

IMS/ESA



# Administration Guide: Transaction Manager

*Version 6*



IMS/ESA



# Administration Guide: Transaction Manager

*Version 6*

**Note**

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

**Fifth Edition (October 1999) (Softcopy Only)**

This edition replaces and makes obsolete the previous edition, SC26-8731-03. This edition is available in softcopy only. The technical changes for this edition are summarized under "Summary of Changes" on page xxi and are indicated by a vertical bar to the left of a change.

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

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 1974, 1999. All rights reserved.**

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

---

# Contents

<b>Notices</b> . . . . .	xvii
Programming Interface Information . . . . .	xvii
Trademarks . . . . .	xvii
Product Names . . . . .	xviii
<b>Preface</b> . . . . .	xix
Prerequisite Knowledge . . . . .	xix
Change Indicators . . . . .	xix
<b>Summary of Changes</b> . . . . .	xxi
Changes to the Current Edition of This Book for V6 . . . . .	xxi
Changes to This Book for Version 6 . . . . .	xxi
Library Changes for V6. . . . .	xxi

---

## Part 1. Administering an IMS Transaction Manager Network . . . . . 1

<b>Chapter 1. The IMS Transaction Manager Network</b> . . . . .	7
Introducing the IMS TM Network . . . . .	7
Programmable Logical Units . . . . .	9
Communications Controller and NCP . . . . .	10
VTAM . . . . .	10
IMS . . . . .	10
Message Processing Programs . . . . .	11
Introducing IMS Data Communication Services . . . . .	11
APPC/IMS and LU 6.2 Devices . . . . .	11
Fast Path . . . . .	11
Extended Terminal Option (ETO) . . . . .	12
Intersystem Communication (ISC) . . . . .	12
Message Format Service (MFS) . . . . .	13
Multiple Systems Coupling (MSC) . . . . .	13
Using DC Control . . . . .	13
Operating an IMS Network . . . . .	14
Operating the Network with APPC/IMS . . . . .	14
Initiating a Session with IMS . . . . .	14
Logging On and Signing Onto IMS . . . . .	14
Logging Off and Signing Off from IMS . . . . .	15
Operating in a Shared-Queues Environment . . . . .	15
Benefits of Using Shared Queues . . . . .	16
Required Components of a Shared-Queues Environment . . . . .	17
Overview of the Common Queue Server . . . . .	18
Using Generic Resource Groups to Balance Sessions . . . . .	19
Using the Fast Path Expedited Message Handler . . . . .	20
Concepts for Understanding the Fast Path EMH . . . . .	20
Criteria for Fast Path Application Programs Using EMH . . . . .	22
<b>Chapter 2. Planning the Network</b> . . . . .	23
Planning for Network Administration . . . . .	23
Documenting Network and Terminal Requirements . . . . .	24
The Terminal Profile . . . . .	24
Terminals Attached Using VTAM . . . . .	24
Understanding the IMS Terminal Network . . . . .	24
Terminal Connections to IMS . . . . .	25
XRF Terminal Support . . . . .	26

Logical Terminals . . . . .	26
APPC/IMS and LU 6.2 Terminal Support . . . . .	27
IMS Messages and Their Scheduling . . . . .	27
Message Flow within the IMS Online System . . . . .	30
Conversational Transactions . . . . .	31
Message Switches . . . . .	32
Logical Terminal Network Design . . . . .	32
Logical-Terminal Chains . . . . .	33
Logical-Terminal Queues . . . . .	33
Separating Input and Output Devices . . . . .	34
Logical and Physical Terminal Relationships . . . . .	35
Master Terminal . . . . .	35
Master Terminals in an XRF Complex . . . . .	35
NTO Terminals . . . . .	36
Terminal Modes and States . . . . .	36
Terminal and User Operating Modes . . . . .	37
Terminal and User States . . . . .	38
Planning for Security . . . . .	40
Transaction Authorization . . . . .	40
Command Authorization . . . . .	41
Transaction Command Security . . . . .	41
Password Security . . . . .	42
Security for APPC/IMS . . . . .	42
Security for ETO . . . . .	42
Planning for a Shared-Queues Environment . . . . .	43
Concepts for Operating in a Shared-Queues Environment . . . . .	43
Planning for Operating in a Shared-Queues Environment . . . . .	48
Planning for Generic Resource Groups . . . . .	56
Requirements for Using Generic Resource Groups . . . . .	57
Restrictions on Using Generic Resource Groups . . . . .	57
Managing Generic Resource Affinities Using IMS . . . . .	57
Managing Generic Resource Affinities Using VTAM . . . . .	58
Specifying Generic Resource Names . . . . .	58
Specifying APPC Generic Resource Names . . . . .	58
Initiating Sessions Using GRSNAMEs or APPLID names . . . . .	58
Changing the Logon Procedure . . . . .	58
Overriding the APPLID Field . . . . .	59
Determining When Affinities Are Terminated . . . . .	59
Terminating Affinities That Persist . . . . .	59
Dropping an IMS System from a Generic Resource Group . . . . .	60
Resetting Terminal Status . . . . .	60
Logging On after IMS Failure with Affinity . . . . .	60
Controlling Generic Resource Member Selection . . . . .	60
Ensuring Consistency across IMS Systems . . . . .	61
Bypassing Affinity Management During the IMS ESTAE Process . . . . .	62
Terminals Used with Fast Path . . . . .	62
Extended Recovery Facility (XRF) and IMS . . . . .	64
Terminals in an XRF Complex . . . . .	65
Class-1 Terminals . . . . .	65
Class-2 Terminals . . . . .	66
Class-3 Terminals . . . . .	66
Specifying Terminal Support . . . . .	67
<b>Chapter 3. Defining the Network . . . . .</b>	<b>69</b>
Preparing for the Operational Network . . . . .	69
Coordinating IMS Definition and Network Definition . . . . .	70

Using IMS as a Host Subsystem . . . . .	70
Defining VTAM Nodes . . . . .	71
Estimating VTAM Storage Requirements . . . . .	71
Determining VTAM Buffer Pool Values . . . . .	71
Determining the NCP Buffer Pool Values . . . . .	72
Determining Static and Dynamic Terminal Signon Requirements . . . . .	72
Checking Requirements for LOGON MODE Tables . . . . .	73
Specifying Initial VTAM Configurations . . . . .	74
Using SON/COS Support in IMS . . . . .	74
Network Operation . . . . .	75
Starting an IMS Network . . . . .	75
Session Initiation . . . . .	75
IMS Transaction Types and Transaction States . . . . .	76
Defining IMS Transaction Types . . . . .	76
Determining Transaction States . . . . .	77
Restarting the Network after Remote Takeover . . . . .	78
VTAM Definition for Transport Manager . . . . .	79
VTAM Buffering and Pacing . . . . .	79
VTAM Missing Interrupt Handler . . . . .	79
<b>Chapter 4. Using Shared Queues in the Network . . . . .</b>	<b>81</b>
Enabling Shared Queues . . . . .	81
Defining the CFRM Policy . . . . .	81
Defining an MVS Log Stream . . . . .	83
CQS Definitions . . . . .	83
IMS Definitions . . . . .	85
Using the Common Queue Server . . . . .	86
Starting CQS . . . . .	86
Restarting CQS. . . . .	86
Using IMS Commands to Access CQS . . . . .	87
Using CQS User-Supplied Exit Routines . . . . .	88
Changing the Size of Coupling Facility Structures . . . . .	89
Initiating System Checkpoint . . . . .	89
Initiating Structure Checkpoint . . . . .	89
Recovering Structures . . . . .	90
Deleting Message Queues . . . . .	90
CQS Logging and the MVS Logger . . . . .	91
Shutting Down CQS . . . . .	91
Tuning for Performance in a Shared-Queues Environment . . . . .	91
Tuning Structures and Parameters. . . . .	91
Recommendation for Fast Path Transactions . . . . .	92
<b>Chapter 5. Editing and Formatting Messages. . . . .</b>	<b>93</b>
Message Format Service . . . . .	93
MFS Components. . . . .	94
Administering MFS . . . . .	96
Advantages to Using MFS. . . . .	97
MFS Control Blocks . . . . .	98
Overview of MFS Components and Operation . . . . .	99
Creating MFS Formats with SDF II . . . . .	99
Basic Edit . . . . .	101
IMS Editing for Intersystem Communication (ISC) . . . . .	102
Transparency Option . . . . .	103
Unprotected Screen Option . . . . .	104
Bypassing MFS Editing . . . . .	104
When To Bypass MFS Editing . . . . .	104

