defined identically, the dynamic transaction routing program is designed to control which transactions they receive. This ensures that 9 of them never receive any Appl\_C transactions, and 3 of them never receive any Appl\_A or Appl\_B transactions.

Figure 15 shows which application-owning regions are allowed to process the 3 applications.

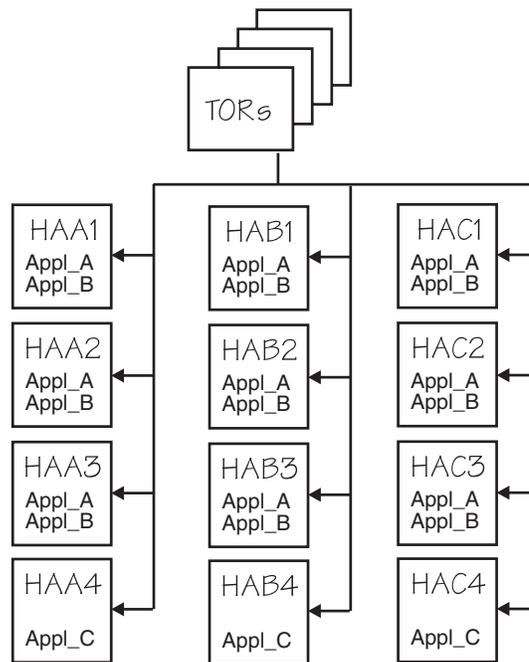


Figure 15. Plan Showing Which Applications the Regions Can Process

Alternatively, if we used the target configuration as shown in Figure 1 on page 18 with only 9 application-owning regions, there would only be 7 application-owning regions to handle applications Appl\_A and Appl\_B, and only 2 for Appl\_C. This would have the disadvantages that:

- A sudden surge of work for Appl\_C would be constrained by the capacity of only 2 application-owning regions.
- A failure of any MVS image nearly halves the processing capacity available to either Appl\_A and Appl\_B or to Appl\_C.

## START Commands In a Dynamic Transaction Routing Environment

When planning your CICSplex configuration for a dynamic transaction routing environment, you need to give special consideration to the use of EXEC CICS START commands.

## Start Commands that Do Not Specify TERMID

In the ideal case, transactions started without an associated terminal could run on any application-owning region. In this case, you define the transaction as a local transaction in all the application-owning regions. The started transaction adds to the load in its application-owning region but, on the assumption that the transactions that are issuing the START commands are evenly balanced across the application-owning regions, the started transactions will also be evenly balanced.

In some existing configurations the workload is separated into single-application application-owning regions, using static routing. However, over time, extensions to one application may require that it has access to transactions belonging to another application. A common way to achieve this is to use function-shipped START requests. The function shipping can be controlled by defining the transactions as remote, or by using the SYSID option on the START command.

To continue developing applications in this way undermines the simplicity of the cloning approach we are advocating. A better approach is to consider combining the applications into a multi-application AOR, and then clone the multi-application AOR. If one of the reasons for separating the applications is to prevent them overwriting each others storage, you can achieve the same protection using transaction isolation.

If it is possible to adopt this approach, you eliminate the need for connections between application-owning regions, and greatly simplify systems management as well as application development.

## START Commands that Specify TERMID

START commands that specify TERMID can be split into two categories: those that specify the principal facility (EIBTRMID) as the target terminal, and those that do not.

**When TERMID specifies the principal facility:** In this case, it is quite likely that you could adopt an alternative application design using the EXEC CICS RETURN command, with the TRANSID and IMMEDIATE options. It is not necessary to make this change, but this approach has great benefits compared with using EXEC CICS START with TERMID(EIBTRMID). It performs much better, it ensures the ordering of transactions, and it allows the dynamic transaction routing algorithm to make a new decision about where the second transaction should run. These benefits can justify the cost of changing the application.

In some existing configurations the workload is separated into single-application AORs, using static routing. However, over time, extensions to one application may require that it has access to transactions belonging to another application. A common way to achieve this is to use function-shipped START requests, specifying the transaction's principal facility as the target terminal. The function shipping can be controlled by defining the transactions as remote in the originating application-owning region, or by using the SYSID option on the START command.

A complication arises if the START command is function shipped to an application-owning region where the terminal (EIBTRMID) is unknown, probably because it was autoinstalled and the definition has not been shipped to the target application-owning region.

There are two ways you can resolve this problem:

1. If you combine the application-owning regions it is not necessary to function ship the START command, in which case the search for the target terminal will generally succeed in finding a definition in the invoking application-owning region. (This assumes the START command does not have a delay that allows the shipped terminal to be deleted.)
2. If you cannot combine the application-owning regions for some reason, you can change the application to use the EXEC CICS RETURN command, with the TRANSID and IMMEDIATE options. This causes the next transaction to be routed, via the terminal-owning region to the target application-owning region by existing mechanisms in the TOR (that is, by static remote transaction definitions or by a workload separation algorithm in the dynamic transaction routing program).

**When *TERMINID* does not specify the principal facility:** In the case where the target terminal is *not* the principal facility of the transaction issuing the START command, the situation is more complicated. The solution here depends on how the application determines the name of the target terminal. If it is a printer, it may well be predefined. If it is autoinstalled, a problem can arise if the terminal definition has not been shipped to the application-owning region. However, your application has to derive its name somehow, and the method should also provide a mechanism for a terminal not-found global user exit program (at exits XALTENF and XICTENF) to determine which terminal-owning region actually owns the target terminal.



## Chapter 6. Planning for VSAM Record-Level Sharing

This chapter discusses planning for migration from a CICS configuration that uses one or more CICS file-owning regions (FORs) to share VSAM files, to a configuration that uses VSAM record-level sharing (RLS). It covers the following topics:

- Concepts and use of RLS
- Preparing for RLS
- Planning migration and coexistence.

### Concepts and use of RLS

RLS is a mode of opening and accessing VSAM data sets, supported by DFSMS Version 1 Release 3 and by CICS. The RLS concept, with SMSVSAM servers replacing CICS file-owning regions, is illustrated in Figure 16.

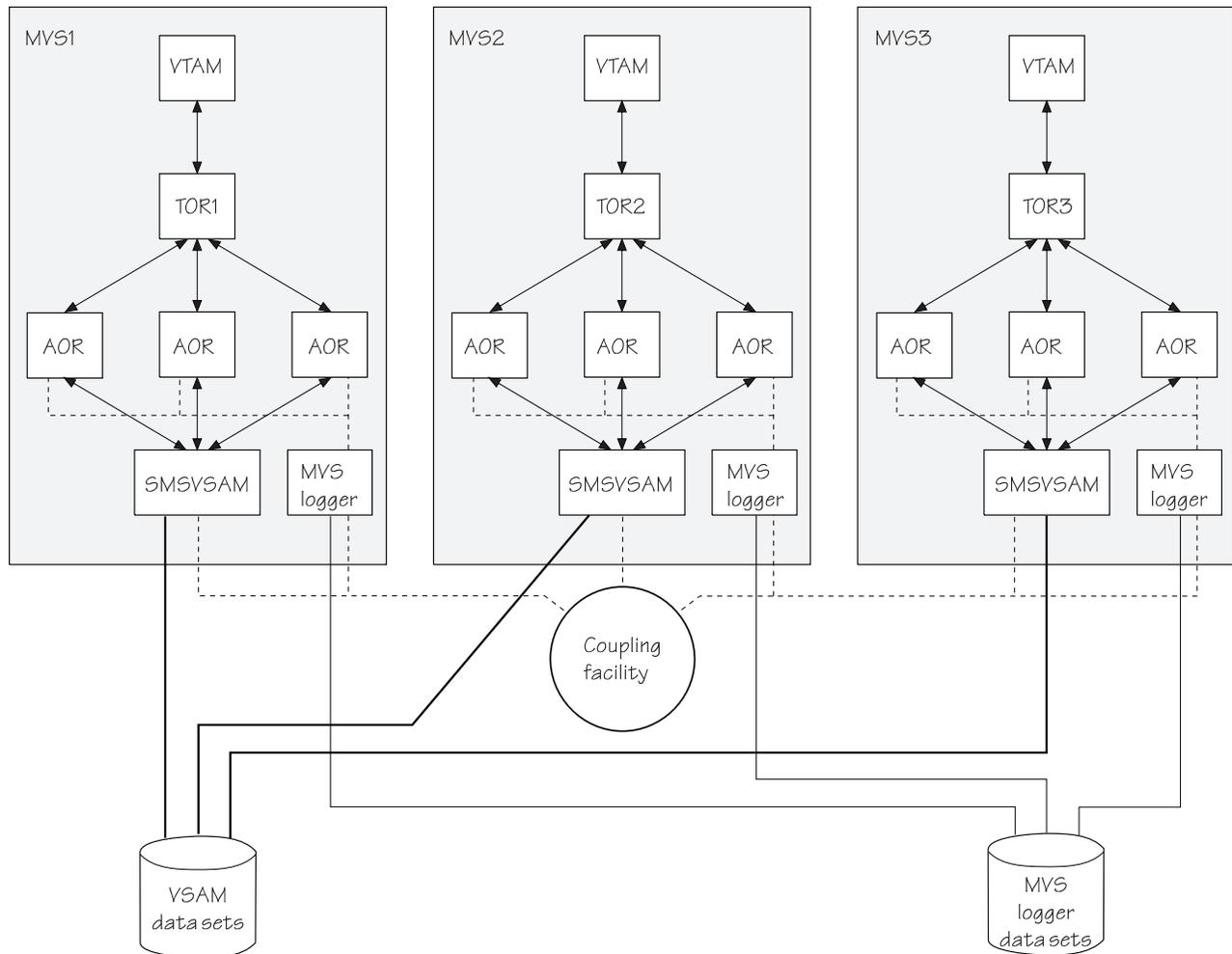


Figure 16. Conceptual View of Parallel Sysplex with an SMSVSAM Server in Each MVS Image

RLS enables VSAM data to be shared, with full update capability, between many applications running in many CICS regions across the sysplex.

With RLS, CICS regions that share VSAM data sets can reside in one or more MVS images within the sysplex.

The SMSVSAM server, which is generally initialized automatically during an MVS IPL, uses the coupling facility for its cache structures and lock structures.

CICS can access a VSAM file in three different modes. These are non-shared resources (NSR) mode, local shared resources (LSR) mode, or record-level sharing (RLS) mode. (CICS does not support VSAM global shared resources (GSR) access mode.) The mode of access is not a property of the data set itself—it is a property of the way that the data set is opened. For example, a given data set can be opened in NSR mode at one time, and RLS mode at another.

## Coupling Facility Requirements for VSAM Record-level Sharing

VSAM record-level sharing requires a coupling facility running coupling facility control code (CFCC) at CFLEVEL=2.

For information about the coupling facility and CFLEVEL, see *PR/SM Planning Guide*.

## Data Set Eligibility

Any VSAM data set supported and managed by CICS file control is eligible to be opened in RLS mode, including the CICS system definition (CSD) data set, subject to the restrictions described under “Restrictions for Data Sets Defined with IMBED”. This includes data sets that have an associated user-maintained shared data table, but not CICS-maintained shared data tables.

The CICS auxiliary temporary storage data sets (DFHAUXT and DFHBUXT) and the CICS transient data intrapartition data set (DFHINTRA) are **not** eligible for RLS-mode access.

## Restrictions for Data Sets Defined with IMBED

You cannot specify RLS access for data sets defined with the IMBED option. If you have data sets that are defined with IMBED, and you want to use them in RLS access mode:

- Redefine the data sets without the IMBED option
- Copy the the old data set to the new using the IDCAMS REPRO function.

You can then use the new data set in RLS mode.

When reviewing your data sets for RLS eligibility, also reconsider the use of the REPLICATE option. Although REPLICATE is supported by RLS, it does not provide any performance benefit, and can be omitted without penalty. Not using replication should save a little space on DASD.

## Choosing Between RLS-Mode and Non-RLS Mode

You should specify RLSACCESS(YES) on resource definitions only when you need to share the referenced data set between more than one CICS region.

The following are some things to consider when you are deciding whether data sets are good candidates for RLS access:

- If you have data sets that are used within a single MVS image, and whose access characteristics make them suitable for use as shared data tables, CICS

shared data table support will give you the greatest benefit. See the *CICS Shared Data Tables Guide* for more information.

- If you have data sets that are referenced by file definitions in only one CICS region, continue using those data sets in non-RLS access mode.
- If you have data sets accessed through an FOR to which you function ship requests from multiple AORs in the same sysplex, switch to RLS access mode, making the FOR redundant.
- If you are moving to a parallel environment, and want to create a number of clones of an AOR that owns its VSAM locally (an AOR/FOR), switch to RLS access mode to enable the cloned AORs to share the data without the need for an FOR.

## Restricting Switching Between RLS Mode and Non-RLS Mode Access

VSAM ensures that a sphere cannot be accessed simultaneously in RLS mode and any other VSAM mode. A sphere is a collection of all the component data sets associated with a given VSAM base data set—the base, index, alternate indexes, and alternate index paths. The first file-open request issued within the sysplex determines the type of access that is allowed until all files for the sphere are closed. Thus, if CICS regions have files open in RLS mode against a data set, batch jobs cannot open the data set in non-RLS mode until the CICS RLS-mode files are closed.

## General Rule About Switching Opening Modes

Except for read-only operations, as a general rule do not switch between RLS and non-RLS modes within CICS. After a data set is accessed for update in RLS mode by CICS, it should always be accessed in RLS mode by CICS. If you need to switch to non-RLS mode for batch processing, you should use the CICS QUIESCE function to take the data set offline from all CICS regions. When you issue the QUIESCE command in one CICS region, it is propagated throughout the CICSplex, causing all CICS regions to close their RLS ACBs. When the quiesce operation is completed by all CICS regions, you can run the non-RLS batch jobs (unless there are retained locks; see “Resolving Retained Locks” on page 76).

Take backups of data set copies for recovery purposes before and after a batch run as you would normally, regardless of whether you are switching from RLS to non-RLS access mode.

## Switching Modes Exception for Read-Only Operations

There is an exception to the general rule about not switching between RLS and non-RLS within CICS. You can switch to non-RLS access on a data set that is normally opened in RLS mode provided access is *restricted to read-only operations*. You might want to do this, for example, to allow continued access for read-only transactions while the data set is being updated by a batch job. CICS and VSAM permit quiesced data sets to be opened in non-RLS mode, but you must ensure that CICS transactions do not update a data set that is being updated concurrently by a batch program.

### Recommended Procedure

The recommended procedure for providing CICS read access to a recoverable data set while it is being updated by a batch job is:

1. Resolve retained locks (see “Resolving Retained Locks” on page 76)
2. Quiesce the data sets.

3. Redefine the files as non-RLS and read-only mode in all relevant CICS regions. You can do this using the CEMT, or EXEC CICS, SET FILE command.

**Note:** If your file definitions specify LSRPOOLIDs that are built dynamically by CICS, consider using the RLSTOLSR system initialization parameter.

4. Open the files in non-RLS read-only mode in CICS.
5. Concurrently, run batch non-RLS.
6. When batch work is finished:
  - a. Close the read-only non-RLS mode files in CICS.
  - b. Re-define the files as RLS mode and with update operations. You can do this using the CEMT, or EXEC CICS, SET FILE command.
  - c. Unquiesce the data sets.
  - d. Open the files in CICS, if not using open on first reference.
  - e. Resume normal running.

## Resolving Retained Locks

VSAM ensures that quiesced data sets can be opened only in non-RLS mode, but the quiesce operation does not guarantee that quiesced data sets can be opened by batch programs. If VSAM is holding any retained locks for a data set that is required by a batch program, the data set open request fails. CICS provides support to help you to resolve retained locks, in the form of SPI commands, and a suite of sample application programs that uses these commands.

You should set up your own procedures for dealing with retained locks, based on the procedures described in the *CICS Recovery and Restart Guide*.

---

## Preparing for RLS

In preparation for RLS, check whether you need to make any application program changes. In particular, review whether you want to use read integrity options, and check for the possible effect of the LOCKED exception condition.

## Read Integrity

You can specify read integrity options for READ-only requests, either on the READ command or on the file control resource definition.

You are recommended to specify repeatable and consistent READs in your applications only when they cannot tolerate “dirty” data. This is to avoid the potential locking overhead from the extra locking needed to ensure read integrity.

Before introducing read integrity, review your applications to see if read integrity is likely to introduce new deadlocks. See the *CICS Application Programming Guide* for information about the increased risk of deadlocks, particularly when defining read integrity on file resource definitions.

## The LOCKED Exception Condition

Check your application programs to ensure that they can handle the retained lock exception condition (LOCKED). The default CICS action for applications that do not handle the LOCKED exception condition, and do not specify HANDLE CONDITION ERROR, is to abend the transaction with an AEX8 abend code. Check existing applications to ensure that they are correctly coded to deal with *unexpected*

*conditions*. This should be of concern only in those application programs that specify NOHANDLE, or imply NOHANDLE by means of the RESP options.

Application programs that use NOHANDLE or RESP to deal with exception conditions in their own way must ensure that they are coded to handle unknown conditions, otherwise errors could occur. For example, if an existing application program contained the following statements, a LOCKED condition would be handled the same as any other known, or unknown, exception conditions that might be returned by CICS:

```
EXEC CICS READ UPDATE
      INTO(CUSTREC-LAYOUT)
      FILE(CUST-FILE-NAME)
      LENGTH(READ-LENGTH)
      RIDFLD(CUSTOMER-NUMBER)
      RESP(EXEC-RESPONSE)
END-EXEC.

EVALUATE EXEC-RESPONSE
  WHEN DFHRESP(NORMAL)
    PERFORM NORMAL-PROCESS
  WHEN OTHER
    PERFORM ERROR-RESPONSE
END-EVALUATE.
```

On the other hand, if an application program uses the EVALUATE clause to test for each of a list of *known* exception conditions, and the WHEN OTHER statement assumes a normal response, the results of a LOCKED condition are unpredictable.

Review suspect application programs and modify them as necessary.

See the *CICS Application Programming Reference* for information about the LOCKED exception condition that can occur on file control commands.

## Defining the Coupling Facility Structures

You must define the structures required by VSAM in the coupling facility before you can use RLS. These are:

- Cache structures and cache sets
- Lock structure
- Structures for use by the CICS log manager

### Defining the Cache Structures and Cache Sets

When you are converting data sets to RLS access mode, define the required cache structures in the coupling facility. Define the size of each coupling facility cache structure to provide approximately the same amount of space as that provided by the LSR pools and hiperspace used by the data sets that reference the cache structure. The structure should be at least large enough so that the coupling facility cache directory contains an entry for each of the RLS local buffers across all systems.

If you are using RLS to replace more than one file-owning region, the size of the cache should be at least as large as the sum of all the LSR pools being replaced.

You can have more than one cache structure defined within a cache set. This is of benefit because it can allow data sets to be reassigned to another cache in the event of a failure of the original cache.

For information about defining cache structures and cache sets, see the *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

## Defining the Lock Structure

To use VSAM RLS, you must define a single, non-volatile, master coupling facility lock structure. This lock structure is used to maintain the record-level locks for all data sets accessed by CICS in RLS mode. You should ensure that the coupling facility lock structure is accessible from all MVS images in the sysplex that need to support VSAM RLS.

The coupling facility master lock structure is named IGWLOCK00. Use the cross-system extended services (XES) coupling definition process to define it. See the *z/OS MVS Setting Up a Sysplex* for general information about XES.

To define the size of the coupling facility master lock structure, use the formula provided by VSAM in the *z/OS DFSMSdfp Storage Administration Reference*.

## Defining Structures for Use by the CICS Log Manager

When you move to an RLS environment from one in which multiple AORs have been accessing data sets in an FOR, the logging activity of the FOR is taken over by the AORs. The coupling facility structure size required by each AOR increases as a consequence of this. See the *CICS Transaction Server for OS/390 Installation Guide* for details on how to calculate the increased structure space required by an AOR on such a move to RLS.

## Defining the Sharing Control Data Sets

Define at least two active sharing control data sets and one spare sharing control data set.

See the *CICS Transaction Server for OS/390 Installation Guide* for information about defining these data sets.

## Defining SMS Storage Classes

All data sets accessed in RLS mode must reside on SMS managed storage. Define the appropriate SMS storage classes for the data sets that you want to access in RLS mode.

For information about defining storage classes for VSAM RLS, see the *z/OS DFSMSdfp Storage Administration Reference*.

## Defining IGDSMSxx Parameters in SYS1.PARMLIB

Review the IGDSMSxx member on SYS1.PARMLIB and set appropriate values for the DEADLOCK\_DETECTION, SMF\_TIME, CF\_TIME, RLS\_INIT, and RLS\_MAX\_POOL\_SIZE parameters.

For information about IGDSMSxx, see the *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

## Defining Deadlock Time Intervals

For files opened in RLS mode, VSAM, and not CICS, is responsible for detecting deadlocks and timeout conditions, and for providing associated diagnostic information.

- VSAM detects and resolves deadlocks between RLS requests.

A VSAM-detected deadlock causes CICS to abend the transaction with an AFCW abend code.

- VSAM detects timeouts that may have been caused by deadlocks between RLS and other resource managers, or caused by a single transaction holding a lock for an excessive amount of time.

For its timeout mechanism, VSAM uses a timeout interval value passed by CICS on the file request. VSAM returns a “timed-out” condition on any requests that wait for a lock for more than the timeout interval, causing CICS to abend the user task with an AFCV abend.

Although it is VSAM that implements the timeout mechanism, you define the timeout value to CICS. You do this using either:

- The DTIMOUT parameter on transaction resource definitions, or
- The CICS system initialization parameter, FTIMEOUT, to provide a global value for the CICS region.

A DTIMOUT value takes precedence over the global value.

## Defining recovery attributes for VSAM Data Sets

To support RLS access, VSAM provides some new data set attributes for defining whether the data set is recoverable or non-recoverable. You specify these when you define (or alter) the data set cluster, and these attributes are stored in the ICF catalog. You can also define backup-while-open (BWO) and forward recovery options in the ICF catalog. For data sets that are being accessed in RLS mode, the recovery and BWO attributes *must* be defined in the ICF catalog. If any of the new ICF attributes are also defined in the CICS file resource definition, they are ignored.

You can also use the ICF catalog to specify the recovery attributes for non-RLS files, in which case the equivalent information in CICS file resource definitions is ignored.

For large numbers of VSAM data sets, where some are accessed in RLS mode and some non-RLS, defining recovery attributes in two different places may add to the difficulty of data administration. In this case, you might want to keep things simple and define the recovery attributes for all files—RLS mode and non-RLS mode—in the ICF catalog.

The recovery-related parameters are:

- LOG(**{NONE|UNDO|ALL}**), to specify whether the data set is not recoverable, backward recoverable, or backward and forward recoverable.
- LOGSTREAMID(*name*) to specify the forward recovery log stream name for data sets defined with LOG(ALL).
- BWO(TYPECICS) to specify BWO support.

For information about these parameters, which are available on the access method services DEFINE CLUSTER and ALTER commands, see *z/OS DFSMS Access Method Services*, SC26-7394.

---

## Planning Migration and Coexistence

You do not have to change your FOR for SMSVSAM, and migrate all your CICS/ESA 4.1 regions to CICS Transaction Server regions, all at the same time. You can migrate progressively, especially during the test and development phases before cut-over into production.

For example, if you currently have a number of CICS/ESA 4.1 regions that access their VSAM files through a CICS/ESA 4.1 file-owning region, one migration approach is as follows:

1. Begin by migrating the FOR to a CICS Transaction Server region.
2. Leave the AORs at the CICS 4.1 level, continuing to function ship file control requests to the new FOR. Initially, the new FOR can continue using VSAM files in non-RLS mode.
3. When you are satisfied that the CICS Transaction Server region is functioning correctly in this mode, redefine the files as RLSACCESS(YES). The AORs continue to function ship their file requests, but the FOR actually uses SMSVSAM to access the data sets.
4. You can now progressively migrate the AORs to CICS Transaction Server for OS/390 Release 1, changing the remote file definitions to local file definitions, and changing the RLSACCESS(NO) attribute to RLSACCESS(YES).

This gradual migration process is illustrated in Figure 17 on page 81. The diagram shows the point in the migration process when 2 of the AORs remain at the CICS/ESA 4.1 level and 2 are migrated to CICS Transaction Server. The VSAM files in the CICS/ESA 4.1 regions are defined as remote and file requests continue to be function shipped to the FOR. The AORs running under CICS Transaction Server access files directly in RLS mode through the services of SMSVSAM.

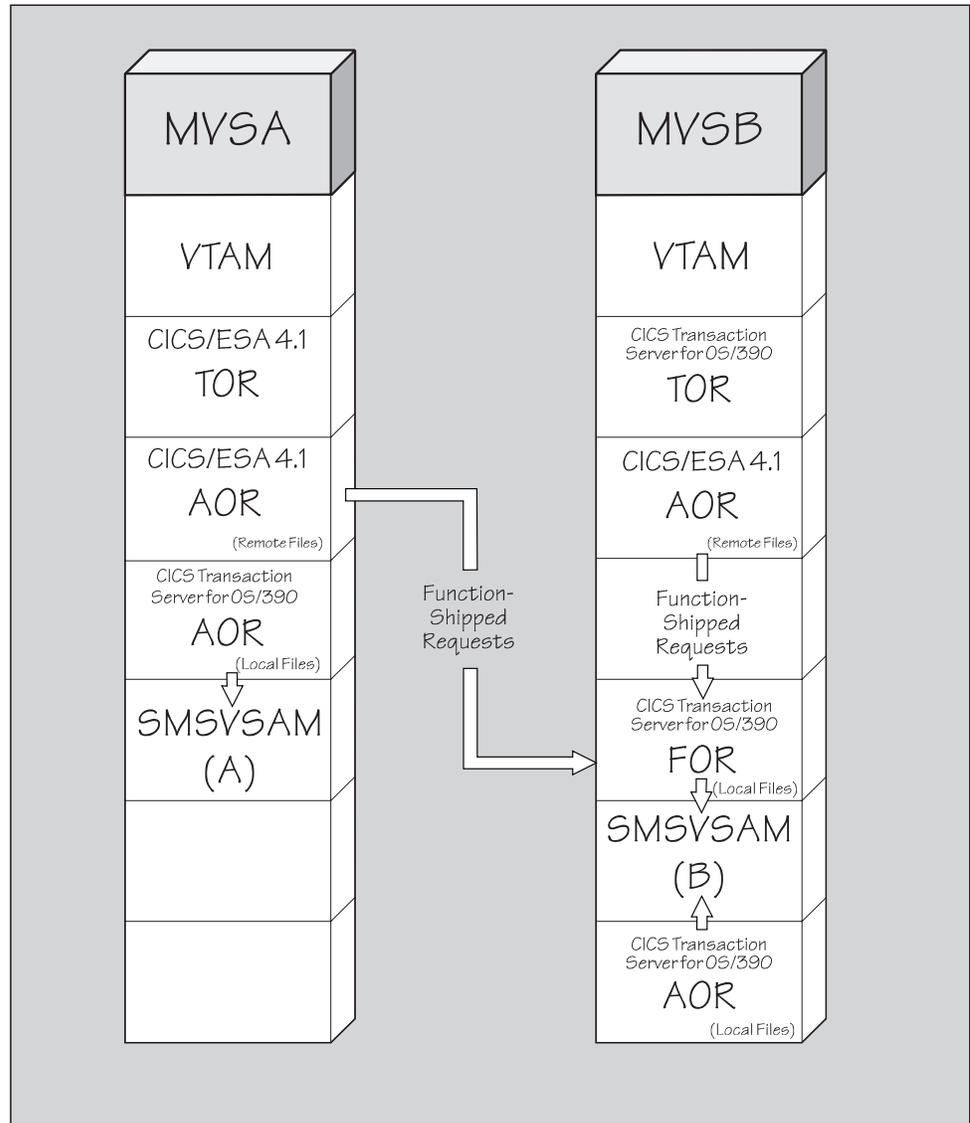


Figure 17. Migration Scenario Using a Mixture of Function Shipping and RLS

## Fallback Planning

Before you migrate a production region you should prepare a plan for reverting to your earlier release level if, for reasons connected with RLS or the MVS system logger, you are unable to continue. Some considerations for such fallback planning are discussed in the *CICS Transaction Server for OS/390 Migration Guide*.



---

## Chapter 7. Planning for Temporary Storage Data Sharing

This chapter discusses some considerations for CICS temporary storage (TS) data sharing across the sysplex using TS data sharing servers.

CICS temporary storage data sharing provides multiregion access to nonrecoverable temporary storage queues. TS data sharing allows your CICS applications to access nonrecoverable TS from multiple CICS regions running on any MVS image within a sysplex. The ability to share TS queues within a sysplex simplifies the migration of existing CICS applications to a Parallel Sysplex environment.

Although TS data sharing queues are not recoverable, and are not backed out as part of CICS transaction backout operations, they are normally preserved across a CICS region restart, or an MVS re-IPL, providing the coupling facility is not stopped and does not fail.

You do not need to change your application programs to exploit TS data sharing. The TS data sharing facility is available through the CICS application programming interface (API) for temporary storage.

Figure 18 on page 84 illustrates a TS data sharing configuration with TS servers that replace queue-owning regions. The TS servers are started by program DFHXQMN.

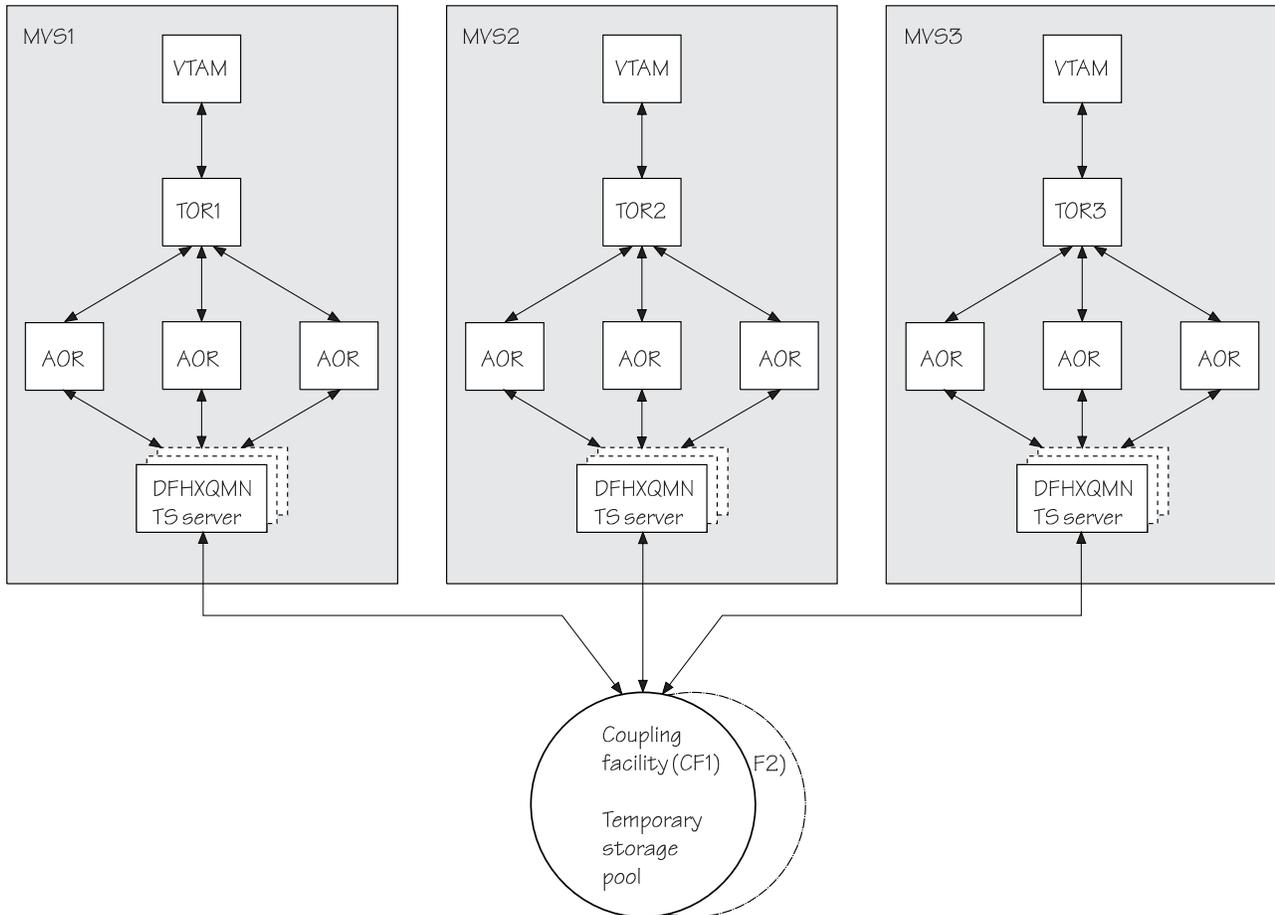


Figure 18. Conceptual View of Parallel Sysplex with a TS Server in Each MVS Image

## TS Pools and the Coupling Facility

CICS stores a set of TS queues that you want to share in a TS pool. Each TS pool corresponds to a coupling facility list structure defined in the CFRM policy.

You can create a single TS pool or multiple TS pools within the sysplex, to suit your requirements. For example:

- You can create separate pools for specific purposes—such as a TS pool for production, or a TS pool for test and development.
- You can create more than one production pool, particularly if you have more than one coupling facility and you want to allocate TS pool list structures to each coupling facility

## Defining Shared TS Queues

The addition of TS data sharing means there are now three types of TS queue. You can define your temporary storage queues as:

1. Local queues
2. Remote queues
3. Shared queues.

All these types of TS queue can be supported concurrently.

You specify the pool for a set of remote TS queues in one of the following ways:

- By defining, in the CSD, a TSMODEL resource definition that specifies the pool name with an associated generic queue name (using the PREFIX attribute, which is the equivalent of the DATAID in a TST entry).

**Note:** You cannot use a TSMODEL resource definition for applications that specify an explicit SYSID on the EXEC CICS temporary storage command. If you have application programs that specify a SYSID, either on the API command itself or added by an XTSEREQ global user exit program, you must continue to use a TST as described in the second method.

- A CICS temporary storage table (TST) using the DFHTST TYPE=SHARED macro. The TYPE=SHARED entry in the TST maps a remote SYSIDNT specified in a TYPE=REMOTE entry to a TS pool name. Thus, if a TS queue is defined as a remote queue in the temporary storage table, CICS checks for a corresponding SYSIDNT in a TYPE=SHARED entry to determine the TS pool name.

See the *CICS Resource Definition Guide* for information on how to define shared TS queues.

---

## The TS Data Sharing Server

Access to a TS pool by CICS transactions running in an AOR is through a TS data sharing server that supports a named pool. In each MVS image in the sysplex, start one TS server for each pool that can be accessed from that MVS image. See the *CICS System Definition Guide* for information about how to setup and start TS servers.

All TS pool access is performed by cross-memory calls to the TS server for the named pool. An AOR can access more than one TS server concurrently. This multi-server access is required if you create multiple pools, because each TS server provides access to only one pool of TS queues.

The methods for specifying a TS pool make it easy to migrate queues from a QOR to a TS data sharing pool. If you have a TYPE=REMOTE entry in the TST that specifies the SYSIDNT of a QOR, add a TYPE=SHARED entry, specifying a corresponding SYSIDNT and the name of the TS pool in which the queue resides.

It is possible to use the TS global user exit, XTSEREQ, to modify or add the SYSID on a TS request so that it references a TS data sharing pool. However, this restricts your use of RDO for temporary storage definitions, and forces you to continue using a temporary storage table with TYPE=SHARED entries. See the *CICS Customization Guide* for information about CICS temporary storage global user exits.

---

## The Subsystem Interface

CICS regions use MVS cross-memory connection services to access the TS data sharing server(s). These services (authorized cross-memory (AXM) server environment services) are defined using the MVS subsystem interface (SSI). AXM uses the SSI definition to schedule initialization in the master scheduler address space. The MVS subsystem interface for AXM is not activated or used.

The AXM subsystem is normally defined in the IEFSSNxx member of SYS1.PARMLIB. This ensures that AXM system services are made automatically available at IPL.

See the *CICS System Definition Guide* for more information about setting up and starting TS servers.

---

## Security

Access to TS pools by CICS regions is controlled by an external security manager, which can be the SureWay Security Server, RACF, or an external security manager that provides equivalent function.

The security checks are to ensure that:

- The TS server is authorized to access the TS pool structure in the coupling facility.
- The TS server is authorized to act as a server for the TS pool.
- The AOR issuing the request is authorized to attach to the TS server.

See the *CICS RACF Security Guide* for information about authorizing access to TS servers and TS pools.

The TS server does not perform security checks on individual requests. The AOR continues to be responsible for resource security checks if you need to control user access to temporary storage queues.

---

## Chapter 8. Planning for Coupling Facility Data Tables

This chapter discusses some considerations for CICS coupling facility data tables (CFDTs) to provide shared file access across the sysplex using CFDT servers.

Coupling facility data tables provide a method of file data sharing, using CICS file control, without the need for a file-owning region, and without the need for VSAM RLS support. CICS CFDT support is designed to provide rapid sharing of working data within a sysplex, with update integrity. The data is held in a coupling facility, in a table that is similar in many ways to a shared user-maintained data table.

Because read access and write access have similar performance, this form of table is particularly useful for scratchpad data. Typical uses might include sharing scratchpad data between CICS regions across a sysplex, or sharing of files for which changes do not have to be permanently saved. There are many different requirements for scratchpad data, and most of these can be implemented using CFDTs. Coupling facility data tables are particularly useful for grouping data into different tables, where the items can be identified and retrieved by their keys. For example, you could use a CFDT to maintain a list of the numbers of lost credit cards.

---

### Comparison with User-maintained Data Tables

To an application, a CFDT appears much like a sysplex-wide user-maintained data table, because it is accessed in the same way using the file control API. However, in a CFDT there is a maximum key-length restriction of 16 bytes.

---

### Coupling Facility Data Table Models

There are two models of coupling facility data table:

- The contention model, which gives optimal performance but generally requires programs written to exploit it. This model is non-recoverable: CFDT updates are not backed out if a unit of work fails.
- The locking model, which is API-compatible with programs that conform to the user-maintained data table subset of the file control API (this subset is nearly, but not quite, the full file control API).

This model can either be:

- **Non-recoverable:** locks do not last until syncpoint, and CFDT updates are not backed out if a unit of work fails, or
- **Recoverable:** coupling facility data tables are recoverable in the event of a unit of work failure and in the event of a CICS region failure (in that updates made by units of work that were in-flight at the time of the CICS failure are backed out).

Figure 19 on page 88 illustrates a coupling facility data table configuration with CFDT servers that replace file-owning regions with user-maintained data tables. The CFDT servers are started by program DFHCFMN.

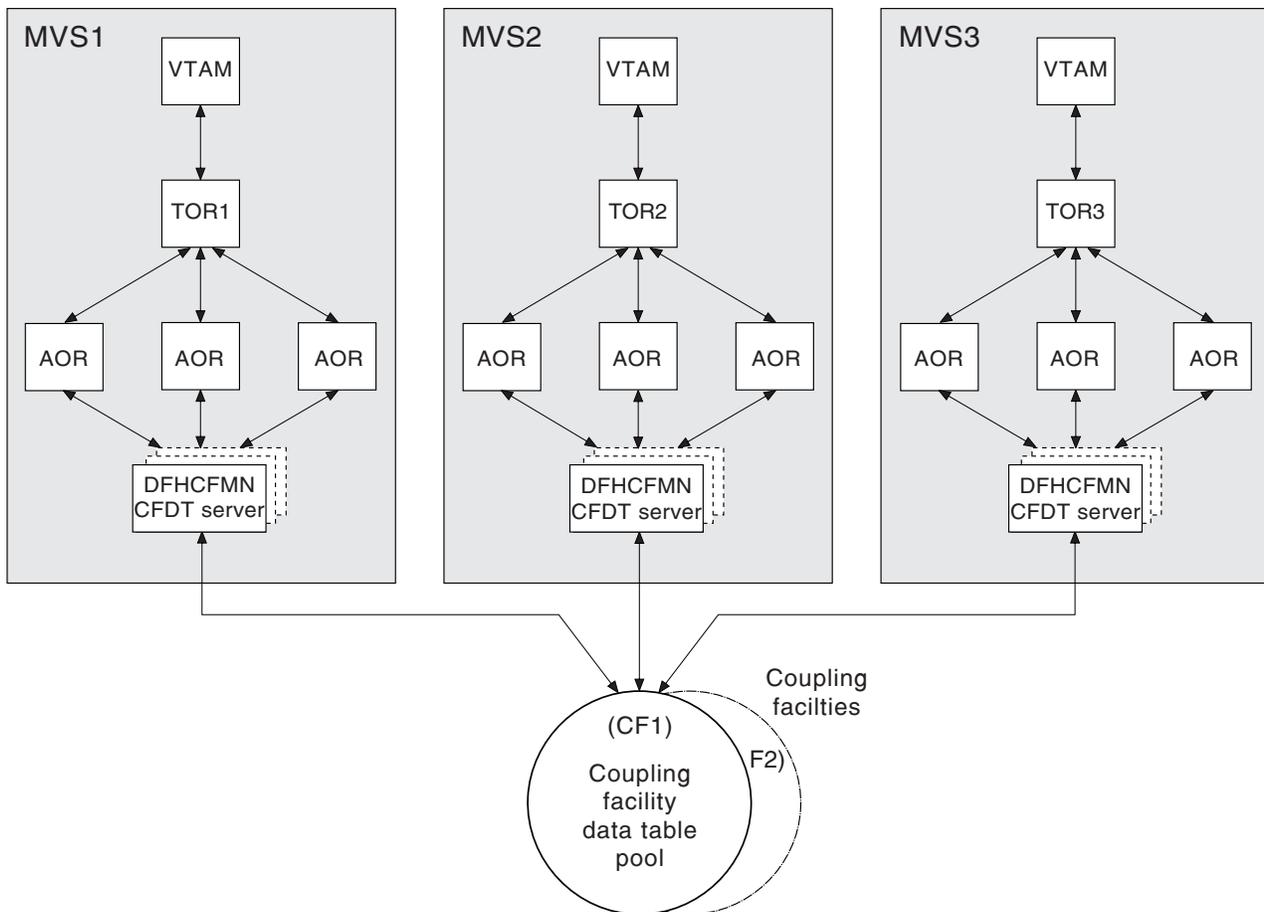


Figure 19. Conceptual View of Parallel Sysplex with a CFDT Server in Each MVS Image

## CFDT Pools and the Coupling Facility

CICS stores one or more CFDTs in a CFDT pool. Each CFDT pool corresponds to a coupling facility list structure defined in the CFRM policy.

You can create a single CFDT pool or multiple CFDT pools within the sysplex, to suit your requirements. For example:

- You can create separate pools for specific purposes—such as a CFDT pool for production, or a CFDT pool for test and development.
- You can create more than one production pool, particularly if you have more than one coupling facility and you want to allocate CFDT pool list structures to each coupling facility

See the *CICS System Definition Guide* for information about defining a list structure for a coupling facility data table pool.

---

## Defining a Coupling Facility Data Table

With coupling facility data tables there are three types of data table. You can define your data table as:

- A CICS-maintained data table
- A user-maintained data table
- A coupling facility data table.

You specify the pool for a coupling facility data table on the file resource definition for the file that your application programs use to reference the table, using the DATATABLE attributes, TABLE and MAXNUMRECS. If you specify TABLE(CF) you also specify the required CFDATATABLE attributes, CFDTPOOL, TABLENAME UPDATEMODEL, and LOAD.

See the *CICS Resource Definition Guide* for information on how to define coupling facility data table file definitions.

---

## The Coupling Facility Data Table Server

Access to a CFDT pool by CICS transactions running in an AOR is through a CFDT server that supports a named CFDT pool. In each MVS image in the sysplex, start one CFDT server for each pool that can be accessed from that MVS image. See the *CICS System Definition Guide* for information about how to setup and start CFDT servers.

All CFDT pool access is performed by cross-memory calls to the CFDT server for the named pool. An AOR can access more than one CFDT server concurrently. This multi-server access is required if you create multiple pools, because each CFDT server provides access to only one pool of coupling facility data tables.

The methods for specifying a CFDT pool make it easy to migrate a file from an FOR to a CFDT pool. If you have a file resource definition specified as TABLE(NO), you can change this to TABLE(CF), and add the other required attributes, such as the pool name, and the CFDT server will automatically build the table in the relevant CFDT pool. You can switch back to using an ordinary VSAM file by changing TABLE(CF) back to TABLE(NO), and any other CFDT attributes are ignored.

---

## The Subsystem Interface

CICS regions use MVS cross-memory connection services to access the CFDT server(s). These services (authorized cross-memory (AXM) server environment services) are defined using the MVS subsystem interface (SSI). AXM uses the SSI definition to schedule initialization in the master scheduler address space. The MVS subsystem interface for AXM is not activated or used.

The AXM subsystem is normally defined in the IEFSSNxx member of SYS1.PARMLIB. This ensures that AXM system services are made automatically available at IPL.

See the *CICS System Definition Guide* for more information about setting up and starting CFDT servers.

---

## Security

Access to CFDT pools by CICS regions is controlled by an external security manager, which can be the SureWay Security Server, RACF, or an external security manager that provides equivalent function.

The security checks are to ensure that:

- The CFDT server is authorized to access the CFDT pool structure in the coupling facility.
- The CFDT server is authorized to act as a server for the CFDT pool.
- The AOR issuing the request is authorized to attach to the CFDT server.

See the *CICS RACF Security Guide* for information about authorizing access to CFDT servers and CFDT pools.

The CFDT server does not perform security checks on individual requests. The AOR continues to be responsible for resource security checks if you need to control user access to files.

---

## Chapter 9. Planning for Named Number Counters

This chapter discusses some considerations for CICS named number counters, which provide a facility for generating unique sequence numbers throughout the sysplex, using named counter servers.

CICS provides this facility for use by applications in a Parallel Sysplex environment (for example, to allocate a unique number for orders or invoices). The facility is provided by a named counter server, which maintains each sequence of numbers as a named counter. Each time a sequence number is assigned, the corresponding named counter is incremented automatically so that the next request gets the next number in sequence.

The named counter server is modeled on the other coupling facility servers used by CICS, and has many features in common with the coupling facility data table server.

A named counter server provides a full set of functions to define and use named counters. Each named counter consists of:

- A 16-byte name
- A current value
- A minimum value
- A maximum value.

The values are internally stored as 8-byte (double word) binary numbers, but the user interface allows them to be treated as any length from 1 to 8 bytes, typically 4 bytes.

Named counters are stored in a pool of named counters, where each pool is a small coupling facility list structure, with keys but no data. The pool name forms part of the list structure name. Each named counter is stored as a list structure entry keyed on the specified name, and each request for the next value requires only a single coupling facility access.

---

### The Named Counter Application Programming Interfaces

You access the named counter through one of two application programming interfaces:

- The CICS API, which is provided for use in CICS application programs
- A callable interface, which can be used in batch jobs, sharing the same named counters as CICS regions

Figure 20 on page 92 illustrates a named counter server configuration. The named counter servers are started by program DFHNCMN.

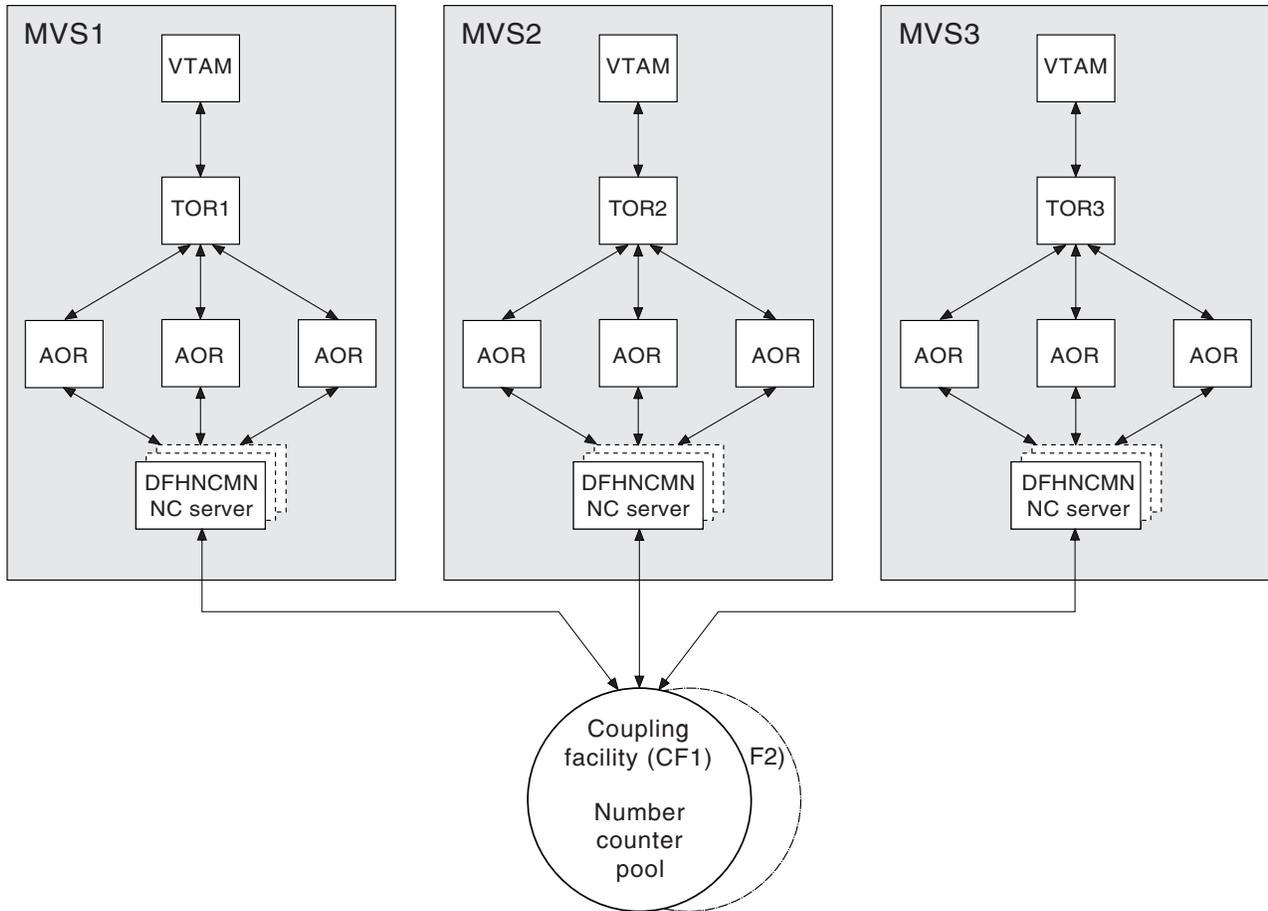


Figure 20. Conceptual View of Parallel Sysplex with a Named Counter Server in Each MVS Image

## Named Counter Pools and the Coupling Facility

CICS stores one or more named counters in a named counter pool. Each named counter pool corresponds to a coupling facility list structure defined in the CFRM policy.

You can create a single named counter pool or multiple pools within the sysplex, to suit your requirements. For example:

- You can create separate pools for specific purposes—such as a named counter pool for production, or a named counter pool for test and development.
- You can create more than one production pool, particularly if you have more than one coupling facility and you want to allocate named counter pool list structures to each coupling facility

See the *CICS System Definition Guide* for information about defining a list structure for a named counter pool.

---

## Defining a Named Counter Options Table

Unlike the other CICS data sharing facilities (coupling facility data tables and temporary storage data sharing) there are no CICS resource definitions for named counters. Instead, you need a named counter options table, DFHNCOPT, and when using named counters from a CICS region, you may also need to specify the NCPLDFT system initialization parameter.

The named counter programming interface determines the actual pool name in response to a request by referring to the DFHNCOPT options table. CICS supplies a default DFHNCOPT in source form, which you can customize and generate using the DFHNCO macro. A typical use of the options table is to enable production and test regions to use a different counter pool without needing to change the pool name in application programs. If the pool name cannot be resolved by reference to the named counter options table in response to a CICS API request, the default pool name specified on the NCPLDFT system initialization parameter is used.

See the *CICS System Definition Guide* for information on how to define a named counter options table.

---

## The Named Counter Server

Access to a named counter pool by CICS transactions running in an AOR, or by a batch program, is through a named counter server that supports a named named counter pool. In each MVS image in the sysplex, start one named counter server for each pool that can be accessed from that MVS image. See the *CICS System Definition Guide* for information about how to setup and start named counter servers.

All named counter pool access is performed by cross-memory calls to the named counter server for the named pool. An AOR or batch program can access more than one named counter server concurrently. This multi-server access is required if you create multiple pools, because each named counter server provides access to only one pool of named counters.

---

## The Subsystem Interface

CICS regions and the callable interface use MVS cross-memory connection services to access the named counter server(s). These services (authorized cross-memory (AXM) server environment services) are defined using the MVS subsystem interface (SSI). AXM uses the SSI definition to schedule initialization in the master scheduler address space. The MVS subsystem interface for AXM is not activated or used.

The AXM subsystem is normally defined in the IEFSSNxx member of SYS1.PARMLIB. This ensures that AXM system services are made automatically available at IPL.

See the *CICS System Definition Guide* for more information about setting up and starting named counter servers.

---

## Security

Access to named counter pools is controlled by an external security manager, which can be the SureWay Security Server, RACF, or an external security manager that provides equivalent function.

The security checks are to ensure that:

- The named counter server is authorized to access the named counter pool structure in the coupling facility.
- The named counter server is authorized to act as a server for the named counter pool.
- The AOR or batch program issuing the request is authorized to attach to the named counter server.

See the *CICS RACF Security Guide* for information about authorizing access to CFDT servers and CFDT pools.

The CFDT server does not perform security checks on individual requests. The AOR continues to be responsible for resource security checks if you need to control user access to files.

---

## Chapter 10. Planning Resource-Owning Regions

Chapter 6 and “Chapter 7. Planning for Temporary Storage Data Sharing” on page 83 describe how you can share VSAM data and CICS temporary storage queues, using SMSVSAM servers and CICS temporary storage servers, respectively. This chapter describes how CICS regions can use CICS function shipping to access data sets and temporary storage queues when the server methods are inappropriate.

There are two main sections:

- “Planning the File-Owning Regions”, which discusses CICS file control data sharing using a file-owning region (FOR)
- “Planning the Queue-Owning Regions” on page 100, which discusses CICS temporary storage data sharing using a queue-owning region (QOR).

---

### Planning the File-Owning Regions

In Chapter 2. Planning the Subsystem Configuration for a Sysplex, the proposed target configuration does not show any file-owning regions (FORs), because it is assumed that the CICS AORs access VSAM data sets in RLS mode. However, it is possible that not all the VSAM data sets used by your CICS applications are suitable candidates for RLS access, or your applications might use some BDAM data sets. This § discusses factors to consider when you need to make such data sets available to all AORs by function shipping file requests to an FOR.

With files defined as remote CICS files, the application-owning regions function ship CICS file control requests to the remote region that is defined as owning the files. This enables CICS to operate a form of data sharing, using its file control facility.

### Function Shipping

Function shipping allows you to define VSAM (and BDAM) data files as remote resources. This allows application programs to request data set services from a connected CICS region where the data sets are physically defined.

The principal reason for function shipping CICS file control requests is to enable more than one CICS region to have access to VSAM and BDAM data that would otherwise be available to only a single CICS region. This is because, for integrity reasons, only one CICS region can have a data set open at any one time; and in a dynamic transaction routing environment you need the ability to access VSAM data from a number of application-owning regions.

Figure 21 on page 96 shows a typical function-shipping configuration. In this example, an application program running in CICS region HAA1 issues a CICS command to read from a CICS VSAM file, FILEX. HAA1 searches for an entry for FILEX in its file control table and finds that the file is defined as a remote file on HFA1. HAA1 sends the request to HFA1, which initiates a CICS-supplied *mirror transaction* (the CSMI transaction for file control requests) to issue the request again on behalf of HAA1. HFA1 searches for an entry for FILEX in its file control table, which it finds is a local entry (although it could have been remote, on yet another system).

HFA1 completes the read request and the mirror task returns the data record (or error information) to HAA1 which, in turn, passes the data to the original requesting program.

Generally, unless the file-owning region is running with long-running mirrors specified (the default for MRO), the mirror task remains active only as long as necessary to complete the request, then terminates. For update requests, however, mirror tasks automatically become long-running because the mirror must wait until the syncpoint request is received from the requestor (the application-owning region).

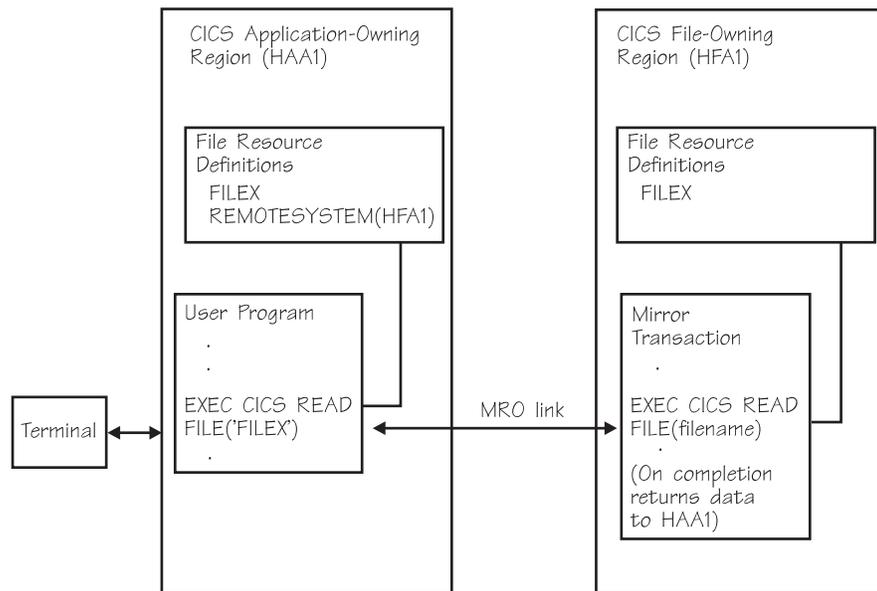


Figure 21. Function Shipping. This diagram illustrates a typical function-shipping configuration. Two CICS systems, HAA1 and HFA1, are linked by an MRO session.

CICS file control manages its own record locks to ensure a record can be accessed by only one file request at a time, using its own record locking mechanism. It also manages data integrity in the event of failure, ensuring that uncommitted updates are backed out.

When planning for the function shipping of CICS file control requests to a file-owning region, you must consider the following factors:

- Number of file-owning regions
- Availability of the data
- Capacity planning
- Data integrity
- Security.

## Planning the Number of File-Owning Regions

You need to consider how many file-owning regions you need in your CICSplex, and the possible effect of function shipping on performance when your VSAM data sets are owned by more than one file-owning region.

When, for a single application program, requests are function shipped for different VSAM resources to more than one file-owning region, this is more costly than function shipping all the requests to the same file-owning region. This is because of the overhead of the attach and detach mechanisms for the mirror tasks, and the

additional MRO flows that are required at the end of each task. Sending all requests from the same transaction instance to the same file-owning region is the most efficient in performance terms, because the requests can all be handled by the same mirror task. To ensure the use of the same mirror to handle separate requests from the same transaction instance, you are recommended to specify long running mirror tasks for your file-owning regions.

Note that browse requests and update requests from an application program are automatically handled by the same mirror. This is illustrated in Figure 22.

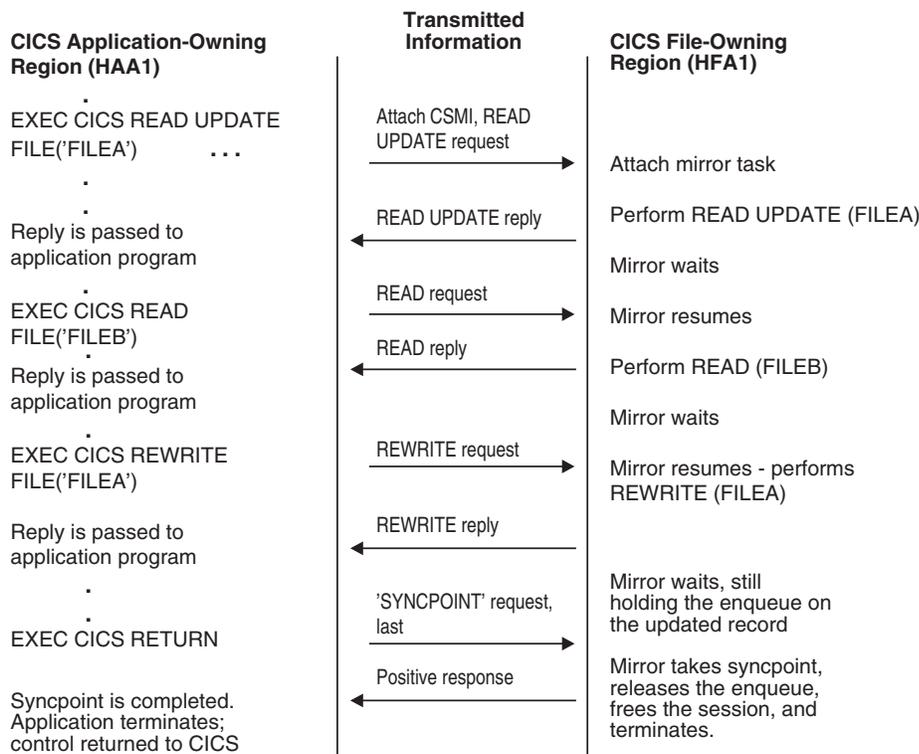


Figure 22. Function Shipping—Update. Because the mirror must wait for the REWRITE, it becomes long-running and does not terminate until SYNCPOINT is received.

**Note:** Figure 22 shows the main intercommunication flows for the simplest case—it is not meant to illustrate all the possible MRO flows.

This means that, for performance reasons, you should consider defining to the same file-owning region all the data sets that are referenced by the same application program.

See the *CICS Intercommunication Guide* for more information about CICS function shipping and mirror transactions.

### Avoiding Bottlenecks in the CICSplex

Sending all requests from the same transaction instance to the same file-owning region is the most efficient in performance terms. On the other hand, if you concentrate the ownership of all the VSAM data sets into one file-owning region you risk overloading that single CICS region. You should study your applications, and their use of the various data sets, and try to determine what is the optimum number of file-owning regions, taking account of all the factors.

Clearly, if some applications have their own data sets, the best approach would probably be to create a separate file-owning region for these data sets. For example, if the application marked Appl\_A illustrated in Figure 15 on page 69, does not share data with any of the other applications, all the Appl\_A data sets could be grouped in one file-owning region. The same is true for the applications labeled Appl\_B and Appl\_C. Therefore, depending on the degree of data sharing between the applications, we could allocate up to three file-owning regions to support the three illustrated applications. Basically, the more file-owning regions you have, the better it is for capacity reasons, and the less risk there is of causing intersystem queuing. Queuing caused by overloading a file-owning region can lead to performance degradation in regions connected to the file-owning region, and this can spread back to affect the terminal-owning regions.

## Ensuring Availability of the Data

In the previous § we discussed the possibility of using more than one file-owning region for capacity reasons. Another reason for operating a CICSplex with more than one file-owning region is availability.

If a CICSplex is configured with only one file-owning region, that region becomes a potential single point of failure. For example, if all the data sets required by the three applications illustrated in Figure 15 on page 69 were owned by a single file-owning region, a failure of that file-owning region would cause the failure of all the applications. Separating the ownership of the files, where possible, into different file-owning regions ensures that the loss of one file-owning region does not cause the loss of all applications.

As a general rule, you should try to avoid installing separate sets of files, required by different applications, in the same file-owning region.

## Capacity Planning Considerations

Another factor to consider when planning the number and allocation of file-owning regions is the use of central processor resources.

Like all CICS regions, a CICS file-owning region runs tasks, such as the MRO mirror task, under the CICS quasi-reentrant TCB, with some limited subtasking under the concurrent TCB. This means that, regardless of how many central processors are available to an MVS image, a CICS file-owning region can only utilize just over one CP on average. The following are some general points to note about the allocating of file-owning regions:

- You should ensure enough processing capacity for any file-owning region allocated to an MVS image in the sysplex.  
If a file-owning region is short of central processor resource, this situation could create a bottleneck that affects all regions in the CICSplex that are linked to that file-owning region.
- If you create multiple file-owning regions, do not assign them all to the same MVS image.  
If an MVS image fails, you lose access to all the VSAM data. Also, positioning a file-owning region on the same MVS image as some of the application-owning regions is better for performance reasons. This is because then at least a proportion of the function-shipped requests are across MRO links within the same MVS image, which gives slightly better performance than XCF/MRO.

## Data Integrity Considerations

When planning the distribution of files among multiple file-owning regions, you must consider the recovery implications for data integrity.

Data integrity in an MRO environment involves CICS *recoverable resources*. These are any resources that have changes backed out, if necessary, during dynamic transaction backout, or during a CICS emergency restart.