Locking and Unlocking the Terminal Keyboard . . . . .	104
IMS Sensitivity to Nongraphic Message Data . . . . .	104
Output Message Segment Editing . . . . .	104
Editing of Input Message Segments by MFS . . . . .	105
Editing of Input Message Segments by Basic Edit . . . . .	105
Controlling Output Devices . . . . .	106
Small Buffer Devices . . . . .	106
Controlling Output . . . . .	107
Using a Printer Component . . . . .	107
Spooled Output Control . . . . .	108
Using Printer Components of the IBM 3270 Information Display System . . . . .	108
Specifying Candidate Printers . . . . .	108
Operational Considerations . . . . .	109
Sharing Printers between Systems . . . . .	110

---

**Part 2. Administering the Extended Terminal Option . . . . . 111**

<b>Chapter 6. Overview of the Extended Terminal Option . . . . .</b>	<b>115</b>
Benefits of Using ETO . . . . .	115
ETO Terminology . . . . .	116
Terminals . . . . .	116
Dynamic Users . . . . .	116
Structures . . . . .	116
Descriptors . . . . .	117
ETO Concepts . . . . .	119
When Structures Are Created and Deleted . . . . .	119
Descriptors and Exit Routines . . . . .	119
How Descriptors Are Created and Used . . . . .	120
How Control Blocks Are Created and Deleted . . . . .	120
Summary of ETO Implementation . . . . .	121
<b>Chapter 7. Administering the Extended Terminal Option . . . . .</b>	<b>123</b>
Planning for ETO . . . . .	124
Identifying Your Requirements . . . . .	124
Restrictions on ETO . . . . .	125
Defining Physical Terminals . . . . .	125
Identifying VTAM Device Types, Screen Sizes, and Models . . . . .	128
Planning a High-Security Environment with ETO . . . . .	134
Planning for MFS . . . . .	135
Planning User IDs . . . . .	135
Planning User Queue Names . . . . .	135
Planning Operations . . . . .	136
Planning for MSC Support with ETO . . . . .	136
Coding ETO Descriptors . . . . .	136
Creating Descriptors Using the System Definition Process . . . . .	137
Storing Descriptors . . . . .	137
Creating Logon Descriptors . . . . .	138
Creating User Descriptors . . . . .	142
Creating MFS Device Descriptors . . . . .	145
Creating MSC Descriptors . . . . .	148
Using Exit Routines . . . . .	148
Starting ETO . . . . .	149
Logging onto ETO Terminals . . . . .	150
Limiting Dynamic Logon To Specific Terminal Types . . . . .	150
Creating and Reusing LTERM Control Blocks . . . . .	150
Using Default CINIT or BIND User Data Formats . . . . .	150



Signing On and Queue LTERM Allocation . . . . .	151
Providing Signon Data. . . . .	152
Providing Signon Data for ISC, SLU-P, Finance, and Output-Only Devices	152
Signing On Multiple Times . . . . .	152
Receiving the Signon Required Message—DFS3649A . . . . .	153
Receiving the Session Status Message—DFS3650 . . . . .	154
ETO Terminal-LTERM Relationship . . . . .	154
How IMS Determines Which Queues to Allocate . . . . .	155
Setting Special Processing Modes . . . . .	155
Using ETO with Printers . . . . .	156
Direct Printing . . . . .	156
Associated Printing . . . . .	156
Defining Your Printers . . . . .	157
Sharing Printers Using ETO. . . . .	158
Using Operator Commands . . . . .	158
Using the /OPNDST Command . . . . .	159
Using the /ASSIGN Command. . . . .	159
Using System Definition Parameters . . . . .	159
Using the DLQT Parameter To Set DEADQ Status Time . . . . .	159
Using Autosignoff (ASOT) . . . . .	160
Using Autologoff (ALOT) . . . . .	161
Using the Autosignoff and Autologoff Timer . . . . .	163
Using Autologon . . . . .	163
Assigning Output. . . . .	164
Asynchronous Output . . . . .	164
Delivering Output Messages to Non-Originating Terminals . . . . .	165
Inadvertent Output Data Streams. . . . .	165
Signing Off . . . . .	165
Logging Off. . . . .	166
Improving Performance by Deleting ETO Control Blocks . . . . .	166
Using the IDC0 Trace Facility . . . . .	166
ETO and LU 6.1 (ISC) Terminals . . . . .	167
ETO and STSN Terminals . . . . .	167
SNA STSN Terminal Considerations. . . . .	167
ETO and 3600/Finance and SLU P . . . . .	168
/SIGN Support for ETO STSN Devices: ISC, Finance, and SLU P . . . . .	168
ETO with Conversation Mode and Response Mode . . . . .	169
Conversation and ETO . . . . .	169
Response Mode . . . . .	170

---

**Part 3. Administering Multiple Systems Coupling . . . . . 171**

<b>Chapter 8. Overview of Multiple Systems Coupling . . . . .</b>	<b>173</b>
Multiple Systems Coupling Concepts and Terminology . . . . .	173
Physical Links. . . . .	173
Logical Links . . . . .	175
Data Partitioning with Multiple Systems Coupling . . . . .	176
Remote and Local Systems. . . . .	177
Flow of Data within Multiple Systems . . . . .	179
Message Routing . . . . .	180
Routing Path . . . . .	181
Logical Destinations . . . . .	181
Input, Destination, and Intermediate Systems . . . . .	182
System Identifiers (SYSIDs). . . . .	184
Using the Destination Name and SYSIDs To Route Messages . . . . .	185
Remote LTERMs. . . . .	187