VSAM files are CICS resources that can be defined as recoverable resources.

The main data integrity factor you have to consider when choosing the number of file-owning regions for your CICSplex is the *in-doubt period* (also sometimes referred to as the *in-doubt window*).

### The In-doubt Period

As shown by the illustration of a function-shipping update request in Figure 22 on page 97, a SYNCPOINT request is sent by the application-owning region to the file-owning region to commit any changes to CICS recoverable resources. Until the file-owning region responds to confirm that it has obeyed the SYNCPOINT request, there is a period during which the application-owning region does not know whether the file-owning region has actually committed the changes.

This period is known as the in-doubt period, and is illustrated in Figure 23.

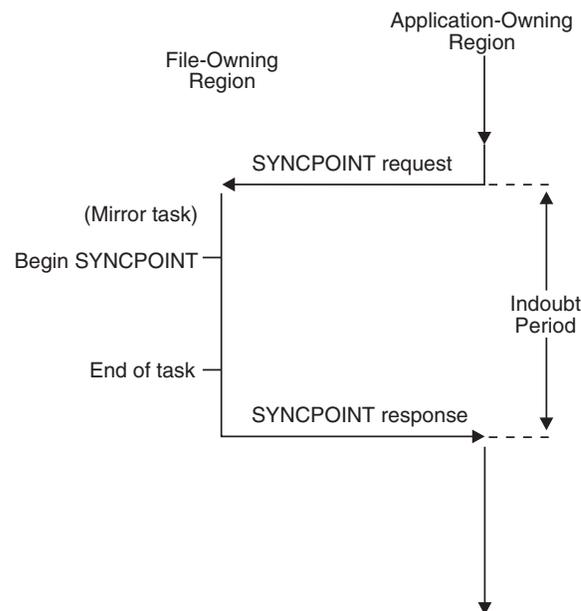


Figure 23. SYNCPOINT Processing in an MRO Environment

In the event of a failure of either the application-owning region or the file-owning region, or the loss of the link between the regions during the in-doubt period, CICS backs out the changes made to any recoverable resources according to the INDOUBT attribute on the transaction resource definition. If both the application-owning region and file-owning region have made changes to (different) recoverable resources, the recovery of these can be inconsistent in certain circumstances. This situation cannot arise for VSAM data in our target CICSplex

shown in Figure 1 on page 18, because the application-owning regions do not own any recoverable VSAM resources. In our CICSplex these are owned by the file-owning region.

However, the integrity exposure does exist in the following types of scenario:

- The same logical unit of work in an application-owning region updates remote recoverable resources that are owned by more than one file-owning region.
- The same logical unit of work in an application-owning region updates some local recoverable resources and some remote recoverable resources owned by a file-owning region.
- The same logical unit of work in an application-owning region updates remote recoverable resources that are owned by different resource managers and one of the resource managers is a CICS file-owning region. For example, resources owned by either DBCTL or DB2 on the one hand, and by a file-owning region on the other.

#### Allocating ownership of recoverable VSAM resources

As a general rule, you should ensure that all recoverable VSAM data sets that are updated by one logical unit of work are owned by the same CICS region—a file-owning region in our case.

This does not mean that you cannot have more than one file-owning region in the CICSplex; it means that you must carefully allocate the ownership of the VSAM data sets on the basis of recoverability and your data integrity requirements:

- Avoid distributing recoverable resources among multiple CICS regions if they are to be updated by the same application transaction.
- Where a transaction accesses data from different VSAM data sets, and some is defined as nonrecoverable data and some as recoverable data, ensure that all the recoverable data sets are owned by the same CICS file-owning region.

The nonrecoverable data sets can be owned by a different file-owning region without risk to data integrity.

**Note:** All these data integrity considerations apply to other CICS recoverable resources, such as temporary storage. See “Planning the Queue-Owning Regions” for information about recoverable queues.

---

## Planning the Queue-Owning Regions

In Chapter 2. Planning the Subsystem Configuration for a Sysplex, the proposed target configuration does not show any queue-owning regions (QORs), because it is assumed that the CICS AORs access remote temporary storage queues through a temporary storage server. However, TS data sharing through a TS server is restricted to non-recoverable TS queues, therefore, if you want to share recoverable temporary storage data between CICS regions a QOR is the only solution .

This topic discusses why you should consider defining temporary storage queues and transient data queues as remote resources owned by a queue-owning region. (The main reason for doing this is to avoid inter-transaction affinity associated with CICS queues.)

It also considers a number of factors that you should take into account when planning to create a queue-owning region.

## Avoiding Inter-transaction Affinity Associated with CICS Queues

To ensure maximum flexibility, dynamic transaction routing should not be constrained by a dependence on CICS temporary storage queues.

As with VSAM files, you can make CICS temporary storage and transient data queues globally accessible across the CICSplex by defining the queues to the application-owning regions as remote queues owned by one or more queue-owning regions. You should consider creating a queue-owning region to avoid potential, or actual, inter-transaction affinity problems associated with the use of CICS temporary storage, or transient data, queues.

For example, consider a temporary storage queue, CICSTSQ1, that is locally owned by CICS application-owning region DAA1, as shown in Figure 24.

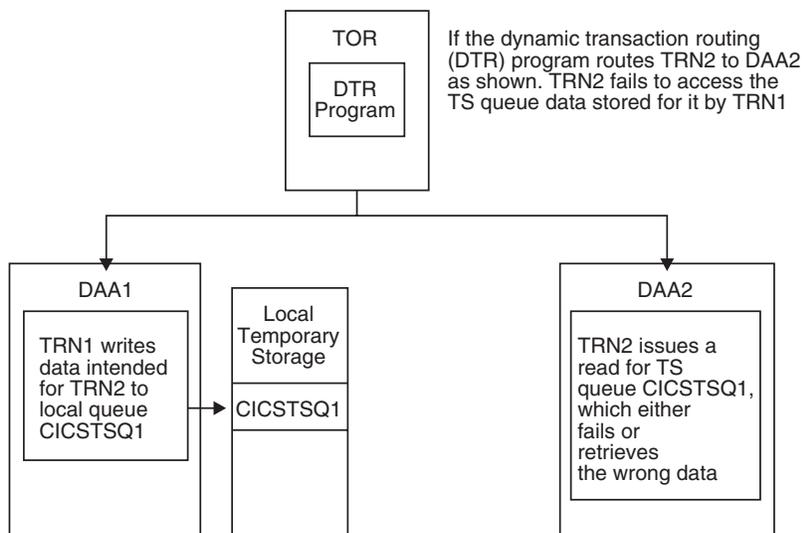


Figure 24. Example of Inter-transaction Affinity Caused by Using Local Temporary Storage

The steps shown in Figure 24 are as follows:

- Transaction TRN1 running in DAA1 writes temporary storage data to CICSTSQ1, which is intended to be read by TRN2, and terminates. Typically, TRN1 might be the first phase of a pseudo-conversational transaction.
- TRN2, the next phase of the pseudo conversation, is invoked by the terminal user, and is routed by the dynamic transaction routing program to a different application-owning region, DAA2.
- The TRN2 task is attached in DAA2, and attempts to read the data item written for it by TRN1 from temporary storage queue, CICSTSQ1. Because the data was written by TRN1 to a local temporary storage queue, owned by CICS DAA1, the attempt to retrieve the data fails. The failure will be one of two types:
  1. CICS returns a read error because TRN2 is attempting to read from a nonexistent queue, or
  2. CICS returns the wrong data if DAA2 also has a temporary storage queue called CICSTSQ1.

How you resolve this situation depends to some degree on the nature of the transactions:

- In a pseudo-conversational transaction, the best way is to change the program to use a COMMAREA to pass data between the phases of the conversation. However, this involves changes to the application programs, which you may not want or be able to do, in which case you have to adopt the same solution as for the non-pseudoconversational case.
- For non-pseudoconversational transactions, you can:
  - Create a remote region to own the queue and handle function-shipped requests for CICSTSQ1 data
  - Define the queue, CICSTSQ1, to the application-owning regions as a remote queue in a CICS temporary storage table (TST)
  - Install the TST defining the remote queues in each application-owning region.

A transaction can issue a request in any application-owning region that contains a remote definition for the queue, and that request is satisfied by being function shipped to the remote queue-owning region.

Of course, not all temporary storage queues need to be shared. Some queues can be locally owned (when there is no question of inter-transaction affinity), or queues can be replicated. Also, provided the affinity is not too pervasive, you can choose to accept a certain level of inter-transaction affinity, and manage this by means of logic in your dynamic transaction routing program. For example, CICSplex SM can manage inter-transaction affinities for and send affected transactions to the same application-owning region.

## Creating a Queue-Owning Region

Some of the factors to consider when you plan the queue-owning regions are the same as those that apply to planning the file-owning regions: These are:

- **How many regions?**
- **Capacity planning**
- **Availability**
- **Security.**

For information about these topics, see “Planning the File-Owning Regions” on page 95.

The additional factors you should take into account when planning to create one or more queue-owning regions are as follows:

1. Naming conventions for remote temporary storage queues
2. Using transient data in a queue-owning region
3. Data integrity considerations.

These are described in the following sections.

### **Naming Conventions for Remote Temporary Storage Queues**

There are two types of temporary storage queue names:

- Names that are generated dynamically by application programs
- Names that are fixed (precoded within an application program).

To define a queue as remote you must include an entry for the queue in a temporary storage table (TST). The TST naming convention allows for dynamic names by accepting generic names formed by a constant prefix, to which a CICS application program can add a variable suffix.

Generic names are formed from the leading characters of the 8-character queue names and can be up to 7 characters long. (Names in a TST entry using all 8 characters are specific temporary storage queue names.)

The usual convention for dynamically naming temporary storage queues is to use a 4-character prefix (for example, the transaction identifier) followed by a 4-character terminal identifier as the suffix. This generates queue names that are unique for a given terminal, and which can be simply defined in a TST by specifying the DATAID parameter. To define them as remote queues, you also need to specify the name of the remote system on the SYSIDNT parameter.

If your naming convention for dynamically named queues does not conform to a generic naming convention such as that just described, the queue may not be capable of being defined as remote, and all transactions that access the queue must be routed to the application-owning region where the queue was created. Furthermore, the CICS/ESA pre-Version 4 temporary storage global user exit points XTSREQ, XTSIN, XTSOUT do not allow you to modify queue names.

CICS/ESA 4.1, however, provides an additional global user exit point, XTSEREQ, that allows you to modify the temporary storage queue name *before* CICS performs TST lookup. You can write a global user exit program for this exit point, to change the name to one that conforms to remote naming conventions. For example, it could transpose the first 4 and last 4 characters if the queue name has a constant suffix rather than a constant prefix.

See Figure 25 for an illustration of the use of a remote queue-owning region.

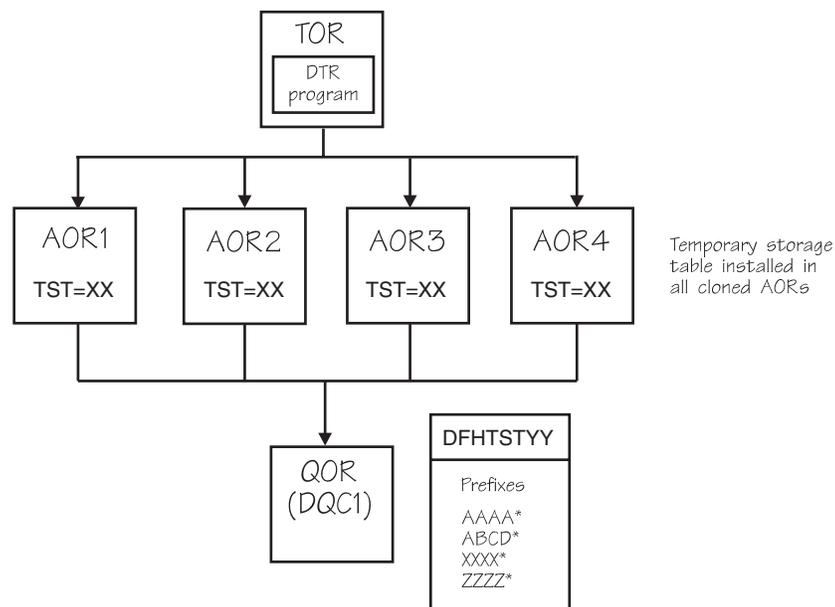


Figure 25. Using Remote Queues to Avoid Inter-transaction Affinity Relating to Temporary Storage. In the TST defined for the AORs (suffix=XX), all the TS queues are remote, defined as owned by the queue-owning region (SYSID=DQC1). The TST defined for the queue-owning region (suffix=YY), defines which queues are recoverable.

**Note:** Using a transaction's task number is not recommended as a way of creating unique queue names, because they are not guaranteed to be unique across the CICSplex.

**Exception Conditions for Globally Accessible Queues:** If you eliminate inter-transaction affinity relating to temporary storage queues by the use of a global queue-owning region, you must also be sure to review exception condition handling. This is because some exception conditions can occur that were not possible when the transactions and the queue were local in the same region. These situations arise because the application-owning region and queue-owning region can fail independently, causing circumstances where:

- The queue already exists, because only the AOR failed while the queue-owning region continued
- The queue is not found, because only the queue-owning region failed while the application-owning region continued.

### Using Transient Data in a Queue-Owning Region

Another form of data queue that CICS application programs commonly use is the transient data (TD) queue.

The dynamic transaction routing considerations for TD queues have much in common with those for temporary storage, and, like temporary storage, not all transient data queues need to be shared. Some queues can be locally owned (when there is no question of inter-transaction affinity), or queues can be replicated.

To enable transactions sharing a TD queue to be dynamically routed to a set of application-owning regions, you must ensure that the TD queues are globally accessible to those application-owning regions.

All TD queues must be defined as remote entries in a destination control table (DCT).

However, there is a restriction for TD queues that use the trigger function (see the *CICS Application Programming Guide* for information about automatic transaction initiation (ATI) and trigger levels). The transaction to be invoked when the trigger level is reached must be defined as a local transaction in the region where the queue resides (in the queue-owning region). Thus the trigger transaction must execute in the queue-owning region. However, any terminal associated with the queue (specified on the DESTFAC parameter) need not be defined as a local terminal in the queue-owning region. This does not create an inter-transaction affinity.

**Note:** Trigger-level transactions that are associated with a terminal must be defined in the terminal-owning regions as remote transactions that are statically-routed. This also means that you need connectivity between all terminal-owning regions and a global queue-owning region.

See Figure 26 on page 105 for an illustration of the use of a remote transient-data queue-owning region.

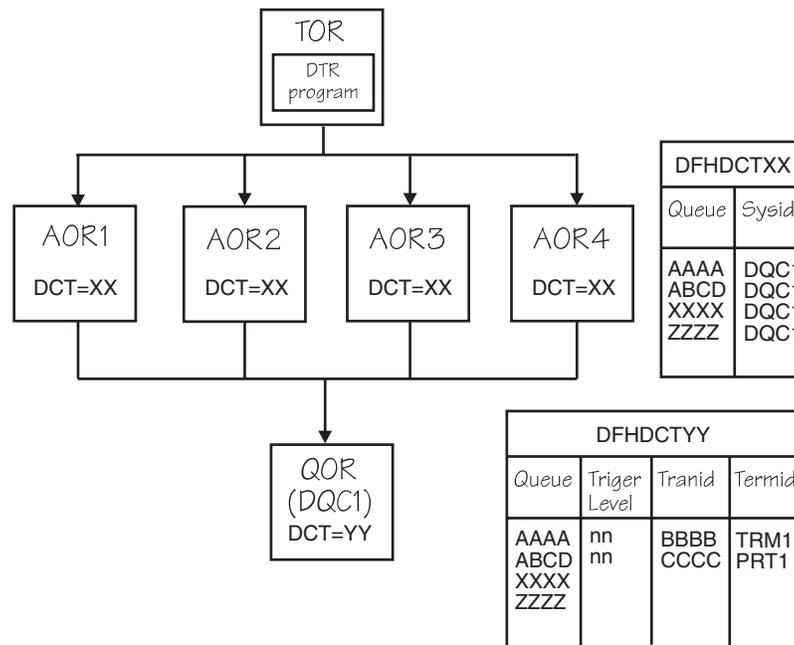


Figure 26. Using Remote Queues to Avoid Inter-transaction Affinity Relating to Transient Data. In the DCT defined for the AORs (suffix=XX), all the TD queues are remote, defined as owned by the queue-owning region (SYSID=DQC1). In the DCT defined for the queue-owning region (suffix=YY), all the TD queues are local, some with trigger levels.

**Exception Conditions for Globally Accessible Queues:** When you eliminate inter-transaction affinity relating to TD queues, by the use of a global queue-owning region, there should not be any new exception conditions (other than SYSIDERR if there is a system definition error or failure).

**Special Considerations for Trigger-Level Transactions:** The transaction defined in a transient data queue definition that specifies the TRIGLEV attribute must be defined as a local transaction in the queue-owning region, otherwise transaction initiation fails.

This means that all such triggered transactions run in the queue-owning region. Thus, the user application programs invoked by the trigger transactions run in the queue-owning region, and the resources required by these programs must be made available to the queue-owning region. Hence it is, in the general case, necessary to provide connectivity from the QOR to the other subsystems (file-owning regions, other queue-owning regions, DBCTL, and DB2) that such programs may require.

If the triggered transaction is associated with a terminal then there must be connectivity between the queue-owning region and the terminal-owning region that owns the terminal. The queue-owning region must also be able to associate the terminal with the correct terminal-owning region. Since the terminal id is specified in the TD queue definition, it is probably sensible to predefine it in the queue-owning region as a remote terminal owned by one of the terminal-owning regions. Alternatively, you could use a “terminal not found” global user exit program to nominate the terminal-owning region that owns the terminal.

In addition, if the triggered transaction has any inter-transaction affinity with other transactions, those transactions must also run in the QOR. You can ensure this happens by defining the other transactions as statically-routed remote transactions in each of the terminal-owning regions, and defining connections between each terminal-owning region and the queue-owning region.

## Data Integrity Considerations

The data integrity considerations for queue-owning regions are very much the same as for file-owning regions.

When planning to manage queues as a remote resource owned by a queue-owning region, you must consider the recovery implications for data integrity.

Temporary storage queues are CICS resources that can be defined as recoverable resources.

The data integrity factor you have to consider when planning the number of queue-owning regions in your CICSplex is the in doubt period.

As stated in “Planning the File-Owning Regions” on page 95, the possibility of in-doubt situations means there is a potential risk to data integrity when recoverable resources are owned by more than one CICS region. This is true if the recoverable resources owned by the different regions are of different types; that is, a mixture of VSAM files and temporary storage queues. For the present, in CICS/ESA 4.1, there is no automatic recovery or protection from the in-doubt situation during recovery processing as in a CICS emergency restart.

Therefore, when planning your MRO configuration to obtain the maximum benefit from the range of facilities offered by the sysplex environment, you must carefully study how your applications use resources such as temporary storage queues. Consider the following questions:

- Are the queues defined as recoverable? If not, there are no data integrity concerns.
- If some queues are defined as recoverable resources, are they accessed within the same unit-of-work as other recoverable resources, so as to cause a potential data integrity exposure?

### Allocating ownership of recoverable resources

As a general rule, you should ensure that all recoverable resources that are updated by one logical unit of work are owned by the same CICS region. If the resources are of different types, such as VSAM data and temporary storage queues, allocate them to a common resource-owning region.

See “Planning the File-Owning Regions” on page 95 for more information about data integrity considerations.

## Performance and Availability

You might also want to consider performance benefits, as well as data integrity, when deciding whether to have separate or combined resource-owning regions. Operating a combined file-owning region/queue-owning region for a given application’s resources offers performance advantages compared with splitting them into separate region types.

There are also availability advantages in having different combined terminal-owning regions/queue-owning regions owning the resources for different applications.



---

## Chapter 11. Planning for IMS DBCTL Multisystem Data Sharing with CICS

This chapter describes how to enable multisystem data sharing for CICS transactions that access IMS full-function and Fast Path databases managed by IMS Database Control (DBCTL). IMS data sharing allows multiple subsystems within a sysplex to have concurrent access, with data integrity, to IMS databases.

Within a sysplex environment, the subsystems can be:

- A number of IMS DBCTL subsystems
- A number of IMS DB online subsystems

This chapter covers the following topics:

- Migrating from CICS local DL/I to CICS with IMS DBCTL.  
If you are currently using IMS databases through the CICS local DL/I interface, you must first migrate to DBCTL.
- Creating multiple IMS DBCTL subsystems.  
If you are already using IMS DBCTL to provide access to the IMS databases, you need to consider replicating the DBCTL subsystems.
- Converting CICS shared database programs to batch message processing (BMP) programs.
- Creating the data sharing environment.  
This involves planning to use the IMS database recovery control (DBRC) together with the IMS resource lock manager (IRLM) subsystem, and the coupling facility.

### Further information

For general information about migrating to, and using, IMS DBCTL with CICS, see the *CICS IMS Database Control Guide*, and the relevant IMS/ESA publications.

---

## Migrating from CICS Local DL/I to IMS DBCTL

You cannot use IMS multisystem data sharing with CICS local DL/I. To enable all the CICS application-owning regions shown in our target configuration to access the IMS databases, you must migrate the databases from CICS ownership to IMS DBCTL ownership.

### When Databases Need to be Migrated

You will need to migrate the IMS databases if they are owned directly by a CICS application-owning region that needs access to the data, or owned by a CICS file-owning region to which multiple application-owning regions function ship their DL/I requests (as illustrated in Figure 27 and Figure 28).

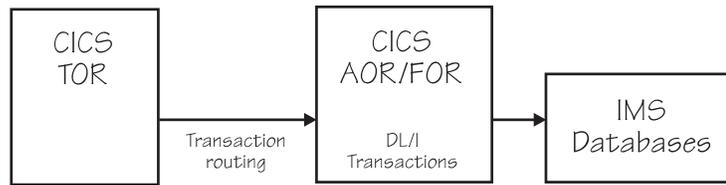


Figure 27. Local DL/I Access Direct from an Application-Owning Region

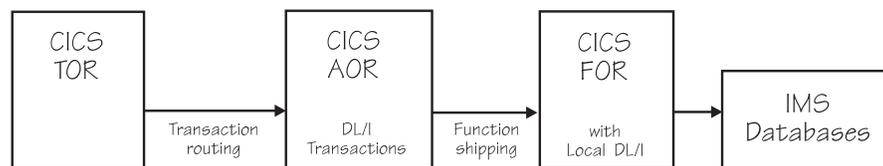


Figure 28. DL/I Access Through Function Shipping to a File-Owning Region

If you are using either of the CICS local DL/I methods shown in Figure 27 and Figure 28, you should change to the system configuration shown in Figure 29.

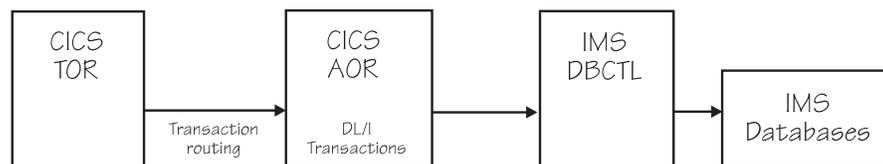


Figure 29. CICS access to IMS databases through the IMS DBCTL interface

## Migrating CICS Parameters to IMS DBCTL

When you migrate to IMS DBCTL, you control some DL/I functions differently from the CICS local DL/I interface:

- To ensure that CICS does not attempt to initialize the CICS local DL/I interface, you should modify your CICS system initialization parameters and specify DLI=NO. If you use the default system initialization table, as recommended in Chapter 14. Planning the Resource Definitions, DLI=NO is the default. All other system initialization parameters that relate to CICS local DL/I are ignored when you specify DLI=NO.
- When you migrate to IMS DBCTL, some of the local DL/I control functions specified by CICS system initialization parameters are replaced by parameters in:
  - The DBCTL system generation macros
  - The DRA startup parameter table (DFSPZPxx)
  - The DBCTL startup JCL.

For general information about these topics, see the *CICS IMS Database Control Guide*.

### Defining the Number of DL/I Threads

When you are using the CICS local DL/I interface, you use the DLTHRED system initialization parameter to define the number of DL/I threads you want CICS to provide. When you migrate to DBCTL, you define the number of threads in the DBCTL stage 1 system generation macros and in the DRA startup parameter table. In the sysplex environment, where you are cloning terminal-owning regions, application-owning regions, and the DBCTL subsystems, you should reduce the number of threads you request for each DBCTL. You should balance the number of threads in each DBCTL against the number of transactions that you anticipate will be processed by the application-owning regions attached to each DBCTL. Note, however, that you cannot similarly reduce the buffers in each DBCTL to the same extent.

For more information about migrating from CICS local DL/I to DBCTL, see the *CICS IMS Database Control Guide*.

---

## Creating Multiple DBCTL Subsystems

You need to create one DBCTL in each MVS image that has CICS application-owning regions needing access to the IMS databases. To clone DBCTL subsystems, use the same DBCTL system generation options and the same startup options.

## The CICS Database Resource Adapter Startup Table

The database resource adapter (DRA) startup parameter table is called DFSPZPxx, where xx is a unique suffix that you define to identify each version of a DRA table.

Each DRA table contains the unique name of the DBCTL subsystem, specified on the DBCTLID parameter, which is used by CICS to identify the DBCTL to which it can connect. In CICS/ESA Version 3, you can specify the DBCTLID **only** in the DRA table, which means you have to generate a DRA startup parameter table for each DBCTL in the sysplex. CICS/ESA 4.1 allows you to override the DBCTLID with the CICS system initialization parameter, INITPARM.

The support in CICS/ESA 4.1 for a generic DRA table enables you to generate one table for the whole sysplex environment, which you can then use to clone DBCTL subsystems, overriding the DBCTLID during CICS startup.

The syntax for the INITPARM parameter string for DBCTL is as follows:

```
INITPARM=(DFHDBCON='suffix,dbctlid')
```

where:

*suffix* is the 2-character alphanumeric DRA table suffix and  
*dbctlid* is the 4-character DBCTL name

## Naming the DBCTL Subsystems

To identify the different DBCTL subsystems with unique names, and yet be able to recognize easily which CICS application-owning regions they support, you are recommended to use a naming convention such as the one described in “Chapter 3. Planning Naming Conventions for CICS and Related Subsystems” on page 27. Using the same naming conventions for DBCTL as for CICS makes it easy to identify subsystem affiliations. If you apply the naming convention to our target

configuration (see Figure 1 on page 18) where we show the three CICS application-owning regions in MVSA as DAA1, DAA2, and DAA3, the associated DBCTL becomes DDA1.

---

## Converting a CICS Shared-Database Program to a BMP Program

Shared database programs can be run only in the local DL/I environment. However, you can convert shared database programs to IMS batch-oriented batch message processing (BMP) programs that can then run in the IMS DBCTL environment.

Advantages to converting include:

- Better performance, because BMP programs communicate directly with DBCTL instead of accessing databases through CICS.
- The ability to use IMS system service requests, such as symbolic checkpoint (CHKP) and extended restart (XRST).
- Access to GSAM databases. It is better to use GSAM databases instead of sequential MVS files because, during extended restart the GSAM databases are repositioned at the record being processed when the checkpoint was written. GSAM can use BSAM on direct access, unit record, and tape devices, and VSAM entry sequenced data sets (ESDSs).
- Logging to the IMS log, rather than the CICS logs.
- Access to IMS DEDBs.

**Note:** An I/O PCB is always present for a BMP program. For a batch program, you can obtain an I/O PCB by specifying the compatibility option in the IMS program specification block (PSB) for the program. For more information on the compatibility option in the PSB, see *IMS/ESA Utilities Reference: System*.

---

## Creating the Data Sharing Environment

IMS multisystem data sharing in the sysplex environment requires the use of two IMS facilities:

- IMS Database Recovery Control (DBRC)
- Internal resource lock manager (IRLM).

You also need the coupling facility.

### Database Recovery Control

If you do not currently use DBRC, you must plan for its use in the sysplex environment.

DBRC runs in its own address space, and is a mandatory component of any DBCTL subsystem. You specify DBRC in the IMS DB stage 1 system definition macros.

DBRC is started automatically by DBCTL to control backup, recovery, and access for all IMS databases that are registered to DBRC. These functions are essential for data sharing.

DBRC automatically records recovery data on a duplicated recovery control data set, known as the RECON data set. It also generates and validates the JCL for backup and recovery jobs using IMS database utilities.

All databases that participate in data sharing must be *registered* to DBRC by name and sharelevel. See the *IMS/ESA Operations Guide* for information about registering databases.

## Internal Resource Lock Manager (IRLM)

If you do not currently use IRLM, you must plan for its use in the sysplex environment. IRLM is a resource-locking component of IMS that runs in its own address space in each MVS image in an IMS data sharing environment. You need one IRLM per MVS image that contains an IMS subsystem. IRLM is mandatory for IMS record-level data sharing. IRLM does the following:

- Takes over as lock manager from the program isolation lock manager
- Maintains the integrity of data shared between multiple subsystems
- Acts as a vehicle for the notification of events such as data set extensions, buffer invalidation (see note), and database backout failures
- Provides support for global commands.

**Note:** Buffer invalidation is a technique used by IRLM and IMS data sharing subsystems in a sysplex environment. Buffer invalidation indicates that data held in a buffer by members of a data sharing group has become invalid as a result of an update performed by a member of the data sharing group.

See *IMS/ESA Installation Volume 1: Installation and Verification* for more information about IRLM.

## Defining IMS Coupling Facility Structures

The coupling facility must be installed and defined to MVS before you can use IMS with data sharing. IMS has three coupling facility structures:

- An IRLM lock table
- An OSAM data cross-invalidation cache
- A VSAM data cross-invalidation cache.

You use coupling facility resource management (CFRM) policies to define coupling facility structures. A single CFRM policy controls how and where the structure resources are allocated. See the *z/OS MVS Setting Up a Sysplex* manual for information about defining a coupling facility policy and the IMS library for details on the size and definition of the three IMS structures.

### Rebuilding IMS Coupling Facility Structures after Failure

If the coupling facility fails and all structures are lost, IMS automatically rebuilds the structures and resumes data sharing.



---

## Chapter 12. Planning for DB2 Subsystem Access from CICS Regions

This chapter discusses some considerations for DB2 data sharing with CICS.

For planning information for DB2 data sharing, see *DB2 for MVS/ESA Version 4 Data Sharing: Planning and Administration*, SC26-3269.

---

### Creating Multiple DB2 Data Sharing Subsystems

DB2 Version 4 supports data sharing across the sysplex. DB2 data sharing enables CICS transactions running in any CICS application-owning region to access the same shared DB2 data.

In the configuration shown in Figure 1 on page 18, we have installed a DB2 subsystem, each with its own IRLM, in each of the MVS images in which there are CICS application-owning regions (MVSA—MVSD). To enable access to shared DB2 data, each AOR must connect with the DB2 that resides in the same MVS image.

Each DB2 that shares data must belong to a DB2 data sharing group, which is a collection of one or more DB2 subsystems accessing shared DB2 data. All members of the data sharing group use the same shared DB2 catalog and directory.

### Using a Common Resource Control Table

You can simplify the task of connecting CICS regions to DB2 subsystems that are members of a DB2 data sharing group by using a common resource control table (RCT). With only one RCT, you uniquely identify the DB2 member to which a CICS region is connecting by specifying the DB2 member subsystem name when you start the CICS-DB2 connection. You can do this in one of two ways. In each example, D# identifies the RCT, and DBA1 identifies the DB2 data sharing group member that runs in MVSA: (Figure 1 on page 18 shows you the configuration with MVSA.)

1. Specify the suffix of the common RCT and the DB2 member subsystem name on the CICS DB2 start command (DSNC). For example:  

```
DSNC STRT D#,DBA1
```
2. Specify the DB2 program, DSN2COM0, as a CICS initialization program in a program list table (PLT), and pass the required parameters to the DB2 program, DSN2STRT, in the CICS system initialization parameter, INITPARM. For example:  

```
INITPARM=(DSN2STRT='D#,DBA1')
```

### Naming the DB2 Subsystems

To uniquely identify the different DB2 subsystems, and also be able to recognize easily which CICS application-owning regions they are connected to, use a naming convention such as the one described in “Chapter 3. Planning Naming Conventions for CICS and Related Subsystems” on page 27. Using the same naming conventions for DB2 as for CICS makes it easy to identify subsystem affiliations. It also helps when using facilities such as the MVS automatic restart manager to identify those subsystems that have an affinity with each other.

If you apply the naming convention to our target configuration (see Figure 1 on page 18 ) where we show the three CICS application-owning regions in MVSA as DAA1, DAA2, and DAA3, the associated DB2 becomes DBA1.

---

## Creating the Data Sharing Environment

DB2 data sharing in a sysplex requires:

- Coupling facility structures
- Internal resource lock manager (IRLM)

## Defining DB2 Coupling Facility Structures

At least one coupling facility must be installed and defined to MVS before you can use DB2 data sharing. DB2 uses these coupling facility structures:

- A single lock structure for each data sharing group, used to control locking.
- A single list structure for each data sharing group, used for intra-group communications and for storing status information about databases. DB2 calls this structure the *shared communications area (SCA)*.
- Multiple cache structures for each data sharing group, used for caching and cross-invalidating data. DB2 calls these *group buffer pools*.

Use the MVS coupling facility resource management (CFRM) policies to define these structures to the sysplex. A CFRM policy determines how and where the structure resources are allocated. See *z/OS MVS Setting Up a Sysplex* for information about how to create a CFRM policy.

For information about how to name the structures and how to estimate their sizes, see *DB2 for MVS/ESA Version 4 Data Sharing: Planning and Administration*.

### Rebuilding DB2 Coupling Facility Structures after Failure

If a coupling facility fails, the SCA and lock structure are automatically rebuilt on another coupling facility, assuming you have made provisions for automatic rebuild on another coupling facility. When these crucial structures are rebuilt, data sharing continues.

DB2 does not rebuild group buffer pools; instead, DB2 uses information from the DB2 logs to recover the data that was in the failed group buffer pool.

In the meantime, MVS reallocates the group buffer pool structure in an active coupling facility so that data sharing can continue.

## Internal Resource Lock Manager (IRLM)

IRLM is the resource-locking component of DB2 that runs in its own address space. Each DB2 must have its own IRLM.

IRLM controls global locking of data shared between multiple DB2s in the data sharing group. It interacts with the coupling facility to ensure that a transaction lock used by one DB2 is not in conflict with a transaction lock used by another DB2.

IRLM also controls the information that DB2 needs for *buffer coherency*, the mechanism by which DB2 ensures that an application never reads down-level data.

For more information about installing and tuning IRLM, see *DB2 for MVS/ESA Version 4 Data Sharing: Planning and Administration*.

---

## Chapter 13. Planning the Log Streams

In CICS/ESA 4.1 and earlier, CICS logging and journaling is performed by the CICS journal control management function, which uses sequential data sets supported by an automatic archiving facility. In CICS 5.1, the earlier journal control management function is replaced by the CICS log manager which, instead of sequential data sets, uses the MVS system logger for all its logging and journaling requirements. Using services provided by the MVS system logger, the CICS log manager supports:

- The CICS system log, which is used for transaction backout, emergency restart, and preserving information for resynchronizing in-doubt units-of-work, even on a cold start. There is no internal dynamic log in CICS 5.1 as in earlier releases of CICS—the system log is used for *all* transaction backout.
- Forward recovery logs, auto-journals, user journals, and a log of logs. These are collectively referred to as general logs to distinguish them from system logs.

Migrating to CICS 5.1 requires careful planning to define all your existing CICS logging and journal requirements to the MVS system logger and CICS log manager.

**Note:** For CICS logging, you can use either the coupling facility, DASD-only, or both the coupling facility and DASD. This book describes the MVS system logger and the use of the coupling facility for CICS. For DASD-only logging, you must have OS/390 V2 R4 or higher installed. For information on DASD-only logging, see *z/OS MVS Setting Up a Sysplex*.

---

### The MVS System Logger and the CICS Log Manager

Before planning the migration to the logger environment, it is important to understand the facilities supported by the MVS system logger and the CICS log manager, and how these have replaced those supported by the obsolete CICS journal control program.

#### Log streams

A sequence of data blocks, with each log stream identified by its own log stream identifier—the log stream name (LSN). Each log stream is written by the MVS system logger to a specified list structure in the coupling facility.

The CICS system log, forward recovery logs, autojournals, log of logs, and user journals map onto specific MVS log streams.

MVS log streams replace the CICS journal data sets.

#### Log stream data sets

Auxiliary storage on DASD for log streams. When a coupling facility list structure is filled to its high offload threshold point or beyond, the MVS system logger begins offloading data from the coupling facility to the DASD log stream data sets.

See the *z/OS MVS Setting Up a Sysplex* for information about the offload threshold parameter.

#### LOGR couple data set

The couple data set that holds the MVS system logger policy information, accessible by all the systems in the sysplex.

Defining the LOGR data set and its policy information is one of the first tasks you must complete when preparing to migrate to an MVS system logger

environment. See the *z/OS MVS Setting Up a Sysplex* for information about the LOGR couple data set and CFRM policies.

### **Coupling facility list structure definitions**

These define all the list structures used by the MVS system logger, that are referenced by log stream definitions. See the *z/OS MVS Setting Up a Sysplex* for information about the LOGR couple data set and CFRM policies.

### **Log stream definitions**

These define all the log streams, including log stream models, that are referenced by the CICS log manager. Log streams are written to a specified list structure within coupling facility.

You define log streams using the MVS administrative data utility, IXCMIAPU. See the *z/OS MVS Setting Up a Sysplex* for information about the IXCMIAPU utility.

### **Journal names**

In general, CICS regions refer to their system logs and general logs by an internal 8-character journal name. For example, the internal name for the CICS system log is DFHLOG, and general logs are identified by journal names of the form DFHJnn, where nn is a number in the range 01 through 99. CICS maps its internal 8-character journal names to MVS log streams by user-defined JOURNALMODEL resource definitions, in which you specify the journal name and the corresponding log stream name.

The JOURNALMODEL resource definition replaces entries in the journal control table, which is obsolete.

An exception to the general rule is where the fully-qualified 44-character log stream name for a VSAM forward recovery log is obtained directly from the ICF catalog.

### **JOURNALMODEL resource definitions**

These define the journal names and their corresponding log stream names. You define these in the CICS system definition (CSD) data set.

The use of these resource definitions is optional. If you don't specify journal models, CICS constructs default log stream names using its region userid, applid, and the journal name.

## **Coupling Facility Requirements for the System Logger**

The MVS system logger requires a coupling facility running coupling facility control code (CFCC) at service level 4.03 (CFLEVEL=1) or higher.

For information about the coupling facility and CFLEVEL, see *PR/SM Planning Guide*.

---

## **Defining the logger environment for CICS**

To plan your migration to the CICS log manager, you need the following publications:

*z/OS MVS Setting Up a Sysplex*, SA22-7625

This book provides information on how to complete the following tasks:

1. Defining the LOGR couple data set and making it available to the sysplex.
2. Defining the CFRM LOGR policy.
3. Defining coupling facility list structures, with each list structure designed to hold one or more log streams.

4. Defining explicit log streams, and log stream models for reference by CICS regions when creating log streams dynamically.

*CICS Transaction Server for OS/390 Installation Guide, GC34-5697*

This book provides information on the following topics:

- **Requirements.** It describes the hardware and software requirements for the CICS log manager.
- **Setting up the environment for the CICS log manager.** It outlines the tasks to be performed and the order in which they must be completed. Where appropriate, this topic refers to the *z/OS MVS Setting Up a Sysplex*.
- **Planning guidelines for CICS log streams.** It includes information about coupling facilities and defining coupling facility structures for log streams. The *CICS Transaction Server for OS/390 Installation Guide* also provides sample JCL to define log stream structures using the IXCMIAPU administrative data utility.
- **Calculating log structure parameter values.** CICS provides a utility program, DFHLSCU, to help you calculate values for the AVGBUFSIZE, INITSIZE, SIZE, and STG\_SIZE parameters:
  - AVGBUFSIZE is a parameter on the DEFINE STRUCTURE specification
  - SIZE and INITSIZE are parameters on the IXCMIAPU CFRM policy specification
  - STG\_SIZE is a parameter on the DEFINE LOGSTREAM specification.

Where the DFHLSCU utility is not appropriate, the *CICS Transaction Server for OS/390 Installation Guide* also provides a number of formulae to help you calculate parameter values.

- **Naming conventions.** Log structure naming conventions are recommended for CICS log streams.
- **Staging data sets.** Recommendations for the use of staging data sets for CICS log streams.

*CICS System Definition Guide, SC34-5725*

This book provides information about:

- Defining CICS system logs
- Defining CICS general log journals
- Journal naming
- How CICS logs and journals map onto MVS system logger log streams
- The CICS journal utility program, DFHJUP

*CICS Operations and Utilities Guide, SC34-5717*

This book provides information on how to use the DFHLSCU utility program.



---

## Chapter 14. Planning the Resource Definitions

This chapter gives examples of some of the key resource definitions needed to support dynamic transaction routing (DTR) across a sysplex. (These examples are based on the naming convention described in “The Naming Convention” on page 31.) It also discusses standard procedures and techniques for cloning CICS regions in a sysplex environment. The main topics are:

- **Defining remote attributes for transaction routing**
  - Defining transactions for static transaction routing
  - Defining transactions for dynamic transaction routing
- **Defining connections and sessions for the MRO links**
  - TOR—AOR links
  - AOR—FOR links
  - AOR—QOR links

When setting up an MRO transaction routing environment, one of the first steps is to decide on the type of transaction routing that you want for each of your transactions. Remote attributes must be specified for each transaction that is to be routed to a remote system by a TOR. You must decide whether transactions are to be dynamically routed by a dynamic transaction routing program, or statically routed according to the resource definition attributes.

Static routing is the easier to implement, and does not require a dynamic transaction routing program, or any of the other supporting features of a dynamic transaction routing environment. However, static transaction routing precludes the workload balancing and high-availability benefits you can obtain by being able to choose the target AOR dynamically, and you are recommended to specify the dynamic option whenever possible.

---

### Defining Remote Attributes for Transaction Routing

The attributes of a transaction resource definition that control transaction routing are shown on the CEDA DEFINE panel under the heading “REMOTE ATTRIBUTES” (see Figure 30 on page 123 for an example).

If you specify DYNAMIC(NO), CICS routes the transaction to the AOR specified on the REMOTESYSTEM parameter.

**Note:** CICS also assumes dynamic routing by default if you omit a transaction definition. In this case, CICS uses the attributes of a default DTR definition whose transaction id you specify on the DTRTRAN system initialization parameter, and which defaults to the CICS-supplied definition, CRTX, if you omit the system initialization parameter.

If you specify DYNAMIC(YES) on a transaction resource definition, CICS routes the transaction to the AOR dynamically selected by a dynamic transaction routing program.

### Defining Transactions for Static Transaction Routing

To define a transaction for static transaction routing, specify the remote attributes as follows:

**DYNAMIC**

Omit this parameter for static routing—it takes the value NO by default.

## REMOTESYSTEM

Specify the name of the remote CICS region to which the TOR is to route the transaction.

Note that when you specify a remote system name for use in the TOR, you can use the same definition in the AOR provided you also specify a program name for the transaction. (A program name is not required for a remote-only transaction definition.) A definition that can be used for both local and remote purposes is described as a dual-purpose resource definition, which is explained in the *CICS Resource Definition Guide*. When CICS installs a dual-purpose transaction definition in both the terminal-owning region and the application-owning region, CICS compares its own system name (taken from the SYSIDNT system initialization parameter) with the REMOTESYSTEM name in the definition, with the following result:

- If the names are different, CICS creates a remote transaction definition.
- If the names are the same, CICS creates a local transaction definition.

## REMOTENAME

Specify a remote name on this parameter when the transaction is known by a different name in the remote system.

If you don't specify a remote name, CICS assumes the transaction is known by the same name in the local and remote regions, and the parameter defaults to the name of the local transaction.

The remainder of the remote attributes can be defined for both static and dynamic transaction routing, and are described in the *CICS Resource Definition Guide*.

An example of a completed CEDA panel for a static transaction routing definition is shown in Figure 30 on page 123.

## Recommendations

In general, you should use static routing only for those transactions that are not suitable candidates for dynamic transaction routing. For example, if you have transactions that create global intertransaction affinity with a permanent lifetime, or transactions that have transaction-system affinity, you are recommended to choose static routing. In the interests of workload balancing, you should aim to keep static routing to a minimum.

See the *CICS Application Programming Guide* manual for information about the constraints imposed by transaction affinities in a dynamic transaction routing environment.

```

OBJECT CHARACTERISTICS                                CICS RELEASE = 0330
CEDA View
Transaction    : AC20
Group         : DFH$CTXT
Description    : Display PRIMARY PANEL of the CICS CUA Sample Application
PROGram      : DFH0VT1
.            :
.            :
REMOTE ATTRIBUTES
DYNAMIC       : NO                               No | Yes
REMOTESystem  : CICA
REMOTENAME    : AC20
TRProf       : DFHCICSS
Localq       :                               No | Yes
SCHEDULING
.            : 001
ALIASES
Alias        : ac20
.            :

APPLID=CICSTOR

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Figure 30. A Remote Transaction Resource Definition that Specifies Static Routing

## Defining Transactions for Dynamic Transaction Routing

There are two methods of indicating to CICS that a transaction is to be dynamically routed.

1. Omitting specific transaction definitions altogether from the terminal-owning regions—this is the recommended method.
2. Specifying dynamic on specific transaction definitions—supported for compatibility with earlier releases.

### No Transaction Definitions in the Terminal-Owning Regions

The easiest method of specifying dynamic transaction routing in a TOR is to omit specific transaction resource definitions altogether.

When CICS receives a transaction that it cannot find in the table of installed transaction resource definitions, CICS uses a special DTR resource definition. You can specify the name of this dynamic transaction routing definition on the DTRTRAN system initialization parameter, which defaults to the CICS-supplied transaction, CRTX.

In the absence of a specific definition for a transaction, CICS attaches the transaction using the attributes from the DTRTRAN-specified transaction, and calls the dynamic transaction routing program to determine to which remote system the transaction is to be routed.

The dynamic transaction routing program can choose to police the transactions itself and for unknown transactions could simply return control to CICS, in which case CICS rejects the transaction as unknown. A better approach might be to let the dynamic transaction routing program simply route the transaction to any one of the cloned application-owning regions, and let the application-owning region reject the transaction as unknown. In each case the terminal operator sees the standard CICS message reporting that 'Transaction ... is unrecognized'.

Omitting specific transaction definitions from the terminal-owning regions is the recommended method for dynamic transaction routing. It reduces system administration effort in maintaining transaction resource definitions, and significantly reduces the size of the transaction table needed in terminal-owning regions. It not only removes the need to define new transactions for dynamic transaction routing, it avoids the need to modify all your existing definitions.

**Note:** See “Chapter 15. Planning CICSplex Security” on page 137 for information about transaction attach security checks for transactions attached using the DTRTAN transaction.

For general information dynamic transaction routing see the *CICS Intercommunication Guide*.

For information about writing a dynamic transaction routing program, see the *CICS Customization Guide*.

### **Defining specific transaction definitions**

If you have specific transaction definitions that you use in your terminal-owning regions, you can continue to use them.

If you want to use these for dynamic transaction routing, specify the remote attributes as follows:

#### **DYNAMIC**

Specify DYNAMIC(YES) for dynamic transaction routing.

#### **REMOTESYSTEM**

Leave this blank unless you want to specify a default remote system.

Normally, with DYNAMIC(YES) defined, the dynamic transaction routing program selects the remote system to which CICS is to route the transaction. However, CICS always passes to the dynamic transaction routing program (via its COMMAREA) a default remote system name. If you leave the remote system name blank, CICS passes its own SYSIDNT value as the default “remote” name, which means that, by default, the transaction runs locally.

#### **REMOTENAME**

Specify a remote name for the transaction on this parameter when the transaction is known by a different name in the remote system.

If you don’t specify a remote transaction name, CICS assumes the transaction is known by the same name in the local and remote regions, and the parameter defaults to the name of the local transaction. The remote name is passed to the dynamic transaction routing program (via the COMMAREA), and can be varied dynamically by the dynamic transaction routing program if the transaction is known by a different name in the remote system.

#### **PROGRAM**

You must specify a program name for a DYNAMIC(YES) transaction. This ensures that, if the transaction is not dynamically routed and CICS has to run it locally, CICS knows which application program to invoke.

See Figure 31 on page 125 for an example of a completed CEDA panel for a dynamic transaction routing definition.

```

OBJECT CHARACTERISTICS                                CICS RELEASE = 0330
CEDA View
Transaction    : AC20
Group          : DFH$CTXT
Description    : PRIMARY PANEL of the CICS CUA Sample Application
PROGram       : DFH0VT1
.             :
.             :
REMOTE ATTRIBUTES
Dynamic        : YES                               No | Yes
REMOTESystem  :
REMOTENAME    : AC20
TRProf        : DFHCICSS
Localq        :                               No | Yes
SCHEDULING    : 001
.             :
ALIASES
Alias         : ac20
.             :

APPLID=CICSTOR

PF 1 HELP 2 COM 3 END                6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Figure 31. A Remote Transaction Resource Definition that Specifies Dynamic Routing

## Defining Connection and Session Definitions

The CICSplex in our target configuration shown in Figure 1 on page 18 requires a number of connection and sessions definitions for MRO. These can be kept to a minimum by adopting the naming conventions described in “Chapter 3. Planning Naming Conventions for CICS and Related Subsystems” on page 27.

As a general rule, you need to create one connection definition, with its supporting sessions definitions, for each CICS region that is linked-to by another CICS region. You can then install that same definition in all the CICS regions that need to link to the CICS region defined by the connection.

By carefully grouping resource definitions, you can install the same CSD groups of definitions in many CICS regions.

The required resource definitions are summarized in the following sections.

**Note:** In the examples of SESSIONS resource definitions that follow, we show how to specify the session names using the old send and receive prefix method, based on the naming recommendations in “Chapter 3. Planning Naming Conventions for CICS and Related Subsystems” on page 27. The recommended way is to omit these prefixes and let CICS allocate them dynamically.

See the *CICS Resource Definition Guide* for information about defining CONNECTION and SESSIONS resource definitions.

## Links from the Terminal-Owning Regions

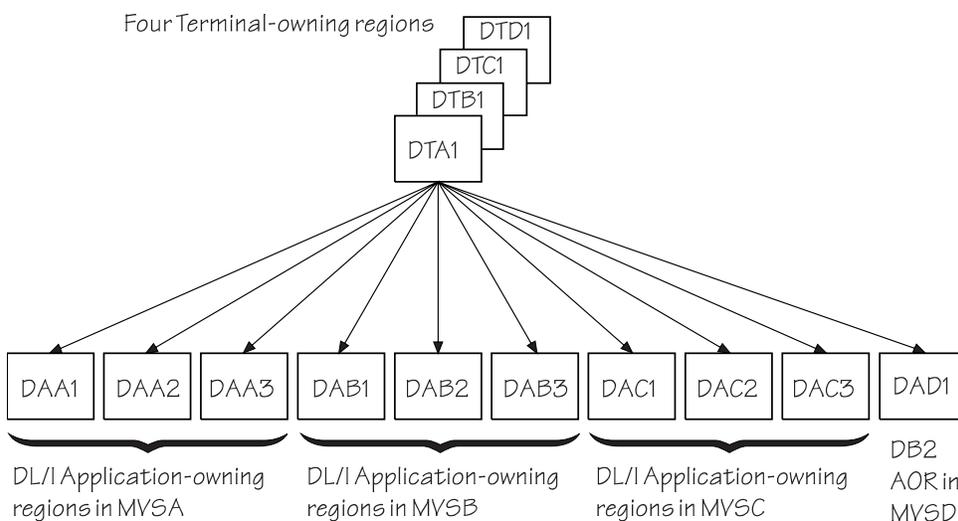
In our target CICSplex, each terminal-owning region requires an MRO connection definition, with supporting sessions, for each application-owning region—a total of 12 of each definition. By creating one generic resource group in the CSD to define

all 12 connections and sessions, the same definitions can be installed in each TOR. In this example, we have used the # character as a generic symbol, to create CONNDT## as the CSD group name, where:

- CONN** indicates the resource type (CONNECTIONS and sessions).
- D** identifies the Dallas CICSplex.
- T** indicates the type of region in which the resource is installed (the terminal-owning regions in this case).
- ##** indicates a generic group that can be installed in more than one region.

Using this technique, the group can be installed in any terminal-owning region that requires a connection to the same set of application-owning regions, making it a simple matter to add another terminal-owning region with links to the same set of application-owning regions, or to add the definition required in terminal-owning region when a new application-owning region is added to the CICSplex.

The definitions required for these links (illustrated in Figure 32) are given in Figure 33 on page 127.



*Figure 32. Required Links to the Application-Owning Regions from the Terminal-Owning Regions. There are 12 in total, one to each of the application-owning regions that support transaction routing.*

```

** CONNECTION Definitions from all TORs to the 12 target AORs.
**
DEFINE CONNECTION(DAA1) GROUP(CONNDT##)
    NETNAME(CICSDAA1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAA1 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAA2) GROUP(CONNDT##)
    NETNAME(CICSDAA2) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAA2 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAA3) GROUP(CONNDT##)
    NETNAME(CICSDAA3) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAA3 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAB1) GROUP(CONNDT##)
    NETNAME(CICSDAB1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAB1 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAB2) GROUP(CONNDT##)
    NETNAME(CICSDAB2) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAB2 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAB3) GROUP(CONNDT##)
    NETNAME(CICSDAB3) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAB3 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAC1) GROUP(CONNDT##)
    NETNAME(CICSDAC1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAC1 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAC2) GROUP(CONNDT##)
    NETNAME(CICSDAC2) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAC2 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAC3) GROUP(CONNDT##)
    NETNAME(CICSDAC3) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAC3 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAD1) GROUP(CONNDT##)
    NETNAME(CICSDAD1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAD1 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAD2) GROUP(CONNDT##)
    NETNAME(CICSDAD2) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAD2 from any TOR in the 'D' CICSplex) ...
*
DEFINE CONNECTION(DAD3) GROUP(CONNDT##)
    NETNAME(CICSDAD3) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(MRO link to DAD3 from any TOR in the 'D' CICSplex) ...

```

*Figure 33. Resource Definitions for Links from Terminal-Owning Regions to Application-Owning Regions (Part 1 of 3). These define the links from a terminal-owning region to each of the transaction routing application-owning regions shown in the target configuration in Figure 1 on page 18. The definitions are the same for all the terminal-owning regions, and therefore can be included in the startup group list for each TOR, using the generic CSD GROUP name CONNDT##.*

```

** SESSIONS Definitions from the 4 TORs to the 12 AORs.
**
DEFINE  SESSIONS(DAA1Y1Z1) GROUP(CONNDT##)
        CONNECTION(DAA1) PROTOCOL(LU61)
        SENDPFX(Y1)    RECEIVEPFX(Z1)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAA1) ...
*
DEFINE  SESSIONS(DAA2Y2Z2) GROUP(CONNDT##)
        CONNECTION(DAA2) PROTOCOL(LU61)
        SENDPFX(Y2)    RECEIVEPFX(Z2)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAA2) ...
*
DEFINE  SESSIONS(DAA3Y3Z3) GROUP(CONNDT##)
        CONNECTION(DAA3) PROTOCOL(LU61)
        SENDPFX(Y3)    RECEIVEPFX(Z3)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAA3) ...
*
DEFINE  SESSIONS(DAB1Y4Z4) GROUP(CONNDT##)
        CONNECTION(DAB1) PROTOCOL(LU61)
        SENDPFX(Y4)    RECEIVEPFX(Z4)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAB1) ...
*
DEFINE  SESSIONS(DAB2Y5Z5) GROUP(CONNDT##)
        CONNECTION(DAB2) PROTOCOL(LU61)
        SENDPFX(Y5)    RECEIVEPFX(Z5)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAB2) ...
*
DEFINE  SESSIONS(DAB3Y6Z6) GROUP(CONNDT##)
        CONNECTION(DAB3) PROTOCOL(LU61)
        SENDPFX(Y6)    RECEIVEPFX(Z6)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAB3) ...
*
DEFINE  SESSIONS(DAC1Y7Z7) GROUP(CONNDT##)
        CONNECTION(DAC1) PROTOCOL(LU61)
        SENDPFX(Y7)    RECEIVEPFX(Z7)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAC1) ...
*
DEFINE  SESSIONS(DAC2Y8Z8) GROUP(CONNDT##)
        CONNECTION(DAC2) PROTOCOL(LU61)
        SENDPFX(Y8)    RECEIVEPFX(Z8)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAC2) ...
*
DEFINE  SESSIONS(DAC3Y9Z9) GROUP(CONNDT##)
        CONNECTION(DAC3) PROTOCOL(LU61)
        SENDPFX(Y9)    RECEIVEPFX(Z9)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAC3) ...
*
DEFINE  SESSIONS(DAD1YAZA) GROUP(CONNDT##)
        CONNECTION(DAD1) PROTOCOL(LU61)
        SENDPFX(YA)    RECEIVEPFX(ZA)    SENDCOUNT(10)
        RECEIVECOUNT(10) DESC(Down sessions with DAD1) ...

```

*Figure 33. Resource Definitions for Links from Terminal-Owning Regions to Application-Owning Regions (Part 2 of 3). These define the links from a terminal-owning region to each of the transaction routing application-owning regions shown in the target configuration in Figure 1 on page 18. The definitions are the same for all the terminal-owning regions, and therefore can be included in the startup group list for each TOR, using the generic CSD GROUP name CONNDT##.*

```

*
DEFINE SESSIONS(DAD2YBZB) GROUP(CONNDT##)
CONNECTION(DAD2) PROTOCOL(LU61)
SENDPFX(YB) RECEIVEPFX(ZB) SENDCOUNT(10)
RECEIVECOUNT(10) DESC(Down sessions with DAD1) ...

*
DEFINE SESSIONS(DAD3YCZC) GROUP(CONNDT##)
CONNECTION(DAD3) PROTOCOL(LU61)
SENDPFX(YC) RECEIVEPFX(ZC) SENDCOUNT(10)
RECEIVECOUNT(10) DESC(Down sessions with DAD1) ...

```

Figure 33. Resource Definitions for Links from Terminal-Owning Regions to Application-Owning Regions (Part 3 of 3). These define the links from a terminal-owning region to each of the transaction routing application-owning regions shown in the target configuration in Figure 1 on page 18. The definitions are the same for all the terminal-owning regions, and therefore can be included in the startup group list for each TOR, using the generic CSD GROUP name CONNDT##.

## Links from the Application-Owning Regions

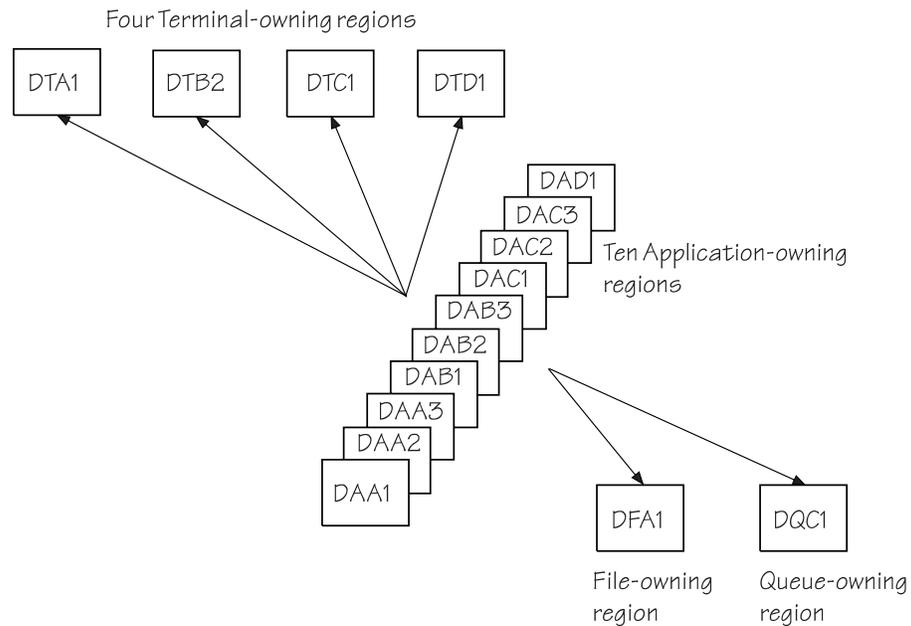


Figure 34. Required Links, 6 in all, from Application-Owning Regions to the Terminal-Owning Regions and Resource-Owning Regions

For the links shown in Figure 34, each of the 12 application-owning regions that supports transaction routing and function shipping requires the following connection and session definitions:

- A connection definition, with supporting sessions, to each terminal-owning region—a total of 4 of each resource type.
- A connection definition, with supporting sessions, to the file-owning region—one only of each resource type.
- A connection definition, with supporting sessions, to the queue-owning region—one only of each resource type.

By creating one generic resource group in the CSD to define all the connections and sessions, the same definitions can be installed in each application-owning region. In this example, we have chosen CONNDA## as the CSD group name, where:

- CONN** indicates the resource type (CONNECTIONS and sessions).
- D** identifies the Dallas CICSplex.
- A** indicates the type of region in which the resource is installed (an application-owning region).
- ##** indicates a generic group that can be installed in more than one region (an application-owning region in this case).

The definitions required for our example configuration are shown in Figure 35.

```

** CONNECTION Definitions from any AOR to the TORs, FOR and QOR
**
DEFINE CONNECTION(DTA1) GROUP(CONNDA##)
    NETNAME(CICSDTA1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DTA1 from any AOR. in the 'D' CICSplex.) ...
*
DEFINE CONNECTION(DTB1) GROUP(CONNDA##)
    NETNAME(CICSDTB1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DTB1 from any AOR. in the 'D' CICSplex.) ...
*
DEFINE CONNECTION(DTC1) GROUP(CONNDA##)
    NETNAME(CICSDTC1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DTC1 from any AOR. in the 'D' CICSplex.) ...
*
DEFINE CONNECTION(DTD1) GROUP(CONNDA##)
    NETNAME(CICSDTD1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DTD1 from any AOR. in the 'D' CICSplex.) ...
*
DEFINE CONNECTION(DFA1) GROUP(CONNDA##)
    NETNAME(CICSDFA1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DFA1 from any AOR. in the 'D' CICSplex.) ...
*
DEFINE CONNECTION(DQC1) GROUP(CONNDA##)
    NETNAME(CICSDQC1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DQC1 from any AOR. in the 'D' CICSplex.) ...
**
** SESSIONS Definitions from any AOR to the TORs, FOR and QOR
**
DEFINE SESSIONS(DTA1W1X1) GROUP(CONNDA##)
    CONNECTION(DTA1) PROTOCOL(LU61)
    SENDPFX(W1) RECEIVEPFX(X1) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(Up sessions with DTA1) ...
*
DEFINE SESSIONS(DTB1W2X2) GROUP(CONNDA##)
    CONNECTION(DTB1) PROTOCOL(LU61)
    SENDPFX(W2) RECEIVEPFX(X2) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(Up sessions with DTB1) ...

```

*Figure 35. Resource Definitions for Links from the Application-Owning Regions to Terminal-Owning Regions and Resource-Owning Regions (Part 1 of 2). These define the links from each application-owning region shown in Figure 1 on page 18. These definitions are the same for all the AORs, and can be included in their start-up group lists, using the generic CSD GROUP name CONNDA##.*

```

*
DEFINE SESSIONS(DTC1W3X3) GROUP(CONNDA##)
CONNECTION(DTC1) PROTOCOL(LU61)
SENDPFX(W3) RECEIVEPFX(X3) SENDCOUNT(10)
RECEIVECOUNT(40) DESC(Up sessions with DTC1) ...

*
DEFINE SESSIONS(DTD1W4X4) GROUP(CONNDA##)
CONNECTION(DTD1) PROTOCOL(LU61)
SENDPFX(W4) RECEIVEPFX(X4) SENDCOUNT(10)
RECEIVECOUNT(40) DESC(Up sessions with DTD1) ...

*
DEFINE SESSIONS(DFA1Y1Z1) GROUP(CONNDA##)
CONNECTION(DFA1) PROTOCOL(LU61)
SENDPFX(Y1) RECEIVEPFX(Z1) SENDCOUNT(40)
RECEIVECOUNT(10) DESC(Down sessions with DFA1) ...

*
DEFINE SESSIONS(DQC1Y2Z2) GROUP(CONNDA##)
CONNECTION(DQC1) PROTOCOL(LU61)
SENDPFX(Y2) RECEIVEPFX(Z2) SENDCOUNT(40)
RECEIVECOUNT(10) DESC(Down sessions with DQC1) ...

```

Figure 35. Resource Definitions for Links from the Application-Owning Regions to Terminal-Owning Regions and Resource-Owning Regions (Part 2 of 2). These define the links from each application-owning region shown in Figure 1 on page 18. These definitions are the same for all the AORs, and can be included in their start-up group lists, using the generic CSD GROUP name CONNDA##.