Routing Exit Routines . . . . .	188
MSC Directed Routing. . . . .	190
Remote Destination Verification . . . . .	191
Logical Link Path . . . . .	191
<b>Chapter 9. Administering Multiple Systems Coupling . . . . .</b>	<b>193</b>
Design Considerations for Multiple Systems . . . . .	193
Minimizing Resource Consumption . . . . .	193
Balancing Resource Demand . . . . .	194
Planning for Conversational Processing . . . . .	194
Routing Exit Routines with Conversations . . . . .	196
Remote Destination Verification for Conversations . . . . .	196
Saving Truncated Data in the SPA . . . . .	196
Conversation Termination . . . . .	197
Abnormal Conversation Termination. . . . .	197
System Definition for Multiple Systems Coupling . . . . .	197
Local System Definitions . . . . .	198
Defining Partner Systems . . . . .	198
Defining the Physical Link . . . . .	199
Defining the Logical Link . . . . .	199
Defining a Logical Path . . . . .	200
Setting Link Priorities for Remote Transactions. . . . .	200
Specifying Routing Exit Routines . . . . .	201
How Network Definition Is Affected by Multiple Systems . . . . .	203
Verifying Transaction Definitions Across Systems . . . . .	203
Using the Multiple Systems Verification Utility . . . . .	203
Verifying the System Definition Status Online . . . . .	204
Security Considerations for Multiple Systems . . . . .	205
Establishing Operating Procedures for Multiple Systems . . . . .	206
Initializing Multisystem Communication. . . . .	206
Terminating Multisystem Communication . . . . .	206
IMS Commands That Affect MSC Operation. . . . .	207
Logical Link Assignments . . . . .	207
Restarting a Logical Link . . . . .	208
Commands That Help Control Resources. . . . .	208
Using the /DISPLAY Command . . . . .	209
Logical Link Path Control. . . . .	210
Recovery Considerations. . . . .	210
Message Recovery . . . . .	211
Stopped Transactions . . . . .	211
Application Program Abnormal Termination . . . . .	211
Dequeuing Messages, Responses, and Transactions . . . . .	211
Monitoring and Tuning Multiple Systems . . . . .	212
Coordinating Performance Information . . . . .	212
Interpreting IMS Monitor MSC Reports. . . . .	213
Determining Cross-System Queuing . . . . .	213
Assessing the Effect of Link Loading . . . . .	214
Assessing Link Queuing Times . . . . .	215
Extracting Multiple-System Transaction Statistics . . . . .	215
Controlling the Log Merge . . . . .	215
Interpreting the Transaction Analysis Report. . . . .	216

---

**Part 4. Administering Intersystem Communication . . . . . 217**

<b>Chapter 10. Overview of Intersystem Communication . . . . .</b>	<b>221</b>
What Is ISC? . . . . .	221

Comparing ISC to MSC . . . . .	222
IMS Facilities Available to ISC . . . . .	223
Distributed Transaction Processing . . . . .	223
Distributed Services . . . . .	224
IMS Transaction Types . . . . .	224
IMS Execution Modes . . . . .	224
IMS Editing Facilities . . . . .	225
ISC Message Integrity . . . . .	225
ISC Security . . . . .	225
ISC in an XRF Complex . . . . .	225
Sample System Configurations . . . . .	226
ISC between IMS and CICS . . . . .	228
Terminal Device-Dependent Data . . . . .	229
Passing CICS Data to IMS . . . . .	229
<b>Chapter 11. VTAM Facilities Used for ISC . . . . .</b>	<b>231</b>
VTAM Commands and Indicators . . . . .	232
Using the VTAM Application Program Interface . . . . .	233
Specifying Logon Modes When Establishing Connection . . . . .	233
Design Considerations for Secondary Logical Units . . . . .	234
<b>Chapter 12. IMS Facilities Affected by ISC . . . . .</b>	<b>235</b>
Editing Messages . . . . .	235
Issuing IMS Commands from an ISC Session . . . . .	236
Effects on Parallel Sessions . . . . .	236
Using IMS Test Mode for ISC Sessions . . . . .	237
IMS Control Block Storage on ISC Parallel Sessions . . . . .	237
Relationship of ISC and IMS Execution Modes . . . . .	238
External Specification of Execution Modes . . . . .	238
Internal Definition of Execution Mode . . . . .	238
Resultant Processing Mode . . . . .	239
LTERM Users (Subpools) and Components . . . . .	240
<b>Chapter 13. Designing Communications Using the ISC Protocol . . . . .</b>	<b>243</b>
Determining Output Protocols . . . . .	243
Accessing Existing Application Programs with ISC . . . . .	244
Accessing Programs That Use MFS . . . . .	245
Accessing Programs That Do Not Use MFS . . . . .	245
Routing Messages . . . . .	246
Considerations for IMS-to-IMS ISC Sessions . . . . .	255
Statically Defining an ISC Node to IMS . . . . .	257
Choosing Parameters: System Design Considerations . . . . .	260
COMM Macro Statement . . . . .	260
NAME Macro . . . . .	261
SUBPOOL Macro . . . . .	261
TERMINAL Macro . . . . .	262
System Definition Summary . . . . .	263
<b>Chapter 14. ISC Protocol Guide and Reference . . . . .</b>	<b>265</b>
Operating the Network . . . . .	266
Making IMS Ready . . . . .	266
Bringing Up an IMS Network . . . . .	266
Shutting Down an IMS Network . . . . .	266
Controlling the Session (Session Control Protocols) . . . . .	267
Initiating a Session . . . . .	267
Establishing Connection with the XRF Complex . . . . .	268

Binding the Session . . . . .	268
Resynchronizing Sessions . . . . .	270
Designing Restart Resynchronization Procedures . . . . .	271
Determining Session Synchronism Using STSN . . . . .	276
Performing the Resynchronization . . . . .	277
Completing Session Initiation . . . . .	278
Running the Session . . . . .	279
Terminating the Session . . . . .	279
Normal Termination . . . . .	279
Abnormal Termination . . . . .	280
Using STSN to Resynchronize Sessions . . . . .	280
Primary-to-Secondary Flow Matrix . . . . .	280
Secondary-to-Primary Flow Matrix . . . . .	282
STSN Command Format . . . . .	283
Controlling Data Flow (DFC Protocols) . . . . .	285
Handling IMS Response Mode or Conversational Output Errors . . . . .	285
Normal Conversation Termination Extension with ISC . . . . .	287
Keeping Half Sessions Synchronized . . . . .	287
Sync Points Requested on Input to IMS . . . . .	288
Sync Points Requested on Output by IMS . . . . .	289
Sync Point and Response Requirements . . . . .	289
Sync-Point Indicators on Messages . . . . .	292
Data Flow Control Protocol Reference . . . . .	295
BID Protocol . . . . .	295
Bracket and Half-Duplex Protocol . . . . .	296
CANCEL Protocol . . . . .	304
Chaining Protocol . . . . .	304
CHASE Protocol . . . . .	305
ERP Purging . . . . .	305
LUSTATUS Protocol . . . . .	308
Paged Messages ERP . . . . .	311
Ready-to-Receive Protocol . . . . .	312
RSHUT Protocol . . . . .	312
Selective Receiver ERP . . . . .	312
Sender ERP . . . . .	316
Sense Codes that IMS Receives . . . . .	319
Sense Codes that IMS Sends . . . . .	319
SIGNAL Protocol . . . . .	320
Symmetrical Session Shutdown for LU 6.1 (SBI and BIS) . . . . .	320
Function Management Headers . . . . .	321
Using FM Headers to Invoke ISC Edit . . . . .	323
Initiating a Process: ATTACH FM Header . . . . .	323
Error Recovery Procedure FM Header . . . . .	324
Resetting the Active Process: RAP FM Header . . . . .	324
Requesting Asynchronous Transaction Processing: SCHEDULER FM Header . . . . .	325
System Message Process (SYSMSG) and Related FM Headers . . . . .	326
<b>Chapter 15. Using MFS with ISC . . . . .</b>	<b>329</b>
Activating MFS Input Formatting . . . . .	329
Activating MFS Output Formatting . . . . .	330
MFS Distributed Presentation Management (DPM) Messages . . . . .	330
MFS Page Delete Function . . . . .	331
MFS Online Error Detection . . . . .	331
Output Errors . . . . .	331
Input Errors . . . . .	331