## Links from the File-Owning and Queue-Owning Regions

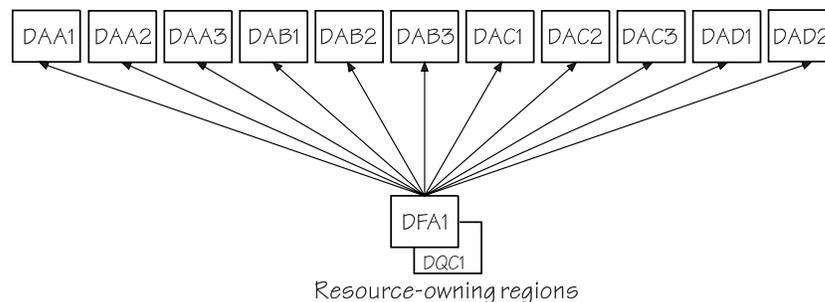


Figure 36. Required Links, 12 in All, from File-Owning and Queue-Owning Regions to All Application-Owning Regions

Each file-owning and queue-owning region requires a connection definition, with supporting sessions, to each application-owning region—a total of 12 of each definition type, as shown in Figure 36. By creating one generic resource group name in the CSD to define all 12 connections and sessions, the same definitions can be installed in the file-owning region and queue-owning region.

**Note:** The definitions you need to install in these regions to connect to the application-owning regions are the same, except for the session counts, as those you need to install in the terminal-owning regions (CSD group CONNDT## in these examples). However, if you are using the old-style prefix-naming for the sessions, you should create a separate group to define different session names (using the “up” convention letters instead of “down” as in CONNDT##). If you are leaving the send and receive prefixes blank and letting CICS allocate the session names, you could copy the same basic resource definitions, and modify them to set an appropriate number of SEND and RECEIVE sessions.

In this example, we have chosen to create a separate group with CONNDR## as the CSD group name, where:

**CONN** indicates the resource type (CONNECTIONS and sessions).

**D** identifies the Dallas CICSplex.

**R** indicates the type of region in which the resource is installed, (resource-owning regions).

**##** indicates a generic group that can be installed in more than one region.

Using this technique, the group can be installed in any resource-owning region that requires a connection to the same set of application-owning regions, making it easy to add another resource-owning region.

The definitions required for these links (illustrated in Figure 36 on page 131) are given in Figure 37.

```
** CONNECTION Definitions from DFA1 and DQC1 to all the AORs.
**
DEFINE CONNECTION(DAA1) GROUP(CONNDR##)
    NETNAME(CICSDAA1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAA1 from DFA1 and DQC1) ...
*
DEFINE CONNECTION(DAA2) GROUP(CONNDR##)
    NETNAME(CICSDAA2) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAA2 from DFA1 and DQC1.) ...
*
DEFINE CONNECTION(DAA3) GROUP(CONNDR##)
    NETNAME(CICSDAA3) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAA3 from DFA1 and DQC1.) ...
*
DEFINE CONNECTION(DAB1) GROUP(CONNDR##)
    NETNAME(CICSDAB1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAB1 from DFA1 and DQC1.) ...
*
DEFINE CONNECTION(DAB2) GROUP(CONNDR##)
    NETNAME(CICSDAB2) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAB2 from DFA1 and DQC1.) ...
*
DEFINE CONNECTION(DAB3) GROUP(CONNDR##)
    NETNAME(CICSDAB3) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAB3 from DFA1 and DQC1.) ...
```

*Figure 37. Resource Definitions for Links from File-Owning and Queue-Owning Regions (Part 1 of 3). These define the links from the file-owning and queue-owning regions to all the application-owning regions shown in Figure 1 on page 18. The definitions are the same for both regions, and therefore can be included in the startup group list for each, using the generic CSD GROUP name, CONNDR##.*

```

DEFINE CONNECTION(DAC1) GROUP(CONNDR##)
    NETNAME(CICSDAC1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAC1 from DFA1 and DQC1.) ...
*
DEFINE CONNECTION(DAC2) GROUP(CONNDR##)
    NETNAME(CICSDAC2) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAC2 from DFA1 and DQC1.) ...
*
DEFINE CONNECTION(DAC3) GROUP(CONNDR##)
    NETNAME(CICSDAC3) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAC3 from DFA1 and DQC1.) ...
*
DEFINE CONNECTION(DAD1) GROUP(CONNDR##)
    NETNAME(CICSDAD1) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAD1 from DFA1 and DQC1.) ...
*
DEFINE CONNECTION(DAD2) GROUP(CONNDR##)
    NETNAME(CICSDAD2) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAD2 from DFA1 and DQC1.) ...
*
DEFINE CONNECTION(DAD3) GROUP(CONNDR##)
    NETNAME(CICSDAD3) ACCESSMETHOD(XM) INSERVICE(YES)
    DESC(Link to DAD3 from DFA1 and DQC1.) ...
**
** SESSIONS Definitions from DFA1 and DQC1 to all the AORs.
**
DEFINE SESSIONS(DAA1W1X1) GROUP(CONNDR##)
    CONNECTION(DAA1) PROTOCOL(LU61)
    SENDPFX(W1) RECEIVEPFX(X1) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(UP sessions with DAA1) ...
*
DEFINE SESSIONS(DAA2W2X2) GROUP(CONNDR##)
    CONNECTION(DAA2) PROTOCOL(LU61)
    SENDPFX(W2) RECEIVEPFX(X2) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(UP sessions with DAA2) ...
*
DEFINE SESSIONS(DAA3W3X3) GROUP(CONNDR##)
    CONNECTION(DAA3) PROTOCOL(LU61)
    SENDPFX(W3) RECEIVEPFX(X3) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(UP sessions with DAA3) ...
*
DEFINE SESSIONS(DAB1W4X4) GROUP(CONNDR##)
    CONNECTION(DAB1) PROTOCOL(LU61)
    SENDPFX(W4) RECEIVEPFX(X4) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(UP sessions with DAB1) ...
*
DEFINE SESSIONS(DAB2W5X5) GROUP(CONNDR##)
    CONNECTION(DAB2) PROTOCOL(LU61)
    SENDPFX(W5) RECEIVEPFX(X5) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(UP sessions with DAB2) ...

```

*Figure 37. Resource Definitions for Links from File-Owning and Queue-Owning Regions (Part 2 of 3). These define the links from the file-owning and queue-owning regions to all the application-owning regions shown in Figure 1 on page 18. The definitions are the same for both regions, and therefore can be included in the startup group list for each, using the generic CSD GROUP name, CONNDR##.*

```

DEFINE SESSIONS(DAB3W6X6) GROUP(CONNDR##)
    CONNECTION(DAB3) PROTOCOL(LU61)
    SENDPFX(W6) RECEIVEPFX(X6) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(Up sessions with DAB3) ...
*
DEFINE SESSIONS(DAC1W7X7) GROUP(CONNDR##)
    CONNECTION(DAC1) PROTOCOL(LU61)
    SENDPFX(W7) RECEIVEPFX(X7) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(Up sessions with DAC1) ...
*
DEFINE SESSIONS(DAC2W8X8) GROUP(CONNDR##)
    CONNECTION(DAC2) PROTOCOL(LU61)
    SENDPFX(W8) RECEIVEPFX(X8) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(Up sessions with DAC2) ...
*
DEFINE SESSIONS(DAC3W9X9) GROUP(CONNDR##)
    CONNECTION(DAC3) PROTOCOL(LU61)
    SENDPFX(W9) RECEIVEPFX(X9) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(Up sessions with DAC3) ...
*
DEFINE SESSIONS(DAD1WAXA) GROUP(CONNDR##)
    CONNECTION(DAD1) PROTOCOL(LU61)
    SENDPFX(WA) RECEIVEPFX(XA) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(Up sessions with DAD1) ...
*
DEFINE SESSIONS(DAD2WBXB) GROUP(CONNDR##)
    CONNECTION(DAD2) PROTOCOL(LU61)
    SENDPFX(WB) RECEIVEPFX(XB) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(Up sessions with DAD2) ...
*
DEFINE SESSIONS(DAD3WCXC) GROUP(CONNDR##)
    CONNECTION(DAD3) PROTOCOL(LU61)
    SENDPFX(WC) RECEIVEPFX(XC) SENDCOUNT(10)
    RECEIVECOUNT(40) DESC(Up sessions with DAD2) ...

```

*Figure 37. Resource Definitions for Links from File-Owning and Queue-Owning Regions (Part 3 of 3). These define the links from the file-owning and queue-owning regions to all the application-owning regions shown in Figure 1 on page 18. The definitions are the same for both regions, and therefore can be included in the startup group list for each, using the generic CSD GROUP name, CONNDR##.*

---

## Cloning CICS Regions

Generating a number of identical CICS regions (clones) to support workload management in a dynamic transaction routing environment means initializing the clone regions with the same resource definitions. You can create clones by specifying the same system initialization parameters and, through the system initialization parameters, specify the same resource definitions.

Using the sysplex target configuration described in “Chapter 2. Planning the Subsystem Configuration for a Sysplex” on page 17 as an example, here are some guidelines for cloning CICS regions.

### Cloning Regions of the Same Type

As a general rule, you can clone only those regions that are of the same type; that is, sets of terminal-owning regions, sets of application-owning regions, and so on.

### Defining Common System Initialization Parameters

For each set of clones, define a common set of system initialization parameters that apply to all the CICS clones in the set.

Use the naming convention to identify the SYSIN member that refers to each clone set, using a generic symbol wherever possible. For example, CICS410.SYSIN(CICS410) for the Dallas terminal-owning regions, and CICS410.SYSIN(CICS410) for the Dallas application-owning regions.

By using symbolic parameters for part of the SYSIN member name, you can use the same line of JCL in your standard CICS startup procedure for all CICS regions. See “Using SYSIN for Common System Initialization Parameters” on page 156 for an example.

## Defining the Unique System Initialization Parameters

For each region, define separately those system initialization parameters that are unique to the region. The only system initialization parameters that must be unique are the identifiers—the APPLID, SYSIDNT, MNSUBSYS, and the DBCTLID parameters.

The most convenient way to define the unique parameters is in the PARM parameter of the EXEC PGM=DFHSIP statement, using symbolic parameters. This provides the greatest degree of flexibility.

## Examples of SYSIN Members for Cloning Application-Owning Regions

To initialize, say, the 12 application-owning regions in the parallel sysplex target configuration, you need:

- The default SIT (DFHSIT)
- All the common system initialization parameters defined in a SYSIN member named, for example, CICS410.SYSIN(CICS410)
- The system initialization parameters that are unique to each region represented by symbols in the CICS startup procedure.

In this illustration, we have assumed that the 12 application-owning regions can all support the same applications—that they are all clones one of another. The following are some factors you should consider when planning which regions are to be exact clones:

- You need multiple application-owning regions for workload balancing and for workload separation. However, regardless of workload, consider making all application-owning regions the same. This can save time and effort.
- Cloning all the application-owning regions in a CICSplex means that there are redundant transaction definitions installed in some regions because of workload separation managed by the terminal-owning regions. Even if an application-owning region is defined to handle transactions for all applications, the dynamic transaction routing program only routes to application-owning regions those transactions that they are allowed to receive.
- Do redundant transaction definitions matter? They are installed above 16MB, and therefore should not be a problem in terms of virtual storage used.  
(The autoinstall facility for programs and mapsets, introduced in CICS/ESA 4.1, means there should not be any redundant program definitions, because these are created dynamically only when needed.)
- Application-owning regions that are special in some way, and need to be distinguished, should not be cloned.
- For CICS cold starts, use multiple CSD group lists to control the definitions in the sets of CICS clones. “Chapter 17. Planning the CICS Startup Procedures” on page 149 contains a number of examples of system initialization parameters defined in members of a SYSIN data set, and each of these specifies 4 CSD

group lists. (Although 4 names is the maximum allowed on the GRPLIST parameter, by specifying generic names you can actually reference many more than 4 lists.)

- All the application-owning regions in the target configuration can be cloned, even though each set of 3 in the 4 MVS images is connected to a different DBCTL and a different DB2.

For examples of what a SYSIN member for the common system initialization parameters might typically contain, see “The Application-Owning Region’s Parameters” on page 152.

---

## Chapter 15. Planning CICSplex Security

This chapter discusses a number of security considerations associated with operating an MRO CICSplex. It covers the following topics:

- Defining bind-time and link security
- Authenticating users in remote MRO regions
- Guaranteeing equal access to resources for cloned application-owning regions.
- MRO link security considerations

---

### Defining Bind-Time and Link Security

MRO bind-time security is changed in CICS/ESA 4.1, and these changes could also affect link security, depending on how you have defined link security on your earlier releases of CICS.

CICS no longer uses its own internal MRO security mechanism, based on the SECURITYNAME parameter from the CONNECTION resource definition, for bind-time security authorization. Instead, it uses the MVS system authorization facility (SAF) interface to call an external security manager, such as RACF, or the SureWay Security Server element of OS/390 or z/OS.

**Note:** When you see the term RACF in the following topics, understand it to mean either the RACF product or the Security Server element to establish the security authorization of each CICS region for MRO logon and connect processing.

You must review the following changes and redefine your bind-time and link security as necessary.

#### Defining Bind-Time Security

The change to CICS bind-time security means that you must define new DFHAPPL.applid profiles to RACF in order to authorize CICS regions to log on to MRO and to connect to one another.

The RACF definitions that you need for MRO security are described in the *CICS RACF Security Guide*.

#### Defining Link Security

There are changes affecting link security as a result of making SECURITYNAME obsolete for MRO in CICS/ESA 4.1.

In earlier releases of CICS, if you do not specify the link USERID parameter on the SESSIONS definition, by default the remote CICS region uses the SECURITYNAME parameter to sign on the link for link security.

In CICS/ESA 4.1, if you do *not* specify the USERID parameter on the SESSIONS definition, a CICS region “signs-on” its partner for link security purposes using the CICS region userid passed by its partner in the bind-time security check. For example, an application-owning region signs on a terminal-owning region using the CICS region userid of the TOR.

For more information, see “MRO Link Security Considerations” on page 140.

---

## Authenticating Users in Remote MRO Regions

Generally, a remote MRO region, such as an application-owning region, signs on userids in the following situations:

- As a result of receiving an MRO request, such as a transaction-routing request or a function-shipping request, when the CONNECTION resource definition specifies ATTACHSEC=IDENTIFY
- As a result of a signon request from a user that has logged on directly through VTAM by means of the application-owning regions APPLID (if known).

The name that CICS passes to RACF as the APPL profile name for signon authentication is different for these two situations.

**Note:** CICS also signs on link userids, but this is a separate topic discussed under “MRO Link Security Considerations” on page 140.

## Authenticating Users Associated with MRO Requests

In CICS/ESA 4.1, a remote region, such as an application-owning region or file-owning region, uses the same security data as the terminal-owning region when signing on a remote user associated with a transaction. The name that the remote CICS passes to RACF as the APPL profile name for signon validation purposes is the same name used by the terminal-owning region. This name can be one of the following:

- The VTAM generic resources name for the CICSplex, if the GRNAME system initialization parameter is specified in the terminal-owning region
- The CICS generic APPLID of the terminal-owning region if GRNAME is not specified.

The terminal-owning regions pass one of these with transaction routing requests for use by application-owning regions when authenticating the user signon. The same APPL profile name is passed to other remote regions by the application-owning regions with function-shipping requests (or when daisy-chaining transaction routing requests). Remote regions use the passed name when performing the user signon that is required for attach-time security checking when ATTACHSEC=IDENTIFY is specified on the link.

This means that users who are authorized to signon to a terminal-owning region are automatically authorized to access other regions via MRO links.

There are some major benefits accruing from this:

- The security administrator does not need to maintain the access list for separate RACF APPL profiles for each application-owning region to which users need to be authorized, to run routed transactions. When users are authorized to the RACF APPL profile for the terminal-owning regions (using the VTAM generic resource name as the APPL profile name), they are automatically authorized to be signed on by CICS in the application-owning regions.
- RACF 2.1 reuses information cached in the virtual lookaside facility (VLF) following the signon in the terminal-owning region, allowing the sign-on required an application-owning region to be performed without any I/O activity.

## Authenticating Users Signing On Directly to Remote Regions

The signon process for users logged on directly to a remote region, such as an application-owning region or file-owning region, is the same as in earlier releases.

The CICS region passes its own APPLID to RACF as the APPL profile name for signon authentication. Thus, unless users are explicitly authorized to that APPLID they cannot sign on directly to that APPLID.

You can use these signon processes to advantage in a CICSplex, by enabling terminal users to be authorized to all the regions in the CICSplex through the CICSplex generic resources name, whilst at the same time preventing them from signing on directly to particular APPLIDs.

---

## Guaranteeing Equal Access to Cloned Application-Owning Regions

It is important to ensure that all the application-owning regions that process the same workload are defined with the same levels of link security for the terminal-owning region connections. This is to ensure that routed transactions have the same security authorizations across the set of cloned application-owning regions, to ensure that transactions that work when routed on one link, do not fail on another.

It is also important to ensure that all links to a resource-owning region from a set of cloned application-owning regions are defined with the same level of link security. This is to ensure that function-shipped requests have the same security authorizations regardless of which of the cloned application-owning regions originated the request.

You are also recommended to use the same CICS region userid for all the cloned regions, of the same type, in a CICSplex. You should consider, for example, using one userid for all the terminal-owning regions, another for all the application-owning regions, and so on. This ensures that:

- The same basic security rules control all the clones
- The minimum number of definitions are required to the external security manager.

In “Chapter 17. Planning the CICS Startup Procedures” on page 149, we recommend that you use a single procedure to start all CICS regions in the CICSplex, and that you start CICS regions as started tasks. To follow this recommendation, and at the same time run each type of region under a unique userid, you need to use MVS JOB support for started tasks and the associated RACF STARTED class profiles.

The earlier method of applying security to started tasks, using the RACF started procedures table to associate the userid with the procedure name, restricts you to only one userid per procedure name. Using STARTED class profiles, with MVS/ESA SP 5.1 JOB support for started tasks, enables you to assign different userids for different invocations of the procedure.

## Defining CICS Region Userids for Started Jobs

If you use a single procedure to start all your CICS regions as started tasks, they all execute under the authorization of the userid associated with the procedure in the RACF started task table, ICHRIN03.

You are recommended to modify your CICS startup procedure so that it can be executed as a **started job** (see Figure 43 on page 159 for an example of such a procedure with a JOB statement). The security mechanism for started jobs enables you to use a different RACF userid for each started job. Alternatively, you could use

one userid for all the regions of the same type. See “MRO Link Security Considerations” for information about using one or more userids for starting CICS regions.

The support for multiple userids for started jobs is provided by the RACF STARTED general resource class, and its associated STDATA segment. You define profiles in this class for each job that needs to run under a unique userid.

The following example shows how to define a generic profile for jobs that are to be started using the CICSTASK procedure, where the job names begin with the letters CICSDDT (the terminal owning regions in our target configuration):

```
RDEFINE STARTED (CICSTASK.CICSDT*) STDATA( USER=(CICSDT##) )
```

When you issue the START command to start CICSTASK with a job name of, say, CICSDDTA1, MVS passes the member name (CICSTASK), and the job name (CICSDDTA1) in order to obtain the userid under which this CICS terminal-owning region is to run. In the example shown above, CICSDT## is the userid under which all the terminal-owning regions run if they are started using started JOB support.

You must ensure that the userids specified in the STARTED class profile are defined to RACF. You must also ensure that the userids are properly authorized to the data set profiles of the CICS regions that run under them.

There are several factors to consider when planning the userids under which the regions in your CICSplex are going to run. On the one hand you might want different authorizations for the different types of region. However, you would probably consider it unnecessary to run every region under a unique userid, and it would cost time and effort to set up and maintain such number of userids. One way of keeping the number of profiles and userids to a minimum is to run all the cloned regions of *each type* under the same CICS region userid. This means that for our target configuration, described in Chapter 2. Planning the Subsystem Configuration for a Sysplex, we could run the Dallas CICSplex under five userids as follows:

<b>CICSDT##</b>	The common region userid for all terminal-owning regions
<b>CICSDDA##</b>	The common region userid for all the application-owning regions
<b>CICSDDFA1</b>	The region userid for the file-owning region
<b>CICSDDQA1</b>	The region userid for the queue-owning region

On the other hand, using different userids for connected regions automatically enforces link security when you are running the CICSplex with RACF security if you omit to specify a link security userid on the SESSIONS definitions for MRO links.

---

## MRO Link Security Considerations

In our target CICSplex configuration shown in Figure 1 on page 18, users signing on to the terminal-owning regions need access to all the application-owning regions. The CICS/ESA 4.1 signon authentication process ensures this (see “Authenticating Users in Remote MRO Regions” on page 138). With this mind, you may conclude that from a security viewpoint, all the CICS regions are one logical entity, all regions of the same type being clones of one another. In this case, you may decide that you don’t need to operate MRO link security, and avoid the overhead on link security checking.

## Bypassing Link Security Checking

You provide the userid for link security checking on the USERID parameter of the SESSIONS resource definition for the MRO connection.

To establish whether you want link security checking, or whether it is to be bypassed, CICS compares its *own* region userid with the link userid defined on the SESSIONS definition:

- If the userids are the same, link security checking is bypassed.
- If they are different, CICS “signs on” on the userid specified on the SESSIONS definition, and uses this on all link security checks.

If you do not specify a link security userid on the SESSIONS definition, CICS uses the region userid of its partner instead. When a CICS MRO region receives its partner’s region userid during bind security checking, it compares this with its own region userid. If they are the same, link security checking is bypassed, and the only security checks made in, for example, an application-owning region are determined by the ATTACHSEC parameter on the connection definition.

For information about the effect of the ATTACHSEC parameter, see Table 4 on page 142.

## Authorizing the Link Userid

CICS calls RACF for the usual signon authentication when signing on the link userid. For this purpose, the CICS region uses its own APPLID as the profile name of the APPL resource class. This means you must authorize the link userid to the APPL profiles of the CICS region in which it is being installed.

For example, if you:

- Specify CICS00 as the userid on a SESSIONS definition, and
- Install the SESSIONS definition in a set of cloned application-owning region whose APPLIDs are CICS00A1, CICS00A2, and CICS00A3

you must authorize userid CICS00 to the application-owning regions’ APPL profiles named CICS00A1, CICS00A2, and CICS00A3.

### Naming Convention for Link Userids

One suggestion is to define a userid that corresponds, in part, to the region userid of the partner region on the MRO link.

For example, if all the terminal-owning regions to which a set of cloned application-owning regions are connected are all running under CICS00##, you could specify CICS00 as the userid in all the SESSIONS definitions installed in the application-owning regions. This ensures that all the link security checks against transactions routed from any terminal-owning region to any application-owning region are made against the same userid (see “Guaranteeing Equal Access to Cloned Application-Owning Regions” on page 139).

The prefix letters of the link userid (CICS00) also help you to relate this link userid with the regions running under the CICS00## region userid and with APPLIDs beginning with CICS00.

## Security in the Receiving Regions

Whether or not you bypass CICS link security checking, you should consider carefully what type of security you want to enforce in all CICS regions that receive requests from MRO partners. The ATTACHSEC parameter offers two options,

LOCAL or IDENTIFY, the effect of which, with and without link security, are shown in Table 4.

Table 4. MRO Security Options

Link Security	ATTACHSEC	
	LOCAL	IDENTIFY
Yes	CICS ignores the userid associated with the transaction, and issues all security checks against the link userid only.	CICS issues two security checks: <ol style="list-style-type: none"> <li>1. Against the userid associated with the transaction</li> <li>2. Against the link userid, ensuring that the transaction cannot access resources not authorized for the link.</li> </ol>
Bypassed	CICS issues all security checks against the CICS default userid (the userid specified on the DFLTUSER system initialization parameter)	CICS issues security checks against the userid associated with the transaction.

**Recommendations for MRO Security:**

1. You are recommended to specify ATTACHSEC=IDENTIFY in application-owning regions and resource-owning regions. Otherwise, every user accessing resources in that region via MRO is granted the same security authorizations.
2. If you choose to bypass link security checking, you are recommended to do so by making the USERID parameter on the SESSIONS definition the same as the CICS region's userid. Do not rely upon the default mechanism (see "Bypassing Link Security Checking" on page 141).
3. If you choose to employ link security checking, you are recommended to do so by specifying an appropriate userid on the USERID parameter of the SESSIONS definition. Do not rely upon the default mechanism (see "Bypassing Link Security Checking" on page 141).
4. For terminal-owning regions, whose main function is dynamic transaction routing, you are recommended to omit definitions for remote transactions, except where necessary (see "No Transaction Definitions in the Terminal-Owning Regions" on page 123). The terminal-owning regions do not perform transaction-attach security checks on transactions that are attached using the DTRTRAN mechanism and dynamically routed to the application-owning regions. If transactions are attached using the DTRTRAN mechanism, and subsequently run locally, CICS does perform the transaction-attach security check.

---

## Chapter 16. Planning for Workload Management

This chapter discusses the MVS and CICSplex SM workload management requirements for a CICSplex.

Basically, there are three main components involved in managing a CICS workload:

### **CICSplex SM**

You provide CICSplex SM with CICS-specific workload specifications and definitions which CICSplex SM needs to support both workload balancing and workload separation in the CICSplex. These workload specifications and definitions ensure that CICSplex SM has all the necessary information about the transactions that comprise the CICSplex workload, and about all the application-owning regions that are available to process the work.

### **A CICS dynamic routing program**

You need a dynamic transaction routing program that can exploit the services of CICSplex SM and route transactions to the appropriate application-owning regions.

If you have CICSplex SM installed, you can use the dynamic transaction routing program provided by CICSplex SM.

### **The MVS workload manager**

You provide the workload manager with a service definition, which MVS needs to ensure it makes the necessary resources available to the CICS regions and associated subsystems that ultimately process the user transactions. The MVS service definition must cover the workload for the entire sysplex, so that MVS can allocate resources for all types of work—online, batch, and system-related (such as JES, SMF, and other system management components).

The service definition is primarily needed for **goal-oriented** workload management.

To operate a CICSplex of the kind discussed in this book, you must use some, or all, of these components to optimize the performance of the CICSplex, and to achieve the objectives defined in an MVS service definition.

---

## Using CICSplex SM for Workload Balancing and Workload Separation

CICSplex SM supports both workload balancing and workload separation.

### **Workload Separation**

In our target configuration we have assumed that all the application-owning regions are identical clones, with each AOR capable of processing the full range of transactions. Nevertheless, we could still use the workload separation function if we wanted to ensure, for example, that CICS-DB2 and CICS-DL/I transactions ran in different application-owning regions. Using CICSplex SM, we could identify and route the CICS-DB2 transactions to only some of the application-owning regions, and route all other transactions to the remaining application-owning regions.

### **Workload Management**

CICSplex SM supports two workload balancing algorithms:

#### **The queue algorithm**

Working with this algorithm, CICSplex SM selects the application-owning region that has the shortest queue of transactions (in proportion to the specified

maximum number of tasks for that region). Although the queue length is the primary factor that CICSplex SM uses in the queue algorithm, there are other factors that also affect the choice of application-owning region.

In our target configuration, where all application-owning regions are running in MVS images residing in CPCs of the same capacity, the MXT system initialization parameter should specify the same values for the cloned application-owning regions. In a situation where all the MXT values are the same, it's the actual queue length that CICSplex SM uses in its algorithm.

### **The goal algorithm**

Working with this algorithm, CICSplex SM selects the application-owning region that is most likely to meet the response time goals set for the transaction in the MVS workload manager service definition. Although the goals set for a transaction are the primary factor that CICSplex SM uses in the goal algorithm, there are other factors that also affect the choice of application-owning region.

For this algorithm, CICSplex SM uses the MVS workload management interfaces to obtain information about MVS workload policies.

For both these methods, CICSplex SM maintains information about the state of the candidate application-owning regions, which it obtains by constant monitoring through its agents resident in the application-owning regions.

For information about the other factors that CICSplex SM uses in these two workload balancing algorithms, see, the *CICSplex SM Concepts and Planning* manual.

### **Where to Start**

To begin with, we recommend that you use the workload balancing queue algorithm. This involves the least administrative effort, and is easier to implement. With this method, you do not need to define an MVS workload manager service definition.

We also recommend that you use the dynamic transaction routing program supplied with CICSplex SM.

## **Implementing Shortest-Queue Workload Balancing**

The CICSplex SM dynamic transaction routing program is controlled by values you supply to CICSplex SM in the form of definitions. These are:

- The workload specification
- The workload definition
- The transaction group

CICSplex SM uses all the available information when selecting the best application-owning region, from its list of available application-owning regions, to which a transaction should be routed.

The information you provide to CICSplex SM can include details about inter-transaction affinities, and CICSplex SM takes these affinities into account when selecting a suitable application-owning region.

### **The CICSplex SM Workload Management Entities**

To implement the shortest-queue algorithm, begin by defining a workload specification (you need only the workload specification for the shortest-queue algorithm). The information it provides includes:

- The name of the specification

- The name of a default set of application-owning regions among which transactions can be routed.
- The type of workload algorithm—QUEUE in this case.

**Workload Separation:** For workload separation purposes, you must also provide workload definitions. Such definitions would be needed for our target configuration to specify separately the DL/I workload and the DB2 workload.

**Inter-Transaction Affinities:** Some of your applications may contain transactions that cannot be freely dynamically routed because of inter-transaction affinities. Although you could resolve problems of affinity by defining such transactions to be statically routed, you should nevertheless consider making them dynamic and let CICSplex SM manage the affinity. For this purpose you define transaction groups, in which you specify details of the affinities that you want CICSplex SM to manage.

Most affinities are of limited duration and scope, often being related to an instance of a pseudo-conversational transaction from a particular terminal. In most cases, CICSplex SM can dynamically route transactions until an affinity begins, observe the affinity restriction while it is in effect, and as soon as the affinity is ended, begin dynamically routing again.

See the *CICS Application Programming Guide* manual for information about the duration and scope of inter-transaction affinities.

For more information about the workload management support provided by CICSplex SM, see the *CICSplex SM Concepts and Planning* manual.

---

## Providing a Dynamic Transaction Routing Program

If you design and write your own dynamic transaction routing program, you must also plan to provide a number of supporting functions that can only be performed outside the dynamic transaction routing program itself. These supporting functions are concerned with monitoring the status of the set of application-owning regions that are candidates for dynamic transaction routing, and providing this information to the dynamic transaction routing programs that are running in the terminal-owning regions.

Without the support of a system management product such as CICSplex SM, you must provide these support functions yourself, as well as write the dynamic transaction routing program. These functions can be very complex and costly to design and implement. For more information about design considerations for developing your own dynamic transaction routing environment, see the *CICS/ESA Dynamic Transaction Routing in a CICSplex*.

An alternative approach, and the one we recommend, is to use CICSplex SM, or an equivalent system management program product. In particular, we recommend that you use a product that provides both a ready-made dynamic transaction routing program **and** all the supporting functions. CICSplex SM provides a programmable interface that you can use in your own dynamic transaction routing program, enabling you to make use of the CICSplex SM workload separation, workload balancing, and affinity functions.

However, we recommend that if you have CICSplex SM installed that you use its dynamic transaction routing program. The dynamic transaction routing program provided with CICSplex SM is called EYU9XLOP. For an overview of the function provided by EYU9XLOP, see *CICSplex SM Concepts and Planning* manual.

---

## The MVS Workload Manager

You can implement CICSplex SM's shortest-queue workload management regardless of whether you are using MVS workload manager for other subsystems with the sysplex.

There are basically three states with regard to MVS workload manager:

- MVS workload management support not active
- MVS workload manager running in compatibility mode
- MVS workload manager running in goal mode.

You can operate your CICSplex with CICSplex SM's shortest-queue algorithm with the MVS workload manager in any of these states.

## Preparing to Migrate to Goal-Mode Workload Management

When you decide to change from shortest-queue to goal-oriented workload management, you will have to determine CICS response times to set in your MVS service class definitions.

The MVS service definition consists of the following elements:

- Service policies
- Workloads
- Service classes
- Resource groups
- Classification rules

These are explained in the *z/OS MVS Planning: Workload Management*.

### Determining CICS response times before defining goals

Before you set goals for CICS work, you can determine CICS current response times by running CICS in compatibility mode with an arbitrary goal. For this purpose, use the SRVCLASS parameter, provided by MVS/ESA SP 5.1 in the installation control specification (ICS). This parameter lets you associate a service class with a report performance group, to be run in compatibility mode. You would then:

1. Define a service policy, with a default service class, or classes, for your CICS work, and specify an arbitrary response time goal (say 3 seconds).
2. Define classification rules for the service class or classes
3. Install the service definition
4. Activate the service policy in compatibility mode.

The average response time for work within the service classes are reported under the report performance group in the RMF Monitor I workload activity report.

This information helps you to set realistic goals for running your CICS work when you switch to goal mode. The reporting data produced in RMF reports:

- Is organized by service class
- Contains reasons for any delays that affect the response time for the service class (for example, because of the actions of a resource manager or an I/O subsystem).

From the reported information, you may be able to determine configuration changes to improve performance.

**Example of using SRVCLASS parameter of IEAICSxx:** To obtain CICS response time information while in compatibility mode, you can set up the following:

- In your service definition, set up the following:
    - A test policy, comprising the following:
 

```
Service Policy Name . . . . : CICSTEST
Description . . . . . : Migration (compatibility) mode
```
    - A workload definition, in which to define the required service class:
 

```
Workload Name . . . . . : CICSALL
Description . . . . . : CICSTEST migration workload
```
    - A service class for all CICS transactions:
 

```
Service Class Name . . . . . : CICSALL
Description . . . . . : All CICS transactions
Workload Name . . . . . : CICSALL
---Period--- -----Goal-----
Action # Duration Imp. Description
— 1 1 1 Average response time of 00:00:03.000
```

**Note:** It does not matter what goal you specify, since it is not used in compatibility mode, but it cannot be discretionary.

  - Specify the name of the service class under the classification rules for the CICS subsystem:
 

```
Subsystem Type . . . . . : CICS
Default Service Class . . : CICSALL
```
- In your ICS member in SYS1.PARMLIB (IEAICSxx), specify:
 

```
SUBSYS=CICS,
SRVCLASS=CICSALL,RPGN=100
```
- Install the workload definition in the coupling facility.
- Activate the test service policy, either by using options provided by the WLM ISPF application, or by issuing the following MVS command:
 

```
VARY WLM,POLICY=TEST
```

You receive response time information about CICS transactions in the RMF Monitor I Workload Activity Report under report performance group 100. For more information about defining performance goals and the use of SRVCLASS, see *z/OS MVS Planning: Workload Management*.

When you are ready to switch from CICSplex SM's shortest-queue to goal-oriented workload management, change the CICSplex SM workload specification from QUEUE to GOAL, and activate the appropriate MVS workload management service policy.



---

## Chapter 17. Planning the CICS Startup Procedures

This chapter discusses methods of starting multiple CICS regions within the sysplex. These methods are designed to reduce the system administration work involved to a minimum.

The chapter covers the following topics:

- The CICS system initialization parameters
- The required CICS system data sets
- The CICS startup procedure.

---

### The CICS System Initialization Parameters

Attempting to tailor every CICS region by defining a unique set of system initialization parameters for each region is not only costly initially, but requires significant effort to maintain. We recommend that you consider the following approach:

- Use the CICS-supplied default table, DFHSIT, to avoid generating many system initialization tables that require identifying suffixes.
- Define a set of system initialization parameters that are common to **types** of region, such as terminal-owning regions or application-owning regions.
- Use the SYSIN data set to override system initialization table default values for parameters that are common to CICS region type.
- Use the MVS PARM parameter on the EXEC PGM=DFHSIP statement to pass those system initialization parameters that must be unique.

Each of these ideas is discussed in a little more detail below.

### Using the Default System Initialization Table

CICS requires a system initialization table at startup to provide the system initialization parameters for the CICS region.

Parameters defined in a system initialization table can be overridden at startup by system initialization parameters supplied in:

- **The PARM parameter of the JCL EXEC statement.**

Use this method to provide the system initialization parameters that are unique to each region.

- **The SYSIN data set.**

Use this method to provide the system initialization parameters that are common to the cloned regions.

**Note:** The CICS system initialization parameters can also be entered at the MVS console, but this is not considered an acceptable option for production systems.

If you define your parameters in the DFHSIT macro, and generate your own table with a suffix, the identifying suffix must be specified to CICS at startup on the SIT parameter.

The method we recommend is to omit the SIT startup parameter from the CICS startup JCL, and allow CICS to load DFHSIT, the unsuffixed default table. You can then supply, from the SYSIN data set, the system initialization parameters that you need to tailor your CICS regions.

**Can I override all the default table parameters?:** You can override all except two of the parameters in the system initialization table. For security reasons, these are:

- The **external security manager (ESM) exits** parameter (ESMEXITS) controlling the passing of installation data to the ESM. The default value is NOINSTLN.
- The **VTAM high performance option** parameter (HPO) controlling the VTAM authorized path feature of the HPO. The default is NO.

If you need to generate a system initialization table in order to modify the default values of these restricted parameters, use the source of the CICS-supplied default table. This is called DFHSIT\$\$, and is supplied in the CICS samples library, CICS410.SDFHSAMP. Change the relevant parameters, and reassemble the unsuffixed default table.

**Customized system initialization tables for cloned regions:** If you prefer to use a customized system initialization table rather than the default table, you should plan to use the same table for each type of region—one for all terminal-owning regions, one for all application-owning regions, and so on.

## Defining Common System Initialization Parameters for Cloned Regions

Study the system initialization parameter requirements for each type of region, and identify those parameters that are applicable to the region type.

Parameters generally are applicable to a specific function. If the region is not required to support a particular function you can ignore that function's parameters.

The CICS region types in an MRO CICSplex are defined by the resources they manage, which in turn determines the kind of function that is needed in the different types of region.

### The Terminal-Owning Region's Parameters

Identify all the parameters that are relevant to a terminal-owning region, and plan what values you need to specify as common parameters for all the cloned terminal-owning regions.

In particular, consider the system initialization parameters that are applicable to the autoinstall function, terminal control, and VTAM. For example, the autoinstall parameters are as follows:

<b>AIEXIT</b>	Names the autoinstall user-exit program.
<b>AILDELAY</b>	Specifies the logoff delay period after which CICS deletes a terminal entry from the terminal control table.
<b>AIQMAX</b>	Specifies the maximum number of terminals that can be queued for autoinstall.
<b>AIRDELAY</b>	Specifies the logoff delay period after which CICS deletes entries for terminals not in session following an emergency restart.

All these are important in a terminal-owning region, and should be defined with the same values for all cloned terminal-owning regions in the CICSplex.

For information about the other system initialization parameters that relate to terminal control, see the table of system initialization parameters grouped by functional area in the *CICS System Definition Guide*.

**VTAM generic resources name and APPLIDs:** All the terminal-owning regions should specify the same generic resource name on the CICS GRNAME system initialization parameter, and this should be defined in the SYSIN data set with all the other system initialization parameters that are common to terminal-owning regions.

However, the APPLID parameter needs to be unique to each region, and should not be included in the SYSIN data set. See “Using the PARM Parameter for Unique System Initialization Parameters” on page 157 for information about supplying the APPLIDs to each CICS region at startup.

### Example

An example of the system initialization parameters that could be defined as common to all the terminal-owning regions in a CICSplex are shown in Figure 38.

```
* *****
* CICS system initialization parameters common to all cloned TORs *
* *****
CICSSVC=218           The CICS SVC number
CSDDISP=SHR          Disposition for the CSD
CSDDSN=CICS410.DFHCS  Data set name of the CSD
DCT=GQ               The global table for all queues in the CICSplex
DTRPGM=WLMDDYP      Dynamic transaction routing exit
FCT=NO              No file control table (using RDO for files)
GMTEXT='Dallas CICSplex Terminal-Owning Region (CICS/ESA 4.1)'
GRNAME=CICSD###     Generic resource name for the Dallas CICSplex
* CSD Group lists for cloned TORs
GRPLIST=(IBMLIST*,ISVLIST*,CICSST##)
JCT=2$              The CICS sample journal control table
* The IRC & ISC parameters required for MRO
IRCSTRT=YES,        Start IRC during initialization
ISC=YES,            Include the interregion communication programs
MXT=50              Set maximum tasks to 50
PLTPI=DT            PLT table is DFHPLTDT
LPA=YES             Use the LPA for defined and eligible modules
TCT=5$              Sequential terminals required
TCTUALOC=ANY        TCT User Areas location - above 16MB
TRTABSZ=1024        1MB internal trace table
```

Figure 38. Example of Common System Initialization Parameters for a Terminal-Owning Region

**Assumptions for the terminal-owning region:** When planning the system initialization parameters for the terminal-owning regions you need to consider the differences in requirements between them and the other regions. In our example shown in Figure 38 we have made the following assumptions:

#### Storage protection and transaction isolation

These options are allowed to default; that is, there is no storage protection and no transaction isolation (STGPROT=NO and TRANISO=NO) in the terminal-owning region.

The main benefits of CICS storage protection and transaction isolation are the protection of CICS storage from user-key transactions and the isolation of user-key task-lifetime storage from other user-key transactions. It is most likely that any user-defined transactions that execute in a terminal-owning region will be running in CICS key, and therefore will not benefit from CICS storage protection or transaction isolation.

### Dynamic storage limits

The dynamic storage limits above and below the line are allowed to default to 5MB below the 16MB line and 20MB above 16 megabytes (DSALIM=5MB and EDSALIM=20MB).

For dynamic transaction routing, CICS attaches a CICS relay transaction for each routed transaction, and these should comprise almost all the tasks that run in the terminal-owning regions. The default limits should be adequate for this workload.

### Maximum number of tasks

The maximum number of tasks is set to 50 because this is the theoretical maximum that can be processed in 1 second by one of the CPCs shown in our target configuration. With each terminal-owning region limited to 50, this gives a maximum number of tasks of 200 across the 4 terminal-owning regions in our target CICSplex.

### Transient data

It is probable that the only transient data queues you need in the terminal-owning region are the system queues required by CICS. Nevertheless, unless virtual storage is limited (and the DCT is located mostly above 16MB), you can use the same destination control table for the terminal-owning regions as for the application-owning regions.

CICS/ESA 4.1 supports remote attributes on intrapartition and extrapartition transient data entries in the DCT. To decide whether a queue is local or remote, CICS compares the SYSIDNT parameter in the DCT with its own system initialization parameter SYSIDNT. If the SYSIDNT names are the same, the queue is local; if they are different, the queue is remote.

### The Application-Owning Region's Parameters

Identify all the parameters that are relevant to an application-owning region, and plan what values you need to specify as common parameters for all the cloned application-owning regions.

In particular, consider all the system initialization parameters that are applicable in any way to the running of user applications. For example, review the groups of parameters that relate to storage manager, parameter manager and system initialization, intersystem communication, transient data and temporary storage, and journaling.

For information about all the system initialization parameters that relate to the various functional areas, see the table of system initialization parameters grouped by function in the *CICS System Definition Guide*.

**APPLIDs:** You can choose not to define the APPLIDs for each application-owning region to VTAM, which means that the APPLIDs are used only by CICS regions for communication with their MRO partners.

Alternatively, you can control terminal access to the application-owning regions by setting appropriate values on the AIQMAX system initialization parameter. By defining the APPLID to VTAM, but specifying AIQMAX=0, you prevent anyone logging on directly to the application-owning regions through autoinstall, in normal circumstances, while retaining the ability to do so should the need arise. You can do this by modifying the AIQMAX value at the console, using the CEMT SET AUTOINSTALL command. To enable you to do this, you should ensure that the necessary console definitions are installed.

You could also explicitly define a limited number of system programmer's terminals for use on the application-owning regions, and restrict access in this way. Another method is to use RACF to prevent unauthorized access to application-owning regions.

**Using RACF to control access to application-owning regions:** You can also use RACF security checking to restrict logon to an application-owning region. In CICS/ESA 4.1, an application-owning region uses the same security check as the terminal-owning region when signing on a remote user associated with a transaction.

See "Chapter 15. Planning CICSplex Security" on page 137 for details.

**Temporary storage:** To avoid any intertransaction affinity connected with the use of temporary storage, you should plan to define temporary storage queues as remote queues. This requires a temporary storage table, specified on the TST system initialization parameter.

### Example

An example of the system initialization parameters that could be defined as common to all the application-owning regions in a CICSplex are shown in Figure 39.

```
* *****
* CICS system initialization parameters common to all cloned AORs *
* *****
AIQMAX=0           No autoinstall terminals in this region
CMDPROT=YES       CICS command address validation/protection
CICSSVC=218       The CICS SVC number
CSDDISP=SHR       Disposition for the CSD
CSDDSN=CICS410.DFHCSDD  Data set name of the CSD
CSDJID=01         Use system log for auto journaling
CWAKEY=CICS       CICS key for the CWA
DSALIMIT=7M       Allocate 7MB for the DSAs below 16MB.
DBP=1$           Dynamic backout program - no local DL/I support
DCT=GQ           The global table for all queues in the CICSplex
EDSALIMIT=100M   Allocate 100M for the DSAs above 16MB.
FCT=NO           No file control table (using RDO for files)
* CSD Group lists for cloned AORs
GRPLIST=(IBMLIST*,ISVLIST*,CICSDA##)
JCT=2$           The CICS sample journal control table
* The IRC & ISC parameters required for MRO
IRCSTRT=YES,     Start IRC during initialization
ISC=YES,         Include the intersystem communication program
LPA=YES         Use the LPA for defined and eligible modules
MXT=20          Set maximum tasks to 20
PLTPI=DA        PLT table is DFHPLTDA
RENTPGM=PROTECT Read-only DSAs required.
SRT=1$         The CICS sample system recovery table
STGPROT=YES     Storage protection on.
TCT=5$         Sequential terminal support required
TCTUAKEY=CICS   CICS key for TCT user areas
TCTUALOC=ANY    Location of TCTUAs - above 16MB
TRTABSZ=2000    Trace table size
TRANISO=YES     Transaction isolation on.
TST=DA         Temporary storage table is DFHTSTDA
WRKAREA=2048    2KB for the common work area (CWA)
```

Figure 39. Example of Common System Initialization Parameters for an Application-Owning Region

**Assumptions for the application-owning region:** When planning the system initialization parameters for the application-owning regions you need to consider the differences in requirements between them and the other regions. In our example shown in Figure 39 on page 153 we have made the following assumptions:

#### **Storage protection and transaction isolation**

The defaults of STGPROT=NO and TRANISO=NO are overridden because storage protection and transaction isolation are required in the regions where user application programs run.

#### **Dynamic storage limits**

You must plan how much storage you want CICS storage manager to use below and above 16 megabytes for the various dynamic storage areas. Allocate as much as possible below 16 megabytes (while still leaving some non-DSA storage to satisfy MVS GETMAIN requests). Allocate sufficient storage above 16MB to satisfy the storage requirements when CICS is running up to maximum task limit (set by the MXT system initialization parameter). You should also set the MVS REGION parameter in the CICS startup JCL to a value that allows the CICS storage parameters to be satisfied.

For information on specifying dynamic storage limits, see the *CICS System Definition Guide*.

#### **Maximum number of tasks**

Setting the maximum number of tasks to 20 allows up to 60 in total if 3 application-owning regions (or 80 if 4 application-owning regions) are running in each CPC. Allowing a maximum number of tasks of 20 allows for some variation in the rate of throughput per application-owning region.

When setting the MXT parameter for your application-owning regions, remember that in an XCF/MRO environment the application-owning regions are supplied with transactions from all the terminal-owning regions, not just those running in its own MVS image.

#### **Transient data**

The application-owning regions share the same DCT with the queue-owning region. By means of the SYSIDNT parameter on the intrapartition and extrapartition queue entries, CICS can determine whether the queues are local or remote.

By default, queues are treated as local queues. You should omit the SYSIDNT parameter from the CICS-required queues, (such as CSMT, CADL, and so on) to ensure each CICS region manages its own queues locally.

#### **The File-Owning Region's Parameters**

It is unlikely that you will have many file-owning regions in a CICSplex, but even if you have only one you should plan to define the system initialization parameters in the SYSIN data set in the same way as for the other regions.

Some of the considerations that apply to the cloned terminal-owning regions also apply to the file-owning regions. An file-owning region typically runs only CICS code, therefore it would not be appropriate to run file-owning regions with storage protection and transaction isolation.

You might also apply some application-owning region considerations to the file-owning regions—for example, by either not defining APPLIDs to VTAM or inhibiting autoinstall for terminals.

You are recommended to specify MROLRM=YES for all file-owning regions to ensure the use of long-running mirror tasks.

## Example

An example of the system initialization parameters that could be defined as common to all the file-owning regions in a CICSplex are shown in Figure 40.

```
* ****
* CICS system initialization parameters common to all cloned FORs *
* ****
AIQMAX=0           No autoinstall terminals in this region
CICSSVC=218       The default CICS SVC number
CSDDISP=SHR       Disposition for the CSD
CSDDSN=CICS410.DFHCS  Data set name of the CSD
DBP=1$           Dynamic backout program - no local DL/I support
DCT=GQ           The global table for all queues in the CICSplex
FCT=NO           No file control table (using RDO for files)
* CSD Group lists for the FOR
GRPLIST=(IBMLIST*,ISVLIST*,CICSDF##)
JCT=2$           The CICS sample journal control table
* The IRC & ISC parameters required for MRO
IRCSTRT=YES,     Start IRC during initialization
ISC=YES,         Include the intersystem communication program
MROLRM=YES      Long-running mirrors required
MXT=200         Set maximum tasks to 200 (for mirrors)
RENTPGM=PROTECT Read-only ERDSA required.
PLTPI=DF        PLT table is DFHPLTDF
SRT=1$         The CICS sample System Recovery Table
TCT=NO         Dummy TCT only required.
TRTABSZ=200     Trace table size
```

Figure 40. Example of Common System Initialization Parameters for a File-Owning Region

**Assumptions for the file-owning region:** When planning the system initialization parameters for an file-owning region you need to consider the differences in requirements between it and the other regions. In our example shown in Figure 40 we have made the following assumption:

### Maximum number of tasks

We have assumed that, if the CICSplex is running at maximum capacity (which is up to 200 tasks, controlled by the MXT values set for the terminal-owning regions.), the file-owning region should be capable of supporting the same number. Thus we have set the maximum number of tasks to 200, allowing the file-owning region to have up to 200 active mirror tasks.

When setting the MXT parameter for your file-owning regions, remember that in an XCF/MRO CICSplex it has to support all the application-owning regions.

### The Queue-Owning Region's Parameters

It is unlikely that you will have many queue-owning regions in a CICSplex, but even if you have only one you should plan to define the system initialization parameters in the SYSIN data set in a similar to the file-owning region.

## Example

An example of the system initialization parameters that could be defined as common to all the queue-owning regions in a CICSplex are shown in Figure 41 on page 156.

```

* *****
* CICS system initialization parameters common to all cloned QORs *
* *****
AIQMAX=0           No autoinstall terminals in this region
CICSSVC=218       The default CICS SVC number
CSDDISP=SHR       Disposition for the CSD
CSDDSN=CICS410.DFHCS D  Data set name of the CSD
DBP=1$           Dynamic backout program - no local DL/I support
DCT=GQ           The global table for all queues in the CICSplex
FCT=NO           No file control table
* CSD Group lists for the QOR
GRPLIST=(IBMLIST*,ISVLIST*,CICSDQ##)
JCT=2$           The CICS sample journal control table
* The IRC & ISC parameters required for MRO
IRCSTRT=YES,     Start IRC during initialization
ISC=YES,         Include the intersystem communication program
MROLRM=YES       Long-running mirrors required
MXT=200          Set maximum tasks to 200 (for mirrors)
RENTPGM=PROTECT Read-only ERDSA required.
PLTPI=DQ         PLT table is DFHPLTDF
SRT=1$          The CICS sample System Recovery Table
TCT=NO           Dummy TCT only required.
TRTABSZ=200      Trace table size

```

Figure 41. Example of Common System Initialization Parameters for a Queue-Owning Region

## Using SYSIN for Common System Initialization Parameters

When you have planned all the common system initialization parameters that you need for the cloned regions, store them for use by CICS in members of a permanent SYSIN data set.

Using a symbolic parameter for part of the member name allows you to use the same JCL procedure for all types of cloned CICS region.

**What about security?:** If you are concerned about operational security for your CICS system initialization parameters, store them in a data set to which access is strictly controlled.

For example, you could place the system initialization parameters for all CICS regions in suitably named members of SYS1.PARMLIB. Alternatively, you could create a CICS.SYSIN data set to which access is equally restricted. If you adopt this approach, SYSIN DD statements for these could be:

```

//SYSIN DD DISP=SHR,DSN=SYS1.PARMLIB(DFHSIP&SIP.)
or
//SYSIN DD DISP=SHR,DSN=CICS.SYSIN(DFHSIP&SIP.)

```

Using the naming conventions described in our target configuration in “Chapter 2. Planning the Subsystem Configuration for a Sysplex” on page 17, the &SIP symbolic parameter represents the CICSplex and type of CICS region. For example, SIP=DT would indicate system initialization parameters for Dallas TORs, and SIP=DA would indicate system initialization parameters ## for the Dallas AORs, and so on. The SIP= parameter is required in the MVS START command for starting all cloned regions.

## Using the PARM Parameter for Unique System Initialization Parameters

There are 3 system initialization parameters that must be unique to each CICS region in a CICSplex. These are the parameters that uniquely identify the region:

### **SYSIDNT**

This system initialization parameter specifies the local name of a CICS region—the name by which CICS identifies itself. The name is displayed on screens in CICS-supplied transactions as SYSID=aaaa.

### **APPLID**

This system initialization parameter is the VTAM application program name (APPL name) for the CICS region. The same name is used to identify the region to MRO, so even if the CICS region is not defined to VTAM, which may be the case for application-owning regions in an MRO CICSplex, you must specify an APPLID parameter. A CICS region's APPLID must correspond to the NETNAME in the connection definitions of its MRO partners.

### **MNSUBSYS**

For CICS regions running in a sysplex that is *not* using goal-oriented workload management, this specifies the 4-character name to be used as the subsystem identification in the monitoring SYSEVENT class records.

If you do not specify a name, the subsystem identification defaults to the first four characters of the CICS generic APPLID.

For more information on the SYSEVENT class of monitoring data and the subsystem identification, and about the implications for SYSEVENT recording in a MVS workload management environment, see the *CICS Performance Guide*.

By specifying these identifiers as symbolic parameters in the PARM parameter in the CICS startup JCL, you can use the same procedure for all the CICS regions in the CICSplex. This is illustrated in Figure 42, which includes a JOB statement as the first entry in the procedure. The JOB statement enables this procedure to use the MVS JOB support for started tasks.

```
//CICSTASK JOB (accounting info),CLASS=X
//CICSTASK PROC START='AUTO',
//          INDEX1='CTS110.CICS510',
//          REGION='32M',
//          SYSIDNT=' ',
//          SIP=' '
//*
//* INDEX1 - High-level qualifier of CICS system data sets
//* REGION - Size of MVS region
//* START - Type of CICS start-up
//* SYSIDNT - Local system identifier of CICS region
//*
//*****
//***** EXECUTE CICS *****
//*****
//INITCICS EXEC PGM=DFHSIP,REGION=&REGION,
//          PARM=('START=&START,APPLID=CICS&SYSIDNT',
//              'INITPARM=(DFHDBCON='XX,&DBCTLID.'')',
//              'SYSIDNT=&SYSIDNT,SYSIN')
```

Figure 42. Defining the CICS System Initialization Parameters in the PARM Parameter

### **Specifying the DBCTL Subsystem Name**

The example of a procedure for starting a CICS region, shown in Figure 42, uses the INITPARM parameter for passing the name of the DBCTL subsystem to which

the CICS region is to connect. It also passes the suffix of the database resource adaptor (DRA) table. If the region does not need to connect to DBCTL, the INITPARM parameter is ignored.

---

## The CICS System Data Sets

Each CICS region requires a number of system data sets without which CICS cannot function. (In general, the data sets cannot be shared between the CICS regions. Two exceptions are the CICS system definition (CSD) data set, which contains resource definitions, and the DFHRPL library concatenation, which defines shared program libraries.)

Some of the CICS system data sets could possibly be omitted from some types of region, on the grounds that a particular resource manager region does not use that data set.

For example, it is possible that a terminal-owning region will never need CICS dump data sets, because it does not execute general user applications. However, it is probable that it will run a number of special user-written application programs, performing some system programmer functions. From time to time these could fail and cause CICS to write a transaction dump.

In our standard CICS startup procedure shown in Figure 43 on page 159, we have included DD statements for all the CICS system data sets. However, you might want to vary the size of the data sets to suit the particular needs of the type of region.

---

## Defining the CICS Startup Procedure for Started Jobs

The procedure shown in Figure 43 on page 159 is designed to be used to start all the CICS regions in a sysplex. Using a single procedure keeps the maintenance of JCL to a minimum.

You are recommended to use the START command, and the JOBNAME parameter, to invoke your procedure for starting each region. The JOBNAME parameter overrides the job name specified in the procedure, ensuring a unique job name for each CICS region. To ensure that the job names are unique across the sysplex, use the &SYSCLONE symbolic parameter as part of the job name.

For example, you can issue the following command to start a CICS TOR in MVSA, where &SYSCLONE is defined as the letter A.

```
START  CICSTASK, JOBNAME=CICSDT&SYSCLONE.1, SYSIDNT=DT&SYSCLONE.1, SIP=T
```

In this example, the &SYSCLONE symbol resolves to the letter A, which corresponds to the letter A in MVSA. If the START command is routed to another MVS image, &SYSCLONE resolves to the character defined for that MVS. Note that you can define the SYSCLONE symbolic parameter as 1 or 2 characters, but the CICS naming convention uses 1 character only (the recovery group) to represent the MVS image. If &SYSCLONE is defined for other purposes as a 2-character symbol, you can define a user symbol for CICS startup JCL.

```

//CICSJOBS JOB (accounting information),MSGLEVEL=(1,1),MSGCLASS=A
//CICSTASK PROC START='AUTO',
// INDEX1='CTS110.CICS510',
// REGION='32M',
// DBCTL=' ',
// DB2=' ',
// OUTC='* ',
// SYSIDNT=' ',
// RDRC='A',
// CSDIND='CICSD###',
// SIP='
//*
//*****
//***** EXECUTE CICS *****
//*****
//*
//INITCICS EXEC PGM=DFHSIP,REGION=&REG,TIME=1440,
// PARM=('START=&START,APPLID=CICSHT61,SYSIDNT=&SYSIDNT,SYSDNT',
// 'INITPARAM=(DFHDBCON='XX,&DBCTL',DSN2STRT='D#,&DB2')')
//*
//SYSIN DD DISP=SHR,DSN=&INDEX1..CICSD###.SYSIN(DFHSIP&SIP.)
//*****
//* THE CICS LIBRARY CONCATENATIONS
//STEPLIB DD DISP=SHR,DSN=&INDEX1..SDFHAUTH
//*****
//* THE CICS LIBRARY (DFHRPL) CONCATENATION
//DFHRPL DD DISP=SHR,DSN=&INDEX1..SDFHLOAD
// DD DISP=SHR,DSN=SYS1.COBOL2.V132.COB2CICS
// DD DISP=SHR,DSN=SYS1.COBOL2.V132.COB2LIB
// DD DISP=SHR,DSN=PP.C3700512.V120CICS.LOADLIB
// DD DISP=SHR,DSN=PP.PLI.V23.PLILINK
//*****
//* THE AUXILIARY TEMPORARY STORAGE DATASET
//DFHTEMP DD DISP=SHR,DSN=&INDEX1..CNTL.CICS&SYSIDNT..DFHTEMP
//*****
//* THE INTRAPARTITION DATASET
//DFHINTRA DD DISP=SHR,DSN=&INDEX1..CNTL.CICS&SYSIDNT..DFHINTRA
//*****
//* THE AUXILIARY TRACE DATASETS
//DFHAUT DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHAUT
//DFHBUT DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHBUT
//*****
//* THE DUMP DATASETS
//DFHDMPA DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHDMPA
//DFHDMPB DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHDMPB
//*****
//* THE CICS SYSTEM LOG DATASETS
//DFHJ01A DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHJ01A
//DFHJ01B DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHJ01B
//DFHJ01X DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHJ01X

```

Figure 43. A Sample CICS Startup Procedure for All CICS Regions in a CICSplex (Part 1 of 2)

```

//*****
//*      AUTOMATIC JOURNAL ARCHIVING DATASETS
//DFHJACD DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHJACD
//DFHJPDS DD DISP=SHR,DSN=&INDEX1..CHCSD###.DFHJPDS
//DFHJOUT DD SYSOUT=(&RDRC,INTRDR)
//*****
//*      THE RESTART DATASET
//DFHRSD DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHRSD
//*****
//*      THE CICS LOCAL CATALOG DATASET
//DFHLCD DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHLCD
//*****
//*      THE CICS GLOBAL CATALOG DATASET
//DFHGCD DD DISP=SHR,DSN=&INDEX1..CICS&SYSIDNT..DFHGCD
//*      AMP=( 'BUFND=5,BUFNI=20,BUFSP=122880' )
//*****
//* EXTRAPARTITION DATASETS
//DFHCXRF DD SYSOUT=&OUTC
//MASTOUT DD SYSOUT=&OUTC,DCB=(DSORG=PS,RECFM=V,BLKSIZE=136)
//BATCHRDR DD SYSOUT=(&RDRC,INTRDR)
//LOGUSR DD SYSOUT=&OUTC,DCB=(DSORG=PS,RECFM=V,BLKSIZE=136)
//MSGUSR DD SYSOUT=&OUTC,DCB=(DSORG=PS,RECFM=V,BLKSIZE=136)
//PLIMSG DD SYSOUT=&OUTC,DCB=(DSORG=PS,RECFM=V,BLKSIZE=137)
//COUT DD SYSOUT=&OUTC,DCB=(DSORG=PS,RECFM=V,BLKSIZE=137)
//* LE/370 QUEUES
//CEEMSG DD SYSOUT=&OUTC
//CEEOUT DD SYSOUT=&OUTC
//*****
//* CAFF TD QUEUE FOR ERROR MESSAGES
//CAFF DD SYSOUT=&OUTC
//*****
//TRACEOUT DD SYSOUT=&OUTC
//SYSABEND DD SYSOUT=&OUTC
//PRINTER DD SYSOUT=&OUTC,DCB=BLKSIZE=121
//CARDIN DD DISP=SHR,DSN=&INDEX1..CHCSD###.SYSIN(CARD&SIP.)
//      PEND
//      EXEC CICSTASK
//* END OF CICS START-UP JOB

```

Figure 43. A Sample CICS Startup Procedure for All CICS Regions in a CICSplex (Part 2 of 2)

**Notes on Sample Procedure:**

1. The JOB statement (for JOB support for started tasks) must be the first statement in the procedure.
2. The terminal-owning regions and the application-owning regions in our CICSplex both use CICS sequential terminal support to start transactions that are needed at startup. Although these could be started by a PLTPI-specified program, it is sometimes easier to use the CARDIN data set, defined in the CICS sample terminal control table as a sequential terminal. The CARDIN DD statement in our sample job uses the & symbol because the data sets are different for terminal-owning regions and application-owning regions.

## Starting the CICS Regions

The following start command, issued to start the tasks listed in the procedure CICSRUN shown in Figure 44 on page 161, starts a set of CICS regions in each MVS to which the start command is directed:

```
ROUTE mvid START CICSRUN
```

```

//*****
//*  PROCEDURE TO START CICS REGIONS AS STARTED JOBS  *
//*  RACF PROFILES SUPPORTING THESE START COMMANDS ARE:
//*
//*      STARTED PROFILES NAMED CICSTASK.CICSDA*
//*              CICSTASK.CICSDF*
//*              CICSTASK.CICSDT*
//*
//*      AUTHORIZED TO RUN UNDER USERIDS CICSDA##
//*              CICSDF##
//*              CICSDT##
//*
//*****
//CICSRUN PROC
//*
//DUMMY EXEC PGM=IEFBRI4
//*
// START  CICSTASK,JOBNAME=CICSMT&SYSCLONE.1,
//        SYSIDNT=HT&SYSCLONE.1,SIP=T
//*
// START  CICSTASK,JOBNAME=CICSDA&SYSCLONE.1,
//        SYSIDNT=HA&SYSCLONE.1,SIP=A
//*
// START  CICSTASK,JOBNAME=CICSDA&SYSCLONE.2,
//        SYSIDNT=HA&SYSCLONE.2,SIP=A
//*
// START  CICSTASK,JOBNAME=CICSDA&SYSCLONE.2,
//        SYSIDNT=HA&SYSCLONE.2,SIP=A
//*
// START  CICSTASK,JOBNAME=CICSDA&SYSCLONE.2,
//        SYSIDNT=HA&SYSCLONE.2,SIP=A

```

Figure 44. Procedure to Start 5 CICS Tasks, 1 for Each CICS Region

Using this technique, you need only one CICS startup procedure, such as CICSTASK, and only one procedure to issue all the START commands.



---

## **Part 3. Migrating IMS Applications**

This explains what changes to make to your IMS environment to migrate to a Sysplex environment.