Paging Errors . . . . .	332
The ATTACH and SCHEDULER FM Headers under MFS. . . . .	333
Data Descriptor FM Headers . . . . .	334
Input Data Descriptor FM Header . . . . .	334
Output Data Descriptor FM Header . . . . .	334
Controlling Demand-Paged Messages: QMODEL FM Headers . . . . .	335
Request (Input) QMODEL FM Headers . . . . .	336
QGETN FM Header . . . . .	336
QGET FM Header . . . . .	336
QPURGE FM Header . . . . .	337
Reply (Output) QMODEL FM Headers . . . . .	337
QXFR FM Header . . . . .	337
QSTATUS FM Header. . . . .	337
The RAP FM Header under MFS. . . . .	338
<b>Chapter 16. FM Header Format Reference . . . . .</b>	<b>339</b>
ATTACH FM Header Format . . . . .	339
ATTIU. . . . .	341
ATTDSP . . . . .	341
ATTDBA . . . . .	342
ATTDPN. . . . .	343
ATTPRN. . . . .	345
ATTRDPN and ATTRPRN . . . . .	346
ATTDQN and ATTDQP . . . . .	347
ATTACC . . . . .	347
Data Descriptor FM Header. . . . .	347
Input Data Descriptor FM Header Format. . . . .	347
Output Data Descriptor FM Header Format . . . . .	348
Error Recovery Procedure FM Header. . . . .	349
QMODEL FM Headers . . . . .	349
QGET FM Header Format . . . . .	349
QGETN FM Header Format. . . . .	350
QPURGE FM Header Format . . . . .	351
QSTATUS FM Header Format . . . . .	352
QXFR FM Header Format . . . . .	353
Reset Attached Process (RAP) FM Header Format . . . . .	355
SCHEDULER FM Header Format . . . . .	355
SYMSMSG Process Headers. . . . .	356
SYSERROR FM Header Format . . . . .	356
SYSSTAT FM Header . . . . .	356

---

**Part 5. Administering IMS in a Client/Server Environment . . . . . 357**

<b>Chapter 17. CPI Communications . . . . .</b>	<b>359</b>
CPI-C Driven Application Programs . . . . .	359
SAA Resource Recovery Commit Processing . . . . .	359
Normal Termination. . . . .	360
Backout Processing. . . . .	360
Abnormal Termination . . . . .	360
Session Failure . . . . .	361
Return Codes . . . . .	361
System Restart/Resolve-in-Doubt Processing . . . . .	361
CPI-C Application Program Recovery . . . . .	362
Programming Requirements . . . . .	362
Pseudonym Files . . . . .	362
Distributing SYNC-Point/Protected Conversations. . . . .	362

Introducing RRS/MVS . . . . .	363
Activating Protected Conversations . . . . .	365
<b>Chapter 18. Administering APPC/IMS and LU 6.2 Devices . . . . .</b>	<b>367</b>
Introducing APPC/IMS . . . . .	367
APPC/IMS Application Program Interface . . . . .	368
Implicit API . . . . .	368
Explicit API . . . . .	369
APPC/IMS Application Programs . . . . .	369
Standard IMS Application Programs . . . . .	369
MSC and Standard IMS Application Programs . . . . .	370
Modified IMS Application Programs . . . . .	371
MSC and Modified IMS Application Programs . . . . .	371
CPI Communications Driven Application Programs . . . . .	372
Using the MOD Name and LTERM Interface . . . . .	373
Establishing APPC/IMS . . . . .	374
TP_Profile . . . . .	374
MVS System Data File Manager Utility (ATBSDFMU) Example . . . . .	376
Side Information — Outbound . . . . .	377
PARMLIB Member . . . . .	378
APPC/MVS Timeout Service . . . . .	379
APPC/MVS Error Extract Service . . . . .	380
Initializing LU 6.2 Descriptors . . . . .	380
Using MSC in an APPC/IMS Environment . . . . .	381
Recovering APPC Transactions in an MSC Environment . . . . .	382
Recoverable versus Irrecoverable Transactions . . . . .	382
Local APPC Transaction Discardability versus Nondiscardability . . . . .	382
OTMA Transaction Discardability versus Nondiscardability . . . . .	383
Transaction Processing Point of Failure . . . . .	383
Recoverability Flows of LU 6.2 Transactions . . . . .	386
Planning for XRF . . . . .	388
Transaction Retry Characteristics . . . . .	389
Qualifying Network LU Names . . . . .	389
DFSAPPC System Service . . . . .	390
Message Switching . . . . .	390
Asynchronous Output Delivery . . . . .	393
APPC Transaction Security . . . . .	394

---

**Part 6. Administering SLU P and Finance Communication. . . . . 397**

<b>Chapter 19. Overview of SLU P and Finance Communication . . . . .</b>	<b>399</b>
The IMS-SLU P Network . . . . .	400
System Configuration . . . . .	400
SLU P and Finance Workstations . . . . .	401
System Controller Application Program . . . . .	402
Writing the Controller Application Program with MFS and XRF . . . . .	402
Writing the Controller Application Program for MFS . . . . .	402
Writing the Controller Application Program for XRF Systems . . . . .	403
Converting Controller Application Programs from Finance to SLU P . . . . .	404
VTAM Facilities Used . . . . .	404
VTAM Commands and Indicators . . . . .	405
Request-Recovery Command . . . . .	406
Change-Direction Indicator . . . . .	406
Establishing Connection and Specifying Logon Modes . . . . .	407
Establishing Connection with the XRF Complex . . . . .	408
Bracket and Send/Receive Management . . . . .	408

<b>Chapter 20. IMS Facilities Used for SLU P and Finance</b>	411
Component Definition	411
LTERM Naming	411
Output Component Selection	412
Input Component Determination	412
Terminal-Response Mode	413
Defining a Workstation for Terminal-Response Mode	415
Output Messages Sent While in a Between-Brackets State	416
Designing for Output Messages Sent While in Between-Brackets State	418
IMS Message Format Service	418
Designing MFS for the Workstation Environment	418
MID/MOD Chaining	419
MFS Output Formatting for the SLU P System	419
MFS Message Recovery	419
MFS Control Functions (Finance)	420
MFS Control Functions (SLU P)	420
MFS Paging and BID Options	421
Display Screen Protection for Finance Stations	421
Extended Output Component Protection (SLU P)	422
Input and Output Editing Options (SLU P)	423
Use of Responses or Brackets to Acknowledge Recoverable Input	424
Message Recovery	425
Message Resynchronization	426
Finance and SLU P in an XRF Complex	427
Fast Path Messages with Finance and SLU P	427
Fast Path Output Messages (Finance)	427
Fast Path Output Messages (SLU P)	428
Fast Path Message Resynchronization	429
<b>Chapter 21. Network Operation for SLU P and Finance</b>	431
Starting an IMS Network	431
Making IMS Ready	431
Session Initiation (Starting Workstations)	431
Session-Initiation Transmission Sequence	432
Controller Application Program Involvement in Message Resynchronization	433
Design Considerations	433
Sequence Number Management	434
Set-and-Test-Sequence-Numbers (STSN)	435
Suspending Output from IMS	438
Session Termination	439
Orderly Termination	440
Immediate Termination	440
Shutting Down an IMS Network	441
SLU P Messages	441
Send/Receive and Bracket Protocol	441
<b>Chapter 22. SLU P Message Protocols</b>	443
General Format of Input Function Management Headers (Finance)	443
Input Message Descriptor Byte (Finance)	444
General Format of Input Function Management Headers (SLU P)	444
Input Message Descriptor Bytes (SLU P)	445
Input Component Identification (SLU P)	446
Input Bracketing Protocol	446
Activating MFS Input Formatting for Finance Workstations	446
Output Messages	447
MFS Distributed Presentation Management Output (SLU P)	449

General Format of Output Function Management Headers (Finance)	450
Output Message Descriptor Byte (Finance)	450
Output Component ID Byte (Finance)	451
MFS Data Bytes (Finance)	451
General Format of Output Function Management Headers (SLU P)	451
Output Message Descriptor Bytes (SLU P)	452
MFS Data Bytes (SLU P)	452
Output Bracketing Protocol	453
Activating MFS Output Formatting	454
Response Requests (Finance)	454
Response Requests (SLU P)	455
Input Response Requirements	455
Output Response Requirements	456
IMS Transaction Types	456
Recoverable-Inquiry Transactions	457
Irrecoverable-Inquiry Transactions	457
Verifying IMS Receipt of Irrecoverable Messages	458
IMS Message Switches	458
IMS Commands	458
VTAM Commands and Indicators	458
MFS Control Requests	459
Error Handling	459
IMS-Detected Errors	459
Controller or Station-Detected Errors	460
VTAM Logical Unit Status (LUSTATUS) Command	461
VTAM Ready-to-Receive (RTR) Command	462
VTAM CANCEL Command	462
VTAM Request-Recovery Command	463

---

**Part 7. Appendixes** . . . . . 465

<b>Appendix A. Using Programmed Symbols for IBM 3270</b>	467
Determining Whether the Buffers Are Loaded	467
Loading the PS Buffer	467
Solving Load Problems	468
Reloading the PS Buffer	469
Application Design Considerations	469
<b>Appendix B. Bind Parameters for SLU P and LU 6.1</b>	471
Finance Communication System Bind Parameters	471
LU Type 6.1 Bind Parameters	473
IMS as Primary Half Session	473
IMS as Secondary Half Session	476
<b>Appendix C. Bind Parameters for SLU 1 and SLU 2</b>	481
SLU 1 Bind Parameters	481
SLU 2 Bind Parameters	483
<b>Appendix D. Format for CINIT User Data Parameters</b>	485
<b>Appendix E. SNA Character String Controls</b>	487
Format Controls	487
Control Function Code Assignments	488
<b>Appendix F. Examples Using ISC Edit ATTACH Parameters</b>	489
ATTACH and SCHEDULER Parameters with ISC Edit	489



ATTACH Parameters with the IMS SYSMMSG Process . . . . .	490
ATTACH and SCHEDULER Parameters with IMS MFS . . . . .	492
<b>Appendix G. How IMS and CICS Use the ISC Interface . . . . .</b>	<b>497</b>
Functions Available to the ISC Session . . . . .	497
General Flow of CICS EXEC Commands within a CICS Application . . . . .	500
CICS to IMS Using SEND/RECEIVE EXEC Commands . . . . .	501
CICS to IMS Using the SEND LAST EXEC Command . . . . .	502
IMS to CICS Using the RECEIVE EXEC Command . . . . .	503
Coding Asynchronous Messages . . . . .	504
CICS to IMS Using the START/RETRIEVE EXEC Commands . . . . .	505
IMS to CICS Using the RETRIEVE EXEC Command . . . . .	506
Commands That Should Not Be Used on an ISC Session . . . . .	507
Selecting Appropriate CICS Installation Options for ISC . . . . .	507
Coding CICS System Generation Options . . . . .	508
Preparing CICS Resource Definition . . . . .	508
Defining IMS-CICS LU 6.1 Links . . . . .	508
Defining Compatible IMS and CICS Nodes . . . . .	509
System Names . . . . .	509
Number of Sessions . . . . .	510
Session Names . . . . .	510
Other Session Parameters . . . . .	510
Defining Multiple Links to an IMS System. . . . .	513
Defining CICS Transactions for IMS-CICS ISC . . . . .	515
Defining CICS Backout In-Doubt Processing . . . . .	515
Defining CICS Transactions for Asynchronous Communication to IMS . . . . .	515
Initiating and Allocating a Session from CICS . . . . .	515
Other Ways of Initiating a Session . . . . .	516
Terminating a Session from CICS . . . . .	516
Designing CICS Applications for ISC . . . . .	517
Understanding Application-Related Concepts . . . . .	517
Subsystem Design—Direct-Control versus Queued . . . . .	518
Synchronous and Asynchronous Processing . . . . .	518
Principal and Alternate Facility . . . . .	520
CICS versus IMS Conversation Mode . . . . .	520
Sending IMS Commands from CICS . . . . .	521
Sync Points . . . . .	522
Coding Function Management Headers for CICS . . . . .	525
ATTACH Function Management Header . . . . .	525
SCHEDULER Function Management Header . . . . .	528
Queue Model Function Management Headers . . . . .	530
Data Descriptor Function Management Header . . . . .	530
System Message Process (SYSMMSG) Function Management Header . . . . .	530
Error Recovery Procedure Function Management Header. . . . .	531
Recovery and Restart Concepts . . . . .	531
Logical Unit of Work . . . . .	531
Recovering Outstanding Message Traffic after a Failure . . . . .	532
Handling Transaction Abends . . . . .	536
IMS Transaction Abend . . . . .	536
CICS Transaction Abend . . . . .	536
Coding CICS Applications for Restart . . . . .	537
<b>Appendix H. ISC Data Flow Control Examples . . . . .</b>	<b>539</b>
Non-MFS Bracket and Half-Duplex Protocol Examples . . . . .	539
MFS Bracket and Half-Duplex Protocol Examples . . . . .	540
MFS Output Examples . . . . .	540