## Migrating IMS TM Applications

---

## Chapter 18. Planning for IMS TM in a Sysplex Environment

For most customers, IMS TM can run in a Parallel Sysplex environment with *no* changes to their current IMS environment, except those necessary to use multisystem data sharing or to share message queues.

Some customers may have to make changes to their current IMS environment in order to migrate to the Parallel Sysplex. For them, we make the following recommendations:

- Set up your IMS RESLIBs so you can clone your IMS subsystems across the Parallel Sysplex.
- Ensure that the IMSID is unique for each IMS subsystem in the Parallel Sysplex so you can move your IMS subsystem to another MVS image if necessary.
- Ensure that all terminal names, LU names, and ETO user IDs in your network are unique.
- Divide your network to balance your workload and to minimize network outage if you lose one of your IMS subsystems.
- Convert batch jobs to BMP programs, especially in a data sharing environment, to minimize the number of connections to the coupling facility.

See the IMS Version 5 or Version 6 library for more detail on the information given below.

---

### Cloning Your IMS Subsystems

While you cannot clone the system definition from a “master” IMS system generation to other IMS subsystems, you can use many of the definitions from one IMS subsystem as defaults for common resources cloned to other IMS subsystems. Each IMS subsystem, however, must have a unique master terminal (MTO) and secondary master terminal. There are also other definitions that must be unique, such as Multiple Systems Coupling (MSC) definitions.

In order to clone IMS subsystems across a Parallel Sysplex, first determine how many unique IMS RESLIB data sets you need. That is, determine what IMS definitions need to be different for each IMS subsystem that cannot be overridden during execution. Then perform IMS system definition stage 1 to create your RESLIBs and put them on shared DASD. Use as many defaults or common coding as possible during stage 1 so you can override these definitions during execution.

As an example of a default definition overridden during execution, let IMS set APPLID=(IMS,IMS2,IMS3) in the COMM macro, then override IMS2 in the IMS procedure using APPLID2=IMSB.

To minimize the number of unique stage 1 definitions you need, consider using the IMS Extended Terminal Option (ETO) to reduce the number of LTERM names in your stage 1.

See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for more information.

### What to Share between IMS Subsystems in a Parallel Sysplex

Table 5 on page 166 lists the IMS data sets that should be shared between IMS subsystems in a Parallel Sysplex, and those that should be unique for each

subsystem. All data sets, both shared and nonshared, should be shared DASD so you can move your IMS subsystems across the Parallel Sysplex. In order to ensure uniqueness for the data sets, name them using the IMSID as the data set high-level qualifier.

Table 5. IMS Data Sets in a Parallel Sysplex

Shared data sets	Nonshared data sets	Comments
ACBLIBn DBDLIB FORMATn IMSTFMT MATRIXn MODBLKSn PGMLIB PROCLIB PSBLIB RECONn RESLIB	DFSOLDSn DFSOLPnn DFSWADSn IEFRDRn IMSMON LGMSG MODSTAT MSDBCPn MSDBDUMP MSDBINIT QBLKS RDSn SHMSG	ACBLIB contents should be identical across the Parallel Sysplex even if you have multiple ACBLIBs. Likewise, PSBLIB and DBDLIB contents should be identical across the Parallel Sysplex even if you have multiple PSBLIBs and DBDLIBs.  For performance reasons, you might want identical FORMATn and PGMLIB data sets.  If you use MSC, you cannot share the MATRIXn, MODBLKSn, or RESLIB data sets because they contain information about each unique MSC link.
RLJCLLIB RLRESLIB		IRLM data sets

It is possible to share the IMSMON data set serially, that is, dynamically allocate it for each IMS subsystem in the Parallel Sysplex. Be sure to activate the IMS monitor for only one IMS subsystem at a time, and deallocate the data set before activating the monitor on another IMS subsystem.

In order to make cloning possible, ensure that all data is accessible across the Parallel Sysplex and that your hardware connectivity is symmetric.

---

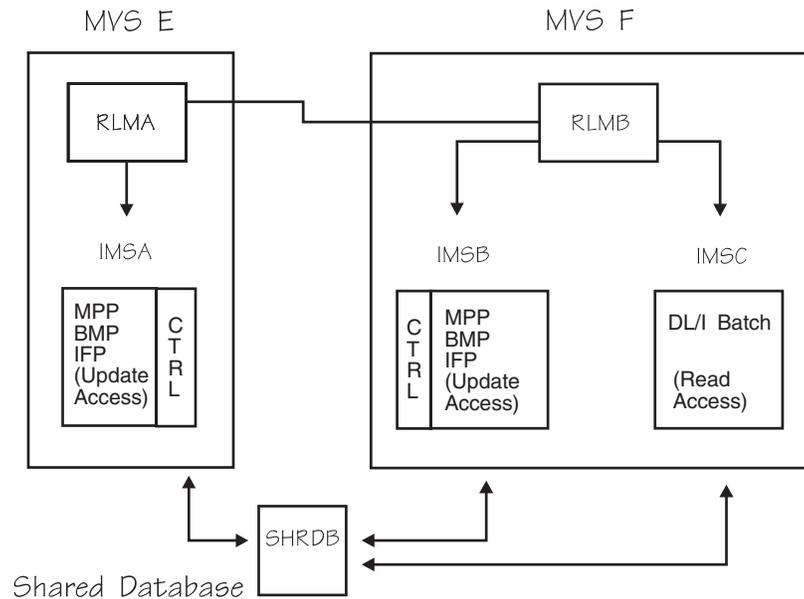
## Ensure Unique IMSIDs

The IMSID is the IMS subsystem identifier. For online control regions, it must be different from any other IMSID or non-IMS subsystem identifier defined to the MVS under which IMS is running. The IMSID is also used to relate messages that are routed to the MVS system console to the corresponding IMS subsystem.

The IMSID specified in the IMSCTRL macro (part of stage 1 of IMS system definition) can be overridden at execution by specifying a keyword in the DFSPBxxx member or a parameter on the EXEC statement.

Having unique IMSIDs lets you move your IMS subsystems to another MVS image when necessary. Having unique IMSIDs also allows MVS automatic restart management to restart a failing IMS on any MVS system or CPC in the Parallel Sysplex without conflicting with any other IMS subsystems.

The following example shows how the IMSID fits into the overall picture of IMS in a data sharing environment.



#### IMSA

**IMS system definition statements include:**

```

IMSCTRL  IMSID=IMSA,DBRC=YES,IRLMNM=RLMA      X
          SYSTEM=(VS/2,(ALL,DB/DC),5.1)
DATABASE DBD=SHRDB,ACCESS=UP
  
```

**PSB generation statements include:**

```

PCB      TYPE=DB,DBDNAME=SHRDB,PROCOPT=A
  
```

#### IMSB

**IMS system definition statements include:**

```

IMSCTRL  IMSID=IMSB,DBRC=YES,IRLMNM=RLMB,    X
          SYSTEM=(VS/2,(ALL,DB/DC),5.1)
DATABASE DBD=SHRDB,ACCESS=UP
  
```

**PSB generation statements include:**

```

PCB      TYPE=DB,DBDNAME=SHRDB,PROCOPT=A
  
```

#### IMSC

**IMS system definition statements include:**

```

IMSCTRL  IMSID=IMSC,DBRC=YES,IRLMNM=RLMB,    X
          SYSTEM=(VS/2,BATCH)
DATABASE DBD=SHRDB,ACCESS=RD
  
```

**PSB generation statements include:**

```

PCB      TYPE=DB,DBDNAME=SHRDB,PROCOPT=GOT
  
```

If MVS A terminates, you (or MVS automatic restart management) can bring up IMSA on another MVS image, within the same CPC or a different CPC. By having unique IMSIDs, you need not worry if there is already an IMS subsystem running on the new MVS image, because that IMS will have a different IMSID.

---

## Ensure Unique Terminal Names, LU Names, and User IDs

Just as with the IMSID, you should ensure that your logical unit names (LU names) and logical terminal (LTERM) names are unique across your IMS network. And you should make your ETO user IDs unique across the Parallel Sysplex if they become LTERMs (if they don't, the user IDs need not be unique). With unique names, you can move any terminal, LU 6.2 application, or user anywhere in the network without your having to worry about duplicate names (which could result in a security exposure in your network if data destined for one LU name, for example, was delivered to a different LU name).

The following is an example of defining an LU name for IMS to VTAM. In the example, IMS is given the SNA LU name SE40IMS.

```
SE40IMS APPL ACBNAME=SE40IMS,AUTH=(ACQ),EAS=100,PARSESS=YES
```

A logical terminal is a symbolic destination that maps to a VTAM node or a BTAM physical terminal. Each logical terminal has an installation-defined name, an LTERM name. You should define a convention for naming LTERMs. One method is to have the LTERM name consist of a combination of the workstation and component identifications. For example, IMS considers the 4704 keyboard and display system components as one component, the 4706 magnetic stripe reader as another component, and the 4710 receipt printer as yet another component. Thus, workstation 100 could have three components: WS100DS (the 4704), WS100MS (the 4706), and WS100RP (the 4710). Such a convention would permit an application running in a message processing program (MPP) region to interrogate the I/O PCB (LTERM name field) to identify the workstation and then be able to specify the proper alternate PCB for output using the CHNG call.

Each IMS LTERM is associated with one input and one output IMS component. The input and output components can be the same component, or different components can be specified. Conversely, IMS does not prevent multiple input or output LTERMs from being associated with a single component. However, doing so may cause problems with determining input components or presenting output.

You can establish relationships between input and output components by using the NAME macro during IMS system definition. Using this macro allows the terminal to indicate its input component and causes output to be returned to a component indicated during IMS system definition. Proper definition and use of input components can reduce or eliminate the need for LTERM naming conventions, MPP change calls, and inserts to alternate PCBs.

APPC/IMS does not use an LTERM for input. APPC/IMS provides two facilities that are similar to LTERMs for output:

- LU 6.2 descriptors. Using LU 6.2 descriptors allows an IMS LTERM name to define an LU 6.2 destination (specifying LUNAME, TPNAME, and other LU 6.2 destination characteristics). These LU 6.2 LTERMs can be used in the same way as regular LTERMs in IMS applications.
- Side information, a symbolic destination name that contains system-defined values provided by CPI communications.

---

## Divide Your Network

To minimize network outage if one of your IMS subsystems should fail, you should divide your network and transaction traffic. One way to accomplish this division is to use IMS Multiple Systems Coupling (MSC). Another way is to use shared message queues (see “Chapter 19. Planning for IMS/ESA Version 6 in a Parallel Sysplex Environment” on page 175).

MSC connects geographically dispersed IMS systems in a way that allows programs and operators of one system access to programs and operators of the connected systems. Communication is permitted between two or more (up to 255) IMS systems running on any supported combination of MVS systems, including the Parallel Sysplex.

## Advantages of MSC

MSC can extend the throughput of an IMS system beyond the capacity of a single processor. This is possible if the IMS applications can be partitioned among systems such that either:

- Applications execute in more than one system with database contents split between systems (horizontal partitioning).<sup>1</sup>
- Applications execute in one system (or Parallel Sysplex) with the complete database that they reference attached to that system (vertical partitioning); the transactions can originate in any system.

A transaction originates in what is called the local system and can be processed in either the local system or a remote system. When a transaction is processed in a remote system, the input LTERM name in the local system is carried over as part of the message. If the processing program uses an alternate PCB to direct messages to other than the input terminals, the destinations need to be declared as remote unless directed routing is used. The technique is to define the LTERM names for all input terminals with NAME macros. You place the NAME macros in a group after the MSNAME macro. Then you have a set of LTERMs that collectively can occur in several system definitions. For example, TERMA can be present in the input system, in an intermediate system, and in the processing system.

You do not have to declare every terminal in the remote system that is entering transactions, just those that enter traffic destined for this local system. If the LTERM in the remote system is for an ETO terminal that enters transactions destined for this system, define the LTERMs using ETO MSC descriptors instead of NAME macros.

## Planning for MSC

When planning for the network, keep in mind that message queues for an input system or an intermediate system have to allow for the remote transactions to be enqueued. The message lengths and their expected loads must be taken into account when allocating space for the message queues. Similarly, you have to allow for the presence of these messages in I/O buffers, even though they are not going to be processed in that system.

---

1. Horizontal partitioning is beyond the scope of a Parallel Sysplex, but could be important if you have geographically remote IMS subsystems in your network.

Although MSC physical and logical links continue to be predefined through IMS system definition, using ETO you can dynamically create MSC remote LTERMs during IMS initialization. ETO also allows you to associate one or more message queues with an MSC logical link.

Each IMS system must be assigned at least one unique system identifier (SYSID) in the MSC system definition process. This SYSID is a local SYSID for the owning system and a remote SYSID for any other IMS system that has a path to this system.

When planning for network availability, remember that MSC does not increase the availability of a single IMS subsystem, but does help increase the availability of your overall IMS network.

Finally, here are some major design considerations for MSC:

- Minimizing resource consumption by defining suitable connections between the systems
- Balancing resource demand by distributing functions among systems to obtain optimal performance
- Program design for multisystem conversational transactions

## Workload Balancing Using MSC

In a Parallel Sysplex especially, you can use MSC to balance your workload across the Parallel Sysplex as well as to minimize network outages. You can define an IMS transaction in a Parallel Sysplex so that no matter where a transaction originates, it always runs on a particular IMS subsystem.

You define each IMS transaction as either local or remote. When a transaction reaches an IMS subsystem, IMS checks to see if the transaction is local or remote; if local, IMS processes the transaction; if remote, IMS passes the transaction along to another IMS subsystem across its MSC links. For example, TRANA can be defined to run on IMSA; if TRANA originates on IMSB, the transaction is processed on IMSA because IMSB knows that TRANA is remote and sends it to IMSA (for which TRANA is local).

You can use the IMS Input Message Routing exit routine to balance workload dynamically by allowing the exit routine to determine whether to process a local transaction or to send it to another IMS subsystem for processing. This means that you can define all your transactions as local across the Parallel Sysplex and let the exit routine route each transaction dynamically to any IMS subsystem.

You can also use the IMS Program Routing exit routine to allow your application programs to dynamically balance their workload by using program-to-program switches to reroute work to any IMS subsystem in the Parallel Parallel Sysplex.

In IMS Version 6, you can also perform workload balancing using VTAM generic resources or shared message queues. For more information on using VTAM generic resources, see “Planning for Using VTAM Generic Resource Groups” on page 178. For more information on using shared message queues, see “Planning for a Shared-Queues Environment” on page 176.

## Flow of Data within Multiple Systems

The flow of a transaction in a multisystem environment requires additional steps than those in a single system environment. These steps are illustrated in Figure 45 on page 172, and explained as follows:

- In the input terminal system, a remote transaction entered from a terminal **(1)** is placed on the message queue **(2)** of the terminal system with the destination of the remote transaction name. The message is queued to the MSNAME associated with the specified remote destination.
- MSC support is responsible for removing the message from the local message queue **(3)**, sending it across the MSC link **(4)**, and placing it on the message queue of the processing system **(5)** to be processed by the application program **(6)**.
- Subsequently, when the application program processes the message and sends a reply back to the originating IMS system as indicated by the SYSID of the message.
- The reply message **(7)** is placed first on the message queue in the processing system with a destination of the input LTERM **(8)**.
- MSC support is responsible for removing the message from the message queue **(9)** and sending it back across the MSC link **(10)**.
- MSC places the message on the message queue of the terminal system **(11)**, and sends it to the input terminal **(12)**.

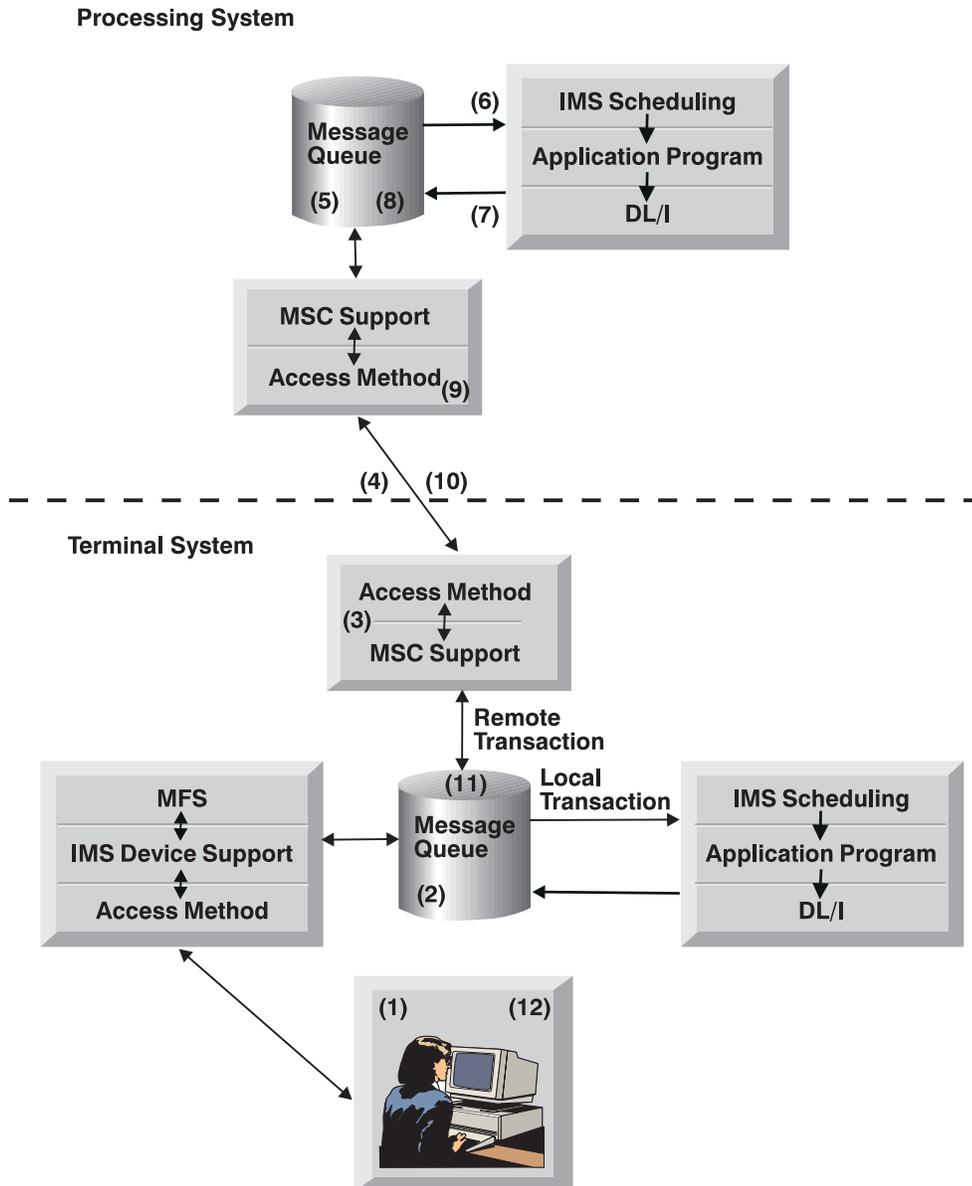


Figure 45. Multiple IMS Systems Transaction Flow

## Convert Batch Jobs to BMP Programs

You should convert batch programs to batch-oriented batch message processing (BMP) programs. Each batch program in a multisystem data sharing environment requires a separate connection to the coupling facility; all BMP programs running in one IMS subsystem require only one connection to the coupling facility. Converting a batch program to a batch-oriented BMP program can be advantageous for the following reasons as well:

- BMPs can send output to the message queues.
- BMPs can access IMS data entry databases (DEDBs) and IMS main storage databases (MSDBs), as well as IMS full-function DL/I databases.

- BMPs simplify program recovery because logging goes to a single system log.<sup>2</sup>
- Restart can be done automatically from the last checkpoint without changing the JCL.

Remember, if you are running in the DBCTL environment, BMPs cannot send output to the message queues and cannot access MSDBs.

Converting batch programs to BMP programs enables you to reduce the “batch window” because you can run the BMP programs with your online workload, instead of having to wait until the online system is not running. This also allows you to keep the data more current.

If you use data sharing, you can run batch programs concurrently with online programs. You can also use BMP programs as an alternative to sharing data between batch programs and your online programs.

**Recommendation:** Code your checkpoints in a way that makes them easy to modify. Converting a batch program to a BMP or converting a batch program to use data sharing requires more frequent checkpoints to minimize the time needed for backout and recovery if the program fails. More frequent checkpoints can also reduce the number of database resources dedicated to one application program at any one time.

There are two requirements for a batch program to be converted to a BMP:

- The program must have an I/O PCB. You can obtain an I/O PCB in batch by specifying the compatibility (CMPAT) option in the program specification block (PSB) for the program.
- BMPs must issue checkpoint calls more frequently than batch programs.

Finally, if a program fails while running in a batch region, you must restart it in a batch region. If a program fails in a BMP region, you must restart it in a BMP region.

---

## MVS Resource Management

IMS support for the MVS Workload Manager (WLM) helps MVS balance the workload mix and prioritization to meet your business objectives. IMS provides MVS with information about the status of IMS units of work (UOWs). MVS uses this information to decide how to allocate its computing resources.

IMS assists the WLM by helping to establish monitoring environments for the UOWs that are under IMS control. The monitoring environments are performance blocks that are built by MVS. IMS initializes these performance blocks with the service classification for the UOW, the time the UOW started, and whether the UOW is active or waiting. IMS then supplies the monitoring environment with information such as total response time and any delays associated with a UOW.

The WLM monitors the UOWs while they are processing by periodically sampling the monitoring environments. If there is contention for resources among the UOWs, the WLM dynamically adapts the resources to ensure that the most critical work completes on time, while less critical work is deferred. The WLM does this by

---

2. If your installation uses direct access storage for the system log in batch, you can specify that you want dynamic backout for the program. In that case, batch recovery is similar to BMP recovery except, of course, with batch you need to manage multiple logs.

referring to the user-defined objectives for each type of UOW. These objectives may be a desired response time or other performance-related goals.

---

## **Availability and Recovery**

In a Parallel Sysplex, if one processor or MVS system fails or becomes unavailable, you can restart your IMS subsystems on another processor or MVS system. If you have ensured unique IMSIDs, terminal and LU names, and user IDs, and have put nonshared data sets on shared DASD, you can restart the failed IMS anywhere in the Parallel Sysplex. The restarted IMS subsystem can resume its normal workload, especially if you are using multisystem data sharing.

---

## Chapter 19. Planning for IMS/ESA Version 6 in a Parallel Sysplex Environment

This chapter describes how to plan for IMS/ESA Version 6 in a Parallel Sysplex environment.

“Parallel Sysplex Migration Requirements”

“Planning for a Shared-Queues Environment” on page 176

“Planning for Using VTAM Generic Resource Groups” on page 178

“Planning for OSAM Database Cache Migration” on page 180

“Planning for Shared SDEPs Migration” on page 180

---

### Parallel Sysplex Migration Requirements

Migrating IMS to a Parallel Sysplex environment enables you to use of a number of the new Version 6 enhancements in addition to those available in Version 5.

#### Migrating IMS

The following is a sample migration path for IMS to the Parallel Sysplex environment:

1. Migrate the MVS level to MVS SP 5.1 (or higher) or OS/390 (or higher).
2. Upgrade your hardware to that required for the Parallel Sysplex environment (for example, Parallel Sysplex Timer and coupling facility).
3. Define your coupling facility structure sizes to be 512 kilobytes or more. This size allows you to use both primary and optional overflow structures. Create structures for each of the following IMS functions:
  - Data sharing
  - Shared message queues and shared Expedited Message Handler (EMH) queues
  - OSAM database cache
  - Shared VSO DEDB areas
  - Shared SDEPs

**Recommendation:** Place high-use structures on separate coupling facilities. For example, your IMS data sharing structures should be on a different coupling facility from your shared-queue structures. Likewise, your DB2 and VSAM structures should be on separate coupling facilities. You can place low-use structures (like RACF) on any coupling facility that has enough space.

4. Bring up the Parallel Sysplex environment.
5. Update IRLM start-up procedures to remove VTAM dependencies and to define the LOCK structure and group name (for data sharing).
6. Start up IMS data sharing with IMS Version 4 or Version 5 using IRLM 2.1 and a coupling facility.
7. Migrate the IMS level to Version 6.
8. When all the IMS systems are at the Version 6 level, you can start the OSAM/VSAM structures and begin sharing messages.

#### Migrating Your Data-Sharing Environment

When implementing Parallel Sysplex data sharing, you should review your naming conventions for IMS log data sets. Troubleshooting and recovery can be simplified if log data sets can be tied to the IMS subsystem that created them.

The following are two possible paths for migrating from nonParallel Sysplex to Parallel Sysplex data sharing (or from IRLM 1.5 to IRLM 2.1):

1. Migration Path 1
  - a. Migrate to MVS SP 5.1 or later, or OS/390 or later.
  - b. Use IMS Version 4, or Version 5 with the IRLM 1.5. This is the nonParallel Sysplex data sharing environment; no structure names are specified. Maintain your existing version of IRLM until IRLM 2.1 is running. This permits easy backing out if any problems occur with the migration from IRLM 1.5 to 2.1.
  - c. Use IMS Version 4, or Version 5 with IRLM 2.1. Specify IRLM structure names only.
  - d. Use IMS Version 6 with IRLM 2.1. Specify IRLM, OSAM, and VSAM structure names.
2. Migration Path 2
  - a. Migrate to MVS SP 5.1.
  - b. Use IMS Version 6 with IRLM 1.5. This is the nonParallel Sysplex data sharing environment; no structure names are specified. Maintain your existing version of IRLM until IRLM 2.1 is running. This permits easy backing out if any problems occur with the migration from IRLM 1.5 to 2.1.
  - c. Use IMS Version 6 with the IRLM 2.1. Specify IRLM, OSAM, and VSAM structure names.

---

## Planning for Migration to IMS Version 6

This section describes how to plan for migrating to the major new functions of IMS Version 6.

### Planning for a Shared-Queues Environment

Operating in a shared-queues environment allows multiple IMS subsystems in a Parallel Sysplex environment to share IMS message queues and Expedited Message Handler (EMH) message queues. A shared-queues environment provides you a single-image view of multiple IMS sys and distributes processing loads between the IMS subsystems. Transactions entered on one IMS system can be made available on the shared queues to any other IMS system capable of processing them. Results of these transactions are then returned to the initiating terminal. End users do not need to be aware of these activities; their view of processing is as if they were operating in a single-system environment.

### Migrating to a Shared-Queues and Shared-EMH Environment

When migrating to a shared-queues and shared-EMH environment, do the following:

- Ensure that you have installed the required software. See “Required Components of a Shared-Queues Environment” on page 177.
- Allocate sufficient space in the coupling facility for the primary and (optional) overflow structures.
- Place the Fast Path and full-function queues in separate coupling facilities.

### Benefits of Using Shared Queues

The major benefits of operating in a shared-queues environment are:

#### **Automatic workload balancing**

A message placed on a shared queue can be processed by any participating IMS system that is available for work.

**Incremental growth**

You can add new IMS subsystems as workload increases.

**Increased reliability**

If one IMS subsystem fails, work that is placed on a shared queue can still be processed by other IMS systems, and with VTAM generic resources, the output can be returned to the originating terminal.

**Recommendations:**

- Enabling shared queues does not require the use of VTAM generic resource groups; however, IBM recommends that you use the two functions together.
- IBM recommends that all data in a Parallel Sysplex be shared across the Parallel Sysplex.

## Required Components of a Shared-Queues Environment

Although you can operate many different configurations of a shared-queues env the required components for shared-queues processing, shown in Figure 46 on page 178, include:

**Common Queue Server (CQS)**

One CQS is required for each client. Each CQS accesses the shared queues, which reside in coupling facility list structures.

**Restriction:** With IMS Version 6, each CQS can have only one client.

**CQS client**

One or more IMS DB/DC or DCCTL subsystems that can access the shared queues using CQS client requests.

**MVS coupling facility list structures**

A type of coupling facility structure that maintains the shared queues.

**MVS system log**

One MVS system log is used for each structure pair. CQS places recovery information about work it has processed and about the list structure pair in the MVS log streams. These log streams are then shared by all CQSs that access the list structure pair.

**CQS checkpoint data set**

One CQS checkpoint data set is maintained for each structure pair of each CQS. The CQS checkpoint data set contains CQS system checkpoint information.

**CQS structure recovery data sets (SRDSs)**

CQS maintains two SRDSs for each structure pair for recovery of the shared queues on the structures. The SRDSs maintain structure checkpoint information for the shared queues.

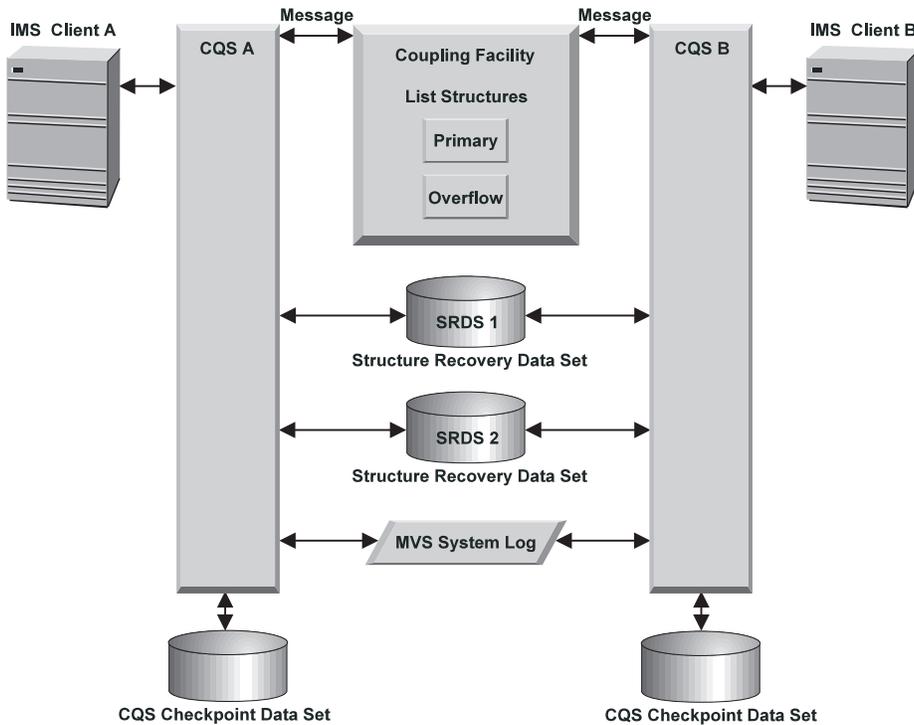


Figure 46. Components of a Shared-Queues Environment

## Planning for the Common Queue Server (CQS)

The Common Queue Server (CQS) is a subsystem which IMS uses to communicate with shared message queues. You can use IMS commands to initiate CQS requests. The CQS address space is started by the IMS subsystem.

CQS performs the following services:

- Notifies registered clients when work exists on the shared queues
- Provides clients with an interface for accessing shared queues and CQS
- Writes CQS system checkpoint information to CQS checkpoint data set
- Writes structure checkpoint information to an SRDS for recovery of a shared-queues list structure
- Provides structure recovery and overflow processing for the shared-queues list structure
- Drives CQS client and user-supplied exit routines
- Provides the Log Print utility, with sample JCL allowing you to print log records from the MVS log.

---

## Planning for Using VTAM Generic Resource Groups

This section presents an overview of VTAM generic resources. For more information on how to use VTAM generic resources with IMS, see *IMS/ESA Administration Guide: Transaction Manager*.

If you are operating in a Parallel Sysplex environment and running multiple IMS systems, you can initiate a session using the name of a VTAM generic resource group. VTAM balances the sessions among generic resource members in a generic

resource group. If you do not require the services of a specific IMS system, initiate the session using a generic resource name, rather than the APPLID name of a specific IMS system.

The benefits of using VTAM generic resource groups include:

**Automatic session workload balancing**

Using VTAM generic resources is complementary to using shared queues; generic resources distributes network traffic among multiple IMS subsystems, while shared queues distributes back-end application workload.

**Single-image resources**

You can access multiple IMS subsystems using a single generic resource name, offering a single-system image while using the resources of many IMS subsystems.

**Enhanced IMS system availability**

In general, if one IMS subsystem fails, you can log onto another IMS subsystem in that generic resource group.

**Exception:** If a terminal is an Intersystem Communication (ISC) terminal, a SLU P terminal, or a Finance terminal, you might not be able to log on to that terminal.

**Share global messages**

You can obtain messages on shared queues from any IMS subsystem in the generic resource group.