MFS Input Examples . . . . .	543
SBI/BIS Examples . . . . .	545
Signal Protocol Example . . . . .	546
<b>Appendix I. ISC Error Recovery Procedure Examples . . . . .</b>	<b>547</b>
Sender-Detected Error Examples. . . . .	547
Receiver-Detected Error Examples . . . . .	548
<b>Appendix J. Sample Program for IMS-CICS ISC . . . . .</b>	<b>551</b>
Installation Procedure . . . . .	551
IMS Sample Program (DFSISC00) . . . . .	551
Job Control Statements for the Sample Program . . . . .	554
IMS System Definition Statements . . . . .	554
MFS Formats . . . . .	555
Program Specification Block (PSB) Generation for the Sample Program . . . . .	556
Application Control Block (ACB) Generation. . . . .	557
<b>Bibliography . . . . .</b>	<b>559</b>
IMS/ESA Version 6 Library . . . . .	559
<b>Index . . . . .</b>	<b>561</b>
<b>Readers' Comments — We'd Like to Hear from You . . . . .</b>	<b>585</b>

---

## Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter 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  
500 Columbus Avenue  
Thornwood, NY 10594  
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling (1) the exchange of information between independently created programs and other programs (including this one) and (2) the mutual use of the information that has been exchanged, should contact:

IBM Corporation  
555 Bailey Avenue, W92/H3  
P.O. Box 49023  
San Jose, CA 95161-9023

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

---

## Programming Interface Information

This book is intended to help system programmers administer an IMS Transaction Manager network. This book documents General-use Programming Interface and Associated Guidance Information provided by IMS.

General-use programming interfaces allow the customer to write programs that obtain the services of IMS.

---

## Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AS/400	MVS
BookManager	MVS/ESA
CICS	NetView
CICS/ESA	OS/390
CICS/MVS	PS/2

DB2  
DFSMS  
ES/9000  
IBM  
IMS  
IMS/ESA

RACF  
SAA  
Series/1  
VM/ESA  
VTAM

Other company, product, and service names, which may be denoted by a double asterisk (\*\*), may be trademarks or service marks of others.

---

## Product Names

In this book, the licensed program “DB2 for MVS/ESA” is referred to as “DB2”.

---

## Preface

This book is for system programmers who administer an IMS/ESA<sup>1</sup> Transaction Manager network for their installations. They can use this book to design a network that runs with the Virtual Telecommunications Access Method (VTAM), the Message Format Service (MFS), Multiple Systems Coupling (MSC), Intersystem Communication (ISC), the Extended Terminal Option (ETO), Remote Site Recovery (RSR), and APPC/IMS.

**Related Reading:** For the definitions of terms used in this book, see the *IMS/ESA Master Index and Glossary*.

---

## Prerequisite Knowledge

You should first read *IMS/ESA Administration Guide: System*. Its introductory chapter, which covers planning activities, the IMS environments, and administration concepts, is particularly useful as a background for this book.

You should be familiar with the Systems Network Architecture (SNA), VTAM concepts, and facilities that govern communication between a VTAM application program and a SNA logical unit. Refer to the following publications:

*Systems Network Architecture Technical Overview*

*Network Program Products General Information*

*Systems Network Architecture Format and Protocol Reference: Architectural Logic*

If you use APPC/IMS, you must be familiar with APPC/MVS in order to correctly define APPC/MVS configurations. See *CPI Communications Reference*.

If you implement ISC sessions between IMS and CICS/MVS or CICS/ESA, you should understand CICS. *CICS/ESA 4.1 Intercommunication Guide* is particularly useful.<sup>2</sup>

These publications are also prerequisites:

*Advanced Function for Communication System Summary*, GA27-3099

*Network Program Products General Information*, GC30-3350

*System/390 MVS Sysplex Overview: An Introduction To Data Sharing*, GC28-1208

---

## Change Indicators

Technical changes are indicated in this publication by a vertical bar (|) to the left of the changed text.

---

1. "IMS/ESA" is hereafter referred to as "IMS."

2. "CICS/MVS" and "CICS/ESA" are hereafter referred to as "CICS."



---

## Summary of Changes

---

### Changes to the Current Edition of This Book for V6

This edition, which is in softcopy only, includes technical and editorial changes.

---

### Changes to This Book for Version 6

The major additions to this book are as follows:

Chapter 1	Overview of IMS shared queues, CQS, and generic resources
Chapter 2	Planning information for IMS shared queues, CQS, and generic resources
Chapter 4	Using shared queues and CQS
Chapter 17	Introduction to RRS/MVS Activating protected conversations

Other major changes to this book are:

- *Appendix B: ETO Terminology and Usage* has been incorporated into the following two chapters:
  - Chapter 6. Overview of ETO
  - Chapter 7. Administering ETO
- *Appendix L: Programming the IRSS Interface* has been removed from this manual, but remains available in softcopy format.
- Organizational, technical, and editorial corrections have been made throughout.

---

### Library Changes for V6

The IMS/ESA Version 6 library differs from the IMS/ESA Version 5 library in these major respects:

- *IMS/ESA Common Queue Server Guide and Reference*  
This new book describes the IMS Common Queue Server (CQS).
- *IMS/ESA DBRC Guide and Reference*  
This new book describes all the functions of IMS Database Recovery Control (DBRC).
- The IMS Application Programming summary books (*IMS/ESA Application Programming: Database Manager Summary*, *IMS/ESA Application Programming: Transaction Manager Summary*, and *IMS/ESA Application Programming: EXEC DLI Commands for CICS and IMS Summary*) are no longer included with the IMS library.
- The Softcopy Master Index is not included.
- All information about IRLM 1.5 and data sharing using IRLM 1.5 has been removed from the IMS V6 books. If you use IRLM 1.5, and want to migrate to using IRLM 2.1 and Sysplex data sharing, see *IMS/ESA Release Planning Guide*.
- The chapter that was titled "Database Control (DBCTL) Interface" in the *IMS/ESA Customization Guide* has been revised for Open Database Access (ODBA) and moved to "Appendix A, Using the Database Resource Adapter (DRA)" in the *IMS/ESA Application Programming: Database Manager*.





---

# Part 1. Administering an IMS Transaction Manager Network

<b>Chapter 1. The IMS Transaction Manager Network</b>	7
Introducing the IMS TM Network	7
Programmable Logical Units	9
Communications Controller and NCP	10
VTAM	10
IMS	10
Message Processing Programs	11
Introducing IMS Data Communication Services	11
APPC/IMS and LU 6.2 Devices	11
Fast Path	11
Extended Terminal Option (ETO)	12
Intersystem Communication (ISC)	12
Message Format Service (MFS)	13
Multiple Systems Coupling (MSC)	13
Using DC Control	13
Operating an IMS Network	14
Operating the Network with APPC/IMS	14
Initiating a Session with IMS	14
Logging On and Signing Onto IMS	14
Logging Off and Signing Off from IMS	15
Operating in a Shared-Queues Environment	15
Benefits of Using Shared Queues	16
Required Components of a Shared-Queues Environment	17
Overview of the Common Queue Server	18
Using Generic Resource Groups to Balance Sessions	19
Using the Fast Path Expedited Message Handler	20
Concepts for Understanding the Fast Path EMH	20
Fast Path Message Scheduling	20
Routing Codes and Balancing Groups	21
Fast Path Input Edit/Routing Exit Routine (DBFHAGU0)	21
Message Buffering with a Fast Path-Capable System	21
Fast Path EMH and Shared Queues	22
Criteria for Fast Path Application Programs Using EMH	22
<b>Chapter 2. Planning the Network</b>	23
Planning for Network Administration	23
Documenting Network and Terminal Requirements	24
The Terminal Profile	24
Terminals Attached Using VTAM	24
Understanding the IMS Terminal Network	24
Terminal Connections to IMS	25
XRF Terminal Support	26
Logical Terminals	26
APPC/IMS and LU 6.2 Terminal Support	27
IMS Messages and Their Scheduling	27
Message Queuing	28
Message Segments	28
Invalid Destinations	28
Transaction Codes	29
Message Scheduling	29
CPI-C Transactions	29
Message Flow within the IMS Online System	30
Message Flow from Terminal to Program	30

Message Flow from Program to Output Terminal . . . . .	30
Message Flow from Program to Alternate Destination. . . . .	30
Message Flow from Program to Program . . . . .	30
Conversational Transactions . . . . .	31
Contents of the SPA . . . . .	31
Message Processing for Conversational Transactions. . . . .	31
ETO Conversations . . . . .	32
Program Switches for Conversational Programs. . . . .	32
Message Switches . . . . .	32
Logical Terminal Network Design . . . . .	32
Logical-Terminal Chains . . . . .	33
Logical-Terminal Queues . . . . .	33
Separating Input and Output Devices. . . . .	34
Logical and Physical Terminal Relationships . . . . .	35
Master Terminal . . . . .	35
Master Terminals in an XRF Complex . . . . .	35
NTO Terminals . . . . .	36
Terminal Modes and States . . . . .	36
Terminal and User Operating Modes . . . . .	37
Terminal and User States . . . . .	38
Planning for Security. . . . .	40
Transaction Authorization . . . . .	40
Using RACF and a Security Profile Based on User IDs . . . . .	41
Using SMU and a Security Profile Based on LTERM Names . . . . .	41
Using the Transaction Authorization Exit Routine (DFSCTRN0) . . . . .	41
Command Authorization . . . . .	41
Using RACF and a Security Profile Based on User IDs . . . . .	41
Using SMU and a Security Profile Based on LTERM Names . . . . .	41
Using the Command Authorization Exit Routine (DFSCCMD0) . . . . .	41
Transaction Command Security. . . . .	41
Password Security . . . . .	42
Using RACF and a Security Profile Based on User IDs . . . . .	42
Using SMU and a Security Profile Based on LTERM Names . . . . .	42
Security for APPC/IMS . . . . .	42
Security for ETO . . . . .	42
Planning for a Shared-Queues Environment . . . . .	43
Concepts for Operating in a Shared-Queues Environment . . . . .	43
The IMS Role in a Shared-Queues Environment . . . . .	43
How IMS Registers Interest in Queues . . . . .	44
Message Flow in a Shared-Queues Environment . . . . .	46
Tracking Units of Work . . . . .	46
Terminal Autologon . . . . .	46
Message Switches . . . . .	47
Dynamic Control Blocks . . . . .	47
The IMS Queue Manager . . . . .	47
Serial Transaction Processing . . . . .	47
Conversational Transactions in a Shared-Queues Environment . . . . .	48
Planning for Operating in a Shared-Queues Environment . . . . .	48
Configuring a Shared-Queues Environment . . . . .	48
Planning to Define List Structures . . . . .	50
Using Associated Printers . . . . .	52
Planning for IMS Front-End Switch . . . . .	52
Determining Eligibility for Sysplex-Wide Processing . . . . .	53
Managing the IMS Dead-Letter Queue . . . . .	54
Using the Expedited Message Handler . . . . .	54
Requeuing Suspended Transactions . . . . .	55

Scheduling AOI Transactions . . . . .	55
Planning for MSC in a Shared-Queues Environment . . . . .	55
Planning for RSR in a Shared-Queues Environment . . . . .	56
Planning for Generic Resource Groups . . . . .	56
Requirements for Using Generic Resource Groups . . . . .	57
Restrictions on Using Generic Resource Groups . . . . .	57
Managing Generic Resource Affinities Using IMS . . . . .	57
Managing Generic Resource Affinities Using VTAM . . . . .	58
Specifying Generic Resource Names . . . . .	58
Specifying APPC Generic Resource Names . . . . .	58
Initiating Sessions Using GRSNAMEs or APPLID names . . . . .	58
Changing the Logon Procedure . . . . .	58
Overriding the APPLID Field . . . . .	59
Determining When Affinities Are Terminated . . . . .	59
Terminating Affinities That Persist . . . . .	59
Dropping an IMS System from a Generic Resource Group . . . . .	60
Resetting Terminal Status . . . . .	60
Logging On after IMS Failure with Affinity . . . . .	60
Controlling Generic Resource Member Selection . . . . .	60
Ensuring Consistency across IMS Systems . . . . .	61
Bypassing Affinity Management During the IMS ESTAE Process . . . . .	62
Terminals Used with Fast Path . . . . .	62
Extended Recovery Facility (XRF) and IMS . . . . .	64
Terminals in an XRF Complex . . . . .	65
Class-1 Terminals . . . . .	65
Class-2 Terminals . . . . .	66
Class-3 Terminals . . . . .	66
Specifying Terminal Support . . . . .	67
<b>Chapter 3. Defining the Network . . . . .</b>	<b>69</b>
Preparing for the Operational Network . . . . .	69
Coordinating IMS Definition and Network Definition . . . . .	70
Using IMS as a Host Subsystem . . . . .	70
Defining VTAM Nodes . . . . .	71
Estimating VTAM Storage Requirements . . . . .	71
Determining VTAM Buffer Pool Values . . . . .	71
Determining the NCP Buffer Pool Values . . . . .	72
Determining Static and Dynamic Terminal Signon Requirements . . . . .	72
Multiple Signons . . . . .	72
Session Status Message—DFS3650 . . . . .	72
Checking Requirements for LOGON MODE Tables . . . . .	73
Coordinating LOGON MODE Identifiers . . . . .	73
Logon Requirements for the Master Terminal . . . . .	74
Specifying Initial VTAM Configurations . . . . .	74
Using SON/COS Support in IMS . . . . .	74
Network Operation . . . . .	75
Starting an IMS Network . . . . .	75
Session Initiation . . . . .	75
IMS Transaction Types and Transaction States . . . . .	76
Defining IMS Transaction Types . . . . .	76
Determining Transaction States . . . . .	77
Restarting the Network after Remote Takeover . . . . .	78
VTAM Definition for Transport Manager . . . . .	79
VTAM Buffering and Pacing . . . . .	79
VTAM Missing Interrupt Handler . . . . .	79