**Recommendation:** If a terminal is in conversation or response mode, do not attempt to obtain messages from it.

## Requirements for Using VTAM Generic Resource Groups

To participate in session balancing using generic resource groups, ensure your installation does each of the following:

- Operates the members of the generic resource group in a Parallel Sysplex environment.
- Operates the minimum release of VTAM and OS/390 or z/OS required for non-APPC generic resources or APPC/IMS generic resources.
- Identifies the IMS subsystems that belong to the generic resource group (by specifying the same IMS generic resource name in the GRSNAME execution parameter of each IMS subsystem). You can also use the /START VGRS command, and include the GRSNAME parameter.
- Defines an APPC generic resource name to MVS, as required for LU 6.2 communications. Define this name on the GRNAME parameter of the LUADD statement in the APPC/MVS APPCPMxx member.
- Defines all IMS subsystems that belong to a generic resource group, using equivalent specifications.

**Related Reading:**

- For information on minimum release levels required for non-APPC and APPC generic resources, see *IMS/ESA Release Planning Guide*
- For information on the /START VGRS command, see *IMS/ESA Operator's Reference*.

## Restrictions on Using VTAM Generic Resource Groups

When creating a generic resource group, system programmers should keep in mind:

- The target of an MSC link cannot be a generic resource name.
- IMS XRF subsystems cannot participate as members of a generic resource group; however, they can be members of the same shared-queues group.

---

## Planning for OSAM Database Cache Migration

To plan for OSAM database cache migration, do each of the following:

- Review the CFRM policy to ensure it is valid for data caching.
- Choose which OSAM subpools to define with the caching option, and choose the level of caching to be used.

To enable OSAM database coupling facility caching, the *co* parameter is used to define OSAM subpools, and is now part of the IOBF statement. For a description of the *co* parameter values, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*

You can assign a specific database data set to a specific subpool by defining an identifier (the *id* parameter) on the IOBF and the DBD statements. By specifying those subpools that can be shared by different database data sets (or those subpools that can be used exclusively by a single database data set), you can control subpool usage.

You can control the subpools that participate in data caching by using the *co* parameter. Also, by using an identifier to assign specific database data sets to specific subpools, you can select the database data sets that participate in data caching.

---

## Planning for Shared SDEPs Migration

No special migration issues exist for this function. You can migrate back-and-forth between IMS releases without an unload, reload, or image copy base change.

---

## Planning for Shared VSO DEDB Areas Migration

No special migration issues exist for this function. However, if you choose to migrate your MSDBs to DEDBs or to VSO DEDB areas, see *IMS/ESA Release Planning Guide*.

---

## Part 4. Appendixes



---

## Appendix. Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Mail Station P300  
2455 South Road  
Poughkeepsie, NY 12601-5400  
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Trademarks

The following terms are trademarks of the IBM Corporation in the United States and/or other countries:

AD/Cycle  
Advanced Peer-to-Peer Networking  
APPN  
BatchPipes  
BookManager  
C/370  
CBIPO  
CBPDO  
CICS  
CICSplex  
CICS/ESA  
DATABASE 2  
DB2  
DFSMS  
DFSMSshm  
DFSMS/MVS  
Enterprise Systems Connection Architecture  
ES/9000  
ESCON  
Hardware Configuration Definition  
Hiperbatch  
IBM  
IBMLink  
IMS/ESA  
MVS/DFP  
MVS/ESA  
NetView  
OPC  
OpenEdition  
OS/2

OS/390  
Parallel Sysplex  
Processor Resource/Systems Manager  
PR/SM  
RACF  
RAMAC  
Resource Link  
Resource Measurement Facility  
RMF  
S/390  
SOMobjects  
SP  
Sysplex Timer  
System/390  
SystemPac  
VTAM  
z/OS

Other company, product, and service names might be trademarks or service marks of others.



---

# Glossary

---

## Sources of Terms and Definitions

This glossary includes terms and definitions from:

- The *IBM Dictionary of Computing* New York: McGraw-Hill, 1994.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

---

## Explanation of Cross-References

The following cross-references are used in this glossary:

- Contrast with.** This refers to a term that has an opposed or substantively different meaning.
- See.** This refers the reader to multiple-word terms in which this term appears.
- See also.** This refers the reader to terms that have a related, but not synonymous, meaning.

## A

**ACDS.** Active control data set.

**ANO/MVS.** Automated Network Operations.

**AOC/MVS.** Automated Operations Control. The licensed program System Automation for OS/390 includes all of the function previously provided by AOC/MVS.

**AOR.** Application-owning region

**APPN.** Advanced Peer-to-Peer Networking.

## B

**basic mode.** A central processor mode that does not use logical partitioning. Contrast with *logically partitioned (LPAR) mode*.

**batch message processing (BMP) program.** An IMS batch processing program that has access to online databases and message queues. BMPs run online, but

like programs in a batch environment, they are started with job control language (JCL).

**batch-oriented BMP program.** A BMP program that has access to online databases and message queues while performing batch-type processing. A batch-oriented BMP does not access the IMS message queues for input or output. It can access online databases, GSAM databases, and MVS files for both input and output.

**BMP.** Batch message processing (BMP) program.

## C

**cache structure.** A coupling facility structure that enables high-performance sharing of cached data by multisystem applications in a sysplex. Applications can use a cache structure to implement several different types of caching systems, including a store-through or a store-in cache.

**cache structure services.** MVS services that enable applications in a sysplex to perform operations such as the following on a coupling facility cache structure:

- Manage cache structure resources
- Store data into and retrieve data from a cache structure
- Manage accesses to shared data
- Determine when shared data has been changed
- Determine whether a local copy of shared data is valid.

**CBPDO.** Custom Built Product Delivery Offering.

**CEC.** Synonym for central processor complex (CPC).

**central processor (CP).** The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

**central processor complex (CPC).** A physical collection of hardware that includes main storage, one or more central processors, timers, and channels.

**CFRM.** Coupling facility resource management.

**channel-to-channel (CTC).** Refers to the communication (transfer of data) between programs on opposite sides of a channel-to-channel adapter (CTCA).

**channel-to-channel adapter (CTCA).** An input/output device that is used by a program in one system to communicate with a program in another system.

**CICS.** Customer Information Control System.

**CICSplex.** A group of connected CICS regions.

**CICSplex SM.** CICSplex System Manager

**CMOS.** Complementary metal-oxide semiconductor.

**COMMDS.** Communications data set.

**complementary metal-oxide semiconductor (CMOS).** A technology that combines the electrical properties of positive and negative voltage requirements to use considerably less power than other types of semiconductors.

**couple data set.** A data set that is created through the XCF couple data set format utility and, depending on its designated type, is shared by some or all of the MVS systems in a sysplex. See also *sysplex couple data set*.

**coupling facility.** A special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex.

**coupling facility channel.** A high bandwidth fiber optic channel that provides the high-speed connectivity required for data sharing between a coupling facility and the central processor complexes directly attached to it.

**coupling services.** In a sysplex, the functions of XCF that transfer data and status between members of a group residing on one or more MVS systems in the sysplex.

**CP.** Central processor.

**CPC.** Central processor complex.

**cross-system coupling facility (XCF).** XCF is a component of MVS that provides functions to support cooperation between authorized programs running within a sysplex.

**CTC.** Channel-to-channel.

## D

**DAE.** Dump analysis and elimination.

**DASD.** Direct access storage device.

**data sharing.** The ability of concurrent subsystems (such as DB2 or IMS DB) or application programs to directly access and change the same data while maintaining data integrity.

**DBCTL.** IMS Database Control.

**DBRC.** Database Recovery Control.

**DB2.** DATABASE 2 for MVS/ESA.

**DB2 data sharing group.** A collection of one or more concurrent DB2 subsystems that directly access and change the same data while maintaining data integrity.

**DB2 PM.** DB2 Performance Monitor.

**DFSMS.** Data Facility Storage Management Subsystem.

**dpAM.** IBM SystemView Data Processing Accounting Manager/MVS.

## E

**EMIF.** ESCON Multiple Image Facility.

**Enterprise Systems Connection (ESCON).** A set of products and services that provides a dynamically connected environment using optical cables as a transmission medium.

**EPDM.** IBM SystemView Enterprise Performance Data Manager/MVS.

**ESCD.** ESCON Director.

**ESCM.** ESCON Manager. The licensed program System Automation for OS/390 includes all of the function previously provided by ESCM.

**ESCON.** Enterprise Systems Connection.

**ETR.** External Time Reference. See also *Sysplex Timer*.

## F

**FOR.** File-owning region.

**frame.** For a System/390 microprocessor cluster, a frame contains one or two central processor complexes (CPCs), support elements, and AC power distribution.

## G

**global resource serialization.** A function that provides an MVS serialization mechanism for resources (typically data sets) across multiple MVS images.

**global resource serialization complex.** One or more MVS systems that use global resource serialization to serialize access to shared resources (such as data sets on shared DASD volumes).

**GSAM.** Generalized Sequential Access Method.

**GTF.** Generalized trace facility.

## H

**Hardware Management Console.** A console used to monitor and control hardware such as the System/390 microprocessors.

**HCD.** Hardware Configuration Definition.

**highly parallel.** Refers to multiple systems operating in parallel, each of which can have multiple processors. See also *n-way*.

## I

**ICMF.** Integrated Coupling Migration Facility.

**image server.** A high-capacity optical storage device or a computer that each computer and image workstation on a network can use to access and retrieve image objects that can be shared among the attached computers and image workstations.

**IMS.** Information Management System.

**IMS DB.** Information Management System Database Manager.

**IMS DB data sharing group.** A collection of one or more concurrent IMS DB subsystems that directly access and change the same data while maintaining data integrity.

**IMS TM.** Information Management System Transaction Manager.

**in-doubt period.** The period during which a unit of work is pending during commit processing that involves two or more subsystems. See also *in-doubt work unit*.

**in-doubt work unit.** In CICS/ESA and IMS/ESA, a piece of work that is pending during commit processing; if commit processing fails between the polling of subsystems and the decision to execute the commit, recovery processing must resolve the status of any work unit that is in doubt.

**integrated operations workstation.** A programmable workstation (PWS) from which an individual can access multiple products to perform a set of tasks, in some cases without knowing which particular product performs a specific task.

**IOCDS.** Input/output configuration data set.

**IOCP.** Input/output configuration program.

**IODF.** Input/output definition file.

**IRLM.** Internal resource lock manager.

**ISPF.** Interactive System Productivity Facility.

## J

**JES2.** Job Entry Subsystem 2.

**JES3.** Job Entry Subsystem 3.

## L

**LIC.** Licensed Internal Code.

**list structure.** A coupling facility structure that enables multisystem applications in a sysplex to share information organized as a set of lists or queues. A list structure consists of a set of lists and an optional lock table, which can be used for serializing resources in the list structure. Each list consists of a queue of list entries.

**list structure services.** MVS services that enable multisystem applications in a sysplex to perform operations such as the following on a coupling facility list structure:

- Read, update, create, delete, and move list entries in a list structure
- Perform serialized updates on multiple list entries in a list structure
- Monitor lists in a list structure for transitions from empty to non-empty.

**lock structure.** A coupling facility structure that enables applications in a sysplex to implement customized locking protocols for serialization of application-defined resources. The lock structure supports shared, exclusive, and application-defined lock states, as well as generalized contention management and recovery protocols.

**lock structure services.** MVS services that enable applications in a sysplex to perform operations such as the following on a coupling facility lock structure:

- Request ownership of a lock
- Change the type of ownership for a lock
- Release ownership of a lock
- Manage contention for a lock
- Recover a lock held by a failed application.

**logical partition (LP).** A subset of the processor hardware that is defined to support an operating system. See also *logically partitioned (LPAR) mode*.

**logically partitioned (LPAR) mode.** A central processor complex (CPC) power-on reset mode that enables use of the PR/SM feature and allows an operator to allocate CPC hardware resources (including central processors, central storage, expanded storage, and channel paths) among logical partitions. Contrast with *basic mode*.

**loosely coupled.** A multisystem structure that requires a low degree of interaction and cooperation between multiple MVS images to process a workload. See also *tightly coupled*.

**LP.** Logical partition.

**LPAR.** Logically partitioned (mode).

## M

**m-image.** The number (m) of MVS images in a sysplex. See also *n-way*.

**massively parallel.** Refers to thousands of processors in a parallel arrangement.

**member.** A specific function (one or more modules/routines) of a multisystem application that is defined to XCF and assigned to a group by the multisystem application. A member resides on one system in the sysplex and can use XCF services to communicate (send and receive data) with other members of the same group.

**microprocessor.** A processor implemented on one or a small number of chips.

**mixed complex.** A global resource serialization complex in which one or more of the systems in the global resource serialization complex are not part of a multisystem sysplex.

**MP.** Multiprocessor.

**MRO.** Multiregion operation.

**MSC.** Multiple Systems Coupling.

**multi-MVS environment.** An environment that supports more than one MVS image. See also *MVS image* and *sysplex*.

**Multiple Systems Coupling (MSC).** An IMS facility that permits geographically dispersed IMS subsystems to communicate with each other.

**multiprocessing.** The simultaneous execution of two or more computer programs or sequences of instructions. See also *parallel processing*.

**multiprocessor (MP).** A CPC that can be physically partitioned to form two operating processor complexes.

**multisystem application.** An application program that has various functions distributed across MVS images in a multisystem environment.

**multisystem environment.** An environment in which two or more MVS images reside in one or more processors, and programs on one image can communicate with programs on the other images.

**multisystem sysplex.** A sysplex in which two or more MVS images are allowed to be initialized as part of the sysplex. See also *single-system sysplex*.

**MVS image.** A single occurrence of the MVS/ESA operating system that has the ability to process work.

**MVS system.** An MVS image together with its associated hardware, which collectively are often referred to simply as a system, or MVS system.

**MVS/ESA.** Multiple Virtual Storage/ESA.

**MVSCP.** MVS configuration program.

## N

**n-way.** The number (*n*) of CPUs in a CPC. For example, a 6-way CPC contains six CPUs.

**NJE.** Network job entry.

## O

**OLTP.** Online transaction processing.

**OPC/ESA.** Operations Planning and Control.

**operating system (OS).** Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible. (T)

## P

**parallel processing.** The simultaneous processing of units of work by many servers. The units of work can be either transactions or subdivisions of large units of work (batch). See also *highly parallel*.

**Parallel Sysplex.** A sysplex that uses one or more coupling facilities.

**partitionable CPC.** A CPC that can be divided into 2 independent CPCs. See also *physical partition*, *single-image mode*, *MP*, *side*.

**physical partition.** Part of a CPC that operates as a CPC in its own right, with its own copy of the operating system.

**physically partitioned (PP) configuration.** A system configuration that allows the processor controller to use both central processor complex (CPC) sides as individual CPCs. The A-side of the processor controller controls side 0; the B-side of the processor controller controls side 1. Contrast with *single-image (SI) configuration*.

**PR/SM.** Processor Resource/Systems Manager.

**processor controller.** Hardware that provides support and diagnostic functions for the central processors.

**Processor Resource/Systems Manager (PR/SM).** The feature that allows the processor to use several MVS images simultaneously and provides logical partitioning capability. See also *LPAR*.

## Q

**QOR.** Queue-owning region.

## R

**RACF.** Resource Access Control Facility.

**RMF.** Resource Measurement Facility.

## S

**SCDS.** Source control data set.

**SDSF.** System Display and Search Facility.

**SEC.** System Engineering Change.

**serialized list structure.** A coupling facility list structure with a lock table containing an array of exclusive locks whose purpose and scope are application-defined. Applications can use the lock table to serialize on parts of the list structure, or resources outside the list structure.

**side.** A part of a partitionable CPC that can run as a physical partition and is typically referred to as the A-side or the B-side.

**single point of control.** The characteristic a sysplex displays when you can accomplish a given set of tasks from a single workstation, even if you need multiple IBM and vendor products to accomplish that particular set of tasks.

**single system image.** The characteristic a product displays when multiple images of the product can be viewed and managed as one image.

**single-image (SI) mode.** A mode of operation for a multiprocessor (MP) system that allows it to function as one CPC. By definition, a uniprocessor (UP) operates in single-image mode. Contrast with *physically partitioned (PP) configuration*.

**single-MVS environment.** An environment that supports one MVS image. See also *MVS image*.

**single-system sysplex.** A sysplex in which only one MVS system is allowed to be initialized as part of the sysplex. In a single-system sysplex, XCF provides XCF services on the system but does not provide signalling services between MVS systems. See also *multisystem sysplex*, *XCF-local mode*.

**SLR.** Service Level Reporter.

**SMF.** System management facilities.

**SMP/E.** System Modification Program Extended.

**SMS.** Storage Management Subsystem.

**SMS communication data set.** The primary means of communication among systems governed by a single SMS configuration. The SMS communication data set (COMMDS) is a VSAM linear data set that contains the current utilization statistics for each system-managed volume, which SMS uses to help balance space usage among systems.

**SMS configuration.** The SMS definitions and routines that the Storage Management Subsystem uses to manage storage.

**SMS system group.** All systems in a sysplex that share the same SMS configuration and communications data sets, minus any systems in the sysplex that are defined individually in the SMS configuration.

**structure.** A construct used by MVS to map and manage storage on a coupling facility. See *cache structure*, *list structure*, and *lock structure*.

**support element.** A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex (CPC).

**symmetry.** The characteristic of a sysplex where all systems, or certain subsets of the systems, have the same hardware and software configurations and share the same resources.

**SYSLOG.** System log

**sysplex.** A set of MVS systems communicating and cooperating with each other through certain multisystem hardware components and software services to process customer workloads. See also *MVS system*, *Parallel Sysplex*.

**sysplex couple data set.** A couple data set that contains sysplex-wide data about systems, groups, and members that use XCF services. All MVS systems in a sysplex must have connectivity to the sysplex couple data set. See also *couple data set*.

**Sysplex Timer.** An IBM unit that synchronizes the time-of-day (TOD) clocks in multiple processors or processor sides. External Time Reference (ETR) is the MVS generic name for the IBM Sysplex Timer (9037).

**system control element (SCE).** Hardware that handles the transfer of data and control information associated with storage requests between the elements of the processor.

**System/390 microprocessor cluster.** A configuration that consists of central processor complexes (CPCs) and may have one or more coupling facilities.

## T

**tightly coupled.** Multiple CPs that share storage and are controlled by a single copy of MVS. See also *loosely coupled*, *tightly coupled multiprocessor*.

**tightly coupled multiprocessor.** Any CPC with multiple CPs.

**TOR.** Terminal-owning region.

**TSCF.** Target System Control Facility. The licensed program System Automation for OS/390 includes all of the function previously provided by TSCF.

## U

**uniprocessor (UP).** A CPC that contains one CP and is not partitionable.

**UP.** Uniprocessor.

## V

**VM.** Virtual Machine.

**VSAM.** Virtual Storage Access Method.

**VTAM.** Virtual Telecommunications Access Method.

## W

**WLM.** MVS workload management.

## X

**XCF.** Cross-system coupling facility.

**XCF PR/SM policy.** In a multisystem sysplex on PR/SM, the actions that XCF takes when one MVS system in the sysplex fails. This policy provides high availability for multisystem applications in the sysplex.

**XCF-local mode.** The state of a system in which XCF provides limited services on one system and does not provide signalling services between MVS systems. See also *single-system sysplex*.

**XRF.** Extended recovery facility.

---

# Index

## A

- AIXIT parameter 150
- AIRDELAY parameter 150
- AIQMAX parameter 150
- AIRDELAY parameter 150
- application-owning region (AOR)
  - clones 10
  - connection definitions 129
  - link security 139
  - parameters for cloning regions 152
  - session definitions 129
  - split from TOR 50
  - target sysplex configuration 20, 21
- applications
  - migration to a sysplex 7, 8
- APPLID parameter
  - startup procedure consideration 157
- availability
  - CICS applications 68

## B

- backup-while-open (BWO) 79
- bind-time security 137
- BMP program
  - convert a CICS shared database program 112

## C

- cache structures, defining 77
- CEDA DEFINE panel 121
- central processor complex (CPC) 9
- CFDT pools and coupling facility 88
- CFDT servers
  - coupling facility data tables 87
- CICS
  - availability 68
  - local DL/I 109
  - recovery consideration 106
  - shared database program 112
  - startup procedure 139, 149
  - system initialization parameters 149
  - transaction routing facility 49
  - transaction throughput 67
- CICS applications
  - migration to a sysplex 8
- CICS local DL/I
  - use in a sysplex 109
- CICS system data sets 158
- CICSplex
  - description 7
  - security 137
- CICSplex SM
  - description 57
  - manage inter-transaction affinities 102
  - relation to WLM 146
  - use in a sysplex 12

- CICSplex SM (*continued*)
  - workload balancing algorithms 143
  - workload management 143
  - workload separation 143
- clones
  - AOR 10
  - CICS regions 134
  - define common parameters 150
  - IMS subsystems 165
  - TORs 62
- cloning CICS regions 10, 68
- CMAS
  - target sysplex configuration 23
- Common Queue Server (CQS) 178
- connection definition
  - application-owning region 129
  - file-owning region 131
  - queue-owning region 131
  - terminal-owning region 125
- CONNECTION resource definition
  - SECURITYNAME parameter 137
- converting IMS batch jobs to BMP programs 172, 173
- coupling facility 4
  - and CFDT pools 88
  - and named counter pools 92
  - and TS pools 84
  - defining 77
  - structures used by DB2 116
- coupling facility data table
  - server 89
- coupling facility data table servers 87
- coupling facility data tables
  - defining 89
  - planning for a sysplex 87

## D

- data flow
  - MSC 171
- data set eligibility for RLS 74
- data tables, eligibility for RLS 74
- DATABASE 2 (DB2)
  - coupling facility maintenance 116
  - target sysplex configuration 23
- Database Recovery Control (DBRC)
  - use in a sysplex 112
- DB/DC environment
  - use in a sysplex 7
- DB2 (DATABASE 2)
  - data sharing 4
  - internal resource lock manager 116
  - naming conventions 115
  - planning for a sysplex 115
  - use in a sysplex 4
- deadlock detection, defining 78
- defining recovery attributes 79
- determining CICS response times 146
- dividing the IMS network 169, 172

- DRA startup parameter table 111
- DTRTRAN system initialization parameter 121, 123
- DYNAMIC parameter 121, 124
- dynamic transaction routing 49, 50
  - benefits 10
  - connection definitions 125
  - defining resources 121
  - improved availability 11
  - improved performance 10
  - planning 53
  - security authorization 139
  - session definitions 125
  - simplified systems management 11
  - transaction definitions
    - defining remote attributes 121
    - for dynamic routing 123
    - for static routing 121
  - use in a sysplex 10
- dynamic transaction routing program
  - CICSplex SM 57
  - manage inter-transaction affinities 102
  - planning 56
  - providing your own 145
  - workload management 143

## E

- EIBTRMID field 70
- EXEC CICS START command 69

## F

- file-owning region (FOR)
  - connection definitions 131
  - parameters for cloning regions 154
  - planning 95, 100
  - session definitions 131
  - target sysplex configuration 22
- follow-on phase
  - definition xii
- function shipping
  - description 95

## G

- generic resources
  - migration 59
  - planning 58
  - planning for using with IMS 178
  - requirements for using 179
  - restrictions on using with IMS 180
- generic resources function
  - description 13

## I

- ICF catalog 79
- IMS 6
  - application compatibility 7
  - automatic restart manager 166
  - availability 174

- IMS 6 (*continued*)
  - batch, converting to BMP programs 172, 173
  - cloning IMS subsystems 165, 166
  - converting batch jobs to BMP programs 172, 173
  - data compatibility 7
  - data sets
    - sharing between IMS subsystems 165
  - Database Recovery Control 112
  - DB2 considerations 23
  - DEDB considerations 23
  - dividing the network 169, 172
  - evolution 6
  - examples 165
    - defining an LU name 168
    - IMSID and system definition 166
    - LTERM naming convention 168
    - overriding system definition 165
  - IMSCTRL macro 166
  - IMSID
    - definition 166
    - ensuring unique 166, 167
    - example 166
  - internal resource lock manager 113
  - IRLM, restriction sharing with DB2 and IMS 28
  - logical terminal (LTERM)
    - APPC/IMS 168
    - definition 168
    - naming convention, example 168
  - LU names
    - defining, example 168
    - ensuring unique 168, 169
  - migrating applications 163, 174
  - migration to a Parallel Sysplex 165
  - moving 166
    - IMS subsystems to other MVS images 166
    - terminals, LU 6.2 applications, or users within a network 168
  - MSC 169
    - advantages 169
    - balancing workload 170
    - definition 169
    - Input Message Routine exit routine (DFSNPRT0) 170
    - minimizing network outages 170
    - planning for 169
    - Program Routing exit routine (DFSCMPRO) 170
    - transaction balancing 170
    - transaction flow 171
  - MVS automatic restart manager 166
  - MVS resource management 173
  - MVS workload manager 173
  - overriding system definition, example 165
  - partitioning applications 169
  - planning for a Parallel Parallel Sysplex environment 174
  - planning for a sysplex environment 163
  - planning for the network 169
  - planning for Version 6 175
  - recovery 174
  - reducing the batch window 173
  - restriction sharing IRLM with DB2 and IMS 28

- IMS 6 (*continued*)
  - sysplex data sharing 109
  - system definition stage 1 165
  - target Sysplex configuration 23
  - terminal names, ensuring unique 168, 169
  - use in a sysplex 6
  - user IDs, ensuring unique 168, 169
- IMS application migration 7
- IMS data sharing 4
- IMS Database Control (DBCTL)
  - creating for a sysplex 111
  - migrate from CICS local DL/I 109
  - naming conventions 111
  - planning 109
  - startup procedure 157
  - target sysplex configuration 22
  - use in a Sysplex 4
- IMSID 166
- INQUIRE command
  - transaction-system affinity 54
- integrity of data 76
- inter-transaction affinity
  - description 54
  - detection 54
  - manage with CICSplex SM 102
  - suspect programming techniques
    - transient data 104
  - temporary storage queue 101
  - transient data 101
  - workload balancing 145
- internal resource lock manager (IRLM)
  - used by DB2 116
  - used by IMS 113
- IRLM (Internal Resource Lock Manager)
  - migration 175

## L

- link userid
  - authorization 141
  - naming convention 141
- locks 76
- log manager
  - planning log streams 117
- log streams
  - planning 117
- logical terminal (LTERM), IMS 168
- LU name
  - workload separation 12

## M

- migrating
  - IRLM (Internal Resource Lock Manager) 175
- migrating IMS applications 163, 174
- MNSUBSYS parameter
  - startup procedure consideration 157
- MRO, multiregion operation
  - transaction routing 49
- MSC (IMS Multiple Systems Coupling) 169

- MSC (Multiple Systems Coupling)
  - data flow 171
- Multiple Systems Coupling (MSC) 169
- multiregion operation
  - function shipping 95
- multiregion operation (MRO)
  - bind-time security 137
  - connection definitions 125
  - link security 140
  - session definitions 125
  - transaction routing 49
  - use in a CICSplex 7
- MVS (Multiple Virtual Storage)
  - use of this term xii
- MVS workload manager (WLM)
  - planning for a CICSplex 143
  - relation to CICSplex SM 146
  - service class definition
    - CICS response times 146

## N

- named counter
  - server 93
- named counter pools and coupling facility 92
- named counters
  - defining 93
- named number counters
  - planning for a sysplex 91
  - servers 91
- naming conventions 15
  - applying to APPLID 34
  - applying to connections 36
  - applying to data sets 44, 158
  - applying to JOB names 43
  - applying to SESSIONS 39
  - applying to subsystem names 34
  - applying to SYSIDNT 36
  - applying to terminals 42
  - applying to VTAM APPL definitions 35, 36
  - applying to VTAM generic resource 34
  - description 31
  - for use with CICS 27
  - objectives 30
- NETNAME
  - workload separation 12
- Notices 183

## P

- Parallel Sysplex data sharing
  - migration 175
- Parallel Sysplex environment
  - migrating 175
- persistent LU-LU sessions
  - description 14
  - planning 62
- product availability
  - follow-on phase xii

## Q

- queue-owning region (QOR)
  - connection definitions 131
  - create 102
  - define as remote 102
  - inter-transaction affinity 101
  - naming conventions 102
  - parameters for cloning regions 155
  - planning 100
  - session definitions 131
  - target sysplex configuration 22
  - transient data 104

## R

- RACF
  - access list
    - APPL profile 138
  - bind-time security 137
  - use in a sysplex 15
- read integrity 76
- record-level sharing (RLS) 73
- recovery attributes, defining 79
- REMTENAME parameter 122, 124
- REMOTESYSTEM parameter 122, 124
- resource manager interface (RMI)
  - use in a sysplex 8
- resource manager regions 8
  - reasons for creating
    - availability 9
    - enhanced multiprocessor performance 9
    - faster restart 9
    - virtual storage constraint relief 9
    - workload separation 9
- restriction sharing IRLM with DB2 and IMS 28
- retained locks 76
- RLS, record-level sharing
  - planning for CICS 73
- RLS migration 73

## S

- SECURITYNAME parameter 137
- session definition
  - application-owning region 129
  - file-owning region 131
  - queue-owning region 131
  - terminal-owning region 125
- SET command
  - transaction-system affinity 54
- shared database program
  - convert to a BMP program 112
- shared message queues
  - benefits of using 176
  - planning for 176
  - required components for 177
- shared queues
  - defining 84
- SMS storage classes, defining 78
- SRVCLASS parameter of IEAICSxx 147
- START command
  - dynamic transaction routing 69

- START command (*continued*)
  - TERMIN keyword 70
- startup procedure
  - CICS regions 149
  - common parameters in SYSIN data set 156
  - DBCTL subsystem name 157
  - defining CICS region userids 139
  - sample job 158
  - unique parameters 157
- static transaction routing 49, 50
- SYSEVENT class 157
- SYSIDNT parameter
  - startup procedure consideration 157
- sysplex
  - benefits of running CICS 7
  - benefits of running IMS 6
- system initialization parameters 149
- system initialization table 149

## T

- temporary storage data sharing
  - planning for a sysplex 83
- temporary storage queue
  - definition in a sysplex 100
- temporary storage table (TST) 102
- TERMINID
  - workload separation 12
- terminal-owning region (TOR)
  - clone 62
  - connection definitions 125
  - omit transaction resource definitions 123
  - parameters for cloning regions 150
  - session definitions 125
  - split from AOR 50
  - target sysplex configuration 20
- transaction affinity
  - description 53
  - inter-transaction affinity 54
  - transaction-system affinity 54
- transaction affinity utility 54
- transaction identifier (TRANSID)
  - workload separation 12
- transaction routing 49
  - CICS 49
  - dynamic transaction routing 49
  - static transaction routing 49
- transaction routing facility 49
- transaction-system affinity 54
- transaction throughput
  - in a sysplex 67
- TRANSID 12
- transient data queue
  - definition in a sysplex 100, 104
- TS data sharing 83
  - CFDT server 89
  - defining coupling facility data tables 89
  - defining named counters 93
  - defining shared queues 84
  - named counter server 93
  - security 86, 90, 94
  - server 85

- TS data sharing 83 *(continued)*
  - subsystem interface 85, 89, 93
  - TS data sharing server 85
- TS pools and coupling facility 84
- TS server
  - non-recoverable TS data sharing 83

## U

- USERID 12
- userid identifier (USERID)
  - workload separation 12

## V

- VSAM
  - files owned by an FOR 95
- VSAM record-level sharing (RLS)
  - CFLEVEL required 74, 118
  - choosing between RLS and non-RLS mode 74
  - data set eligibility 74
  - planning for CICS 73
- VTAM
  - generic resources function 13, 58
  - persistent LU-LU sessions 14, 62
  - use in a sysplex 13

## W

- workload balancing
  - algorithms 143
  - shortest queue 144
- workload management
  - determining CICS response times 146
  - planning for a CICSplex 143
  - using SRVCLASS parameter, example of 147
- workload separation 143



---

# Readers' Comments — We'd Like to Hear from You

**z/OS**  
**Parallel Sysplex Application Migration**

**Publication No. SA22-7662-00**

**Overall, how satisfied are you with the information in this book?**

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

**How satisfied are you that the information in this book is:**

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

**Please tell us how we can improve this book:**

Thank you for your responses. May we contact you?  Yes  No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

\_\_\_\_\_  
Name

\_\_\_\_\_  
Address

\_\_\_\_\_  
Company or Organization

\_\_\_\_\_  
Phone No.



Fold and Tape

Please do not staple

Fold and Tape



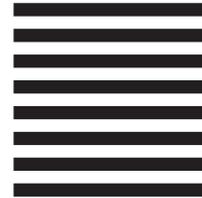
NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation  
Department 55JA, Mail Station P384  
2455 South Road  
Poughkeepsie, NY  
12601-5400



Fold and Tape

Please do not staple

Fold and Tape





File Number: S370/S390-34  
Program Number: 5694-A01



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SA22-7662-00