<b>Chapter 4. Using Shared Queues in the Network</b>	81
Enabling Shared Queues	81
Defining the CFRM Policy	81
Defining Message Queue List Structures	82
Defining Fast Path Message Queue List Structures	82
Defining an MVS Log Stream	83
CQS Definitions	83
IMS Definitions	85
Using the Common Queue Server	86
Starting CQS	86
Restarting CQS	86
CQS Warm Start	86
CQS Cold Start	87
Using IMS Commands to Access CQS	87
Using CQS User-Supplied Exit Routines	88
Changing the Size of Coupling Facility Structures	89
Initiating System Checkpoint	89
Checkpoint Data Sets	89
How CQS Restarts after System Checkpoint	89
Initiating Structure Checkpoint	89
Recovering Structures	90
Rebuilding Structures	90
Copying Structures	90
Recovering Structures	90
Deleting Message Queues	90
CQS Logging and the MVS Logger	91
Shutting Down CQS	91
Tuning for Performance in a Shared-Queues Environment	91
Tuning Structures and Parameters	91
Recommendation for Fast Path Transactions	92
<b>Chapter 5. Editing and Formatting Messages</b>	93
Message Format Service	93
MFS Components	94
Administering MFS	96
Advantages to Using MFS	97
MFS Control Blocks	98
Overview of MFS Components and Operation	99
Creating MFS Formats with SDF II	99
Basic Edit	101
IMS Editing for Intersystem Communication (ISC)	102
Transparency Option	103
Unprotected Screen Option	104
Bypassing MFS Editing	104
When To Bypass MFS Editing	104
Locking and Unlocking the Terminal Keyboard	104
IMS Sensitivity to Nongraphic Message Data	104
Output Message Segment Editing	104
Editing of Input Message Segments by MFS	105
Editing of Input Message Segments by Basic Edit	105
Controlling Output Devices	106
Small Buffer Devices	106
Controlling Output	107
Using a Printer Component	107
Spooled Output Control	108
Using Printer Components of the IBM 3270 Information Display System	108

Specifying Candidate Printers . . . . .	108
Operational Considerations . . . . .	109
Sharing Printers between Systems . . . . .	110



---

# Chapter 1. The IMS Transaction Manager Network

This chapter introduces the major network components, and describes the way communications are established, messages are transmitted, and communications are terminated between IMS and a logical unit.

## **In this Chapter:**

- “Introducing the IMS TM Network”
- “Introducing IMS Data Communication Services” on page 11
- “Using DC Control” on page 13
- “Operating an IMS Network” on page 14
- “Operating in a Shared-Queues Environment” on page 15
- “Using Generic Resource Groups to Balance Sessions” on page 19
- “Using the Fast Path Expedited Message Handler” on page 20

---

## **Introducing the IMS TM Network**

IMS is a general-purpose database/transaction manager system that provides the necessary support for an advanced telecommunications network. Virtual Telecommunications Access Method (VTAM) controls the physical transmission of data in the network, directing data to IMS from various logical units and from IMS to the appropriate logical units.

An IMS telecommunications network must include the following components:

- IMS
- Virtual Telecommunications Access Method (VTAM) <sup>3</sup>
- Communications hardware (such as control units)
- Terminals

The network can optionally include any of the following components:

- IMS Data Communications services, such as:
  - Extended Terminal Option (ETO)
  - Fast Path
  - Message Format Service (MFS)
  - Intersystem Communication (ISC)
  - Multiple Systems Coupling (MSC)
- IMS shared queues
- VTAM generic resource groups
- Advanced Program-to-Program Communication (APPC)
- Open Transaction Manager Access (OTMA)

In addition, an IMS telecommunications network can operate within one of the following frameworks:

- IBM Systems Network Architecture (SNA), which brings together multiple products in a unified design. SNA formally defines the functional responsibilities of the network components.

---

3. Although IMS uses the network facilities of VTAM, it can also control devices that use BTAM or BSAM. VTAM is the preferred access method for IMS.

## Introducing the IMS TM Network

- A client-server environment, using APPC or OTMA.

### Definitions:

- A *logical unit* is an addressable resource that can be an application program, a terminal, or a subsystem such as CICS. A logical unit can also be a component of a general-purpose terminal system that consists of a programmable controller and its attached operator terminals, printers, and auxiliary control units.
- The word *terminal* is used throughout this manual to describe devices and also applies to controllers and remote subsystems. The operator terminals can be keyboard printers, display stations with keyboards, communication terminals, or a mixture of these devices.

A network consisting of IMS and programmable logical units enables users to distribute functions throughout network components. This distribution of function reduces processing requirements that are placed on the central processor (also referred to as the *host*), and it can reduce the impact on the rest of the network when one component encounters a problem.

Figure 1 illustrates the components of a complete communications network system. The arrows in Figure 1 indicate communications that occur between components. Figure 1 shows the following components:

- IMS and its application programs
- VTAM
- The MVS/ESA operating system (including APPC/MVS if APPC/IMS is used)
- IBM 37x5 Communications Controllers and Network Control Programs (NCP)
- Two logical units (User terminals)

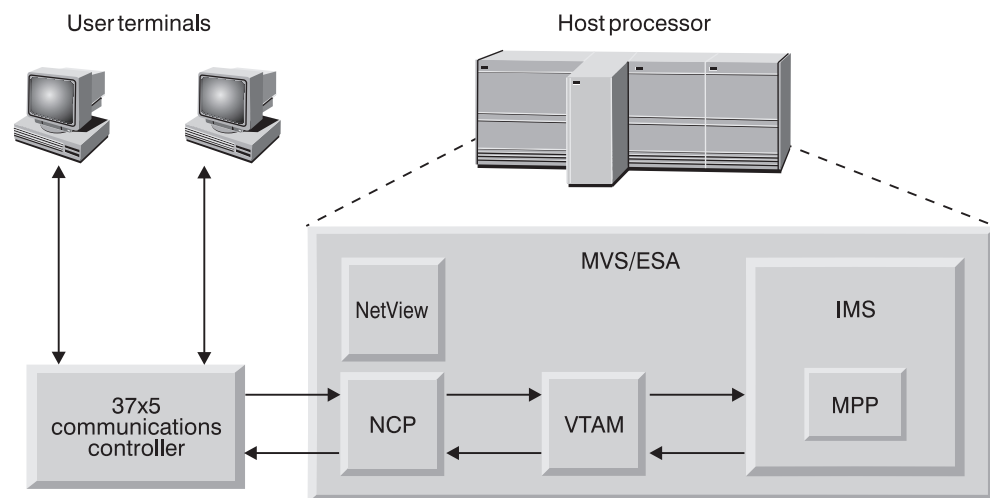


Figure 1. Components of a Network

The following summarizes how each of the components in Figure 1 participates in the network:

### Application Program

- Reads data from terminal
- Writes data to processor
- Reads data from processor
- Writes data to terminal



### Communications Controller

- Adds line control characters
- Transmits data
- Receives data
- Removes line control characters
- Checks for transmission errors
- Controls buffering

### Network Control Programs (NCP)

- Sends and receives data from communication lines and adapters
- Checks for and records errors

### Virtual Telecommunications Access Method (VTAM)

- Connects and disconnects terminal from the network
- Sends data from terminal to IMS
- Permits both monitoring and modifying of the network
- Sends data from IMS to the terminal
- Manages physical network (with NetView)

### IMS

- Checks transaction security
- Schedules the proper Message Processing Program (MPP)
- Directs output to the proper terminal
- Provides checkpoint and recovery capabilities

### MPP

- Processes input transactions
- Performs database processing
- Produces reply
- Executes application program

Planning an IMS network requires an understanding of each component and of its relationship to the others.

## Programmable Logical Units

### Definitions:

- A *programmable logical unit (LU)* is an input/output device that is in session with IMS. Application programs in remote logical units can be designed to control more than one terminal.
- When a logical unit informs VTAM that it wants to communicate with IMS, VTAM notifies IMS using the VTAM Logon exit routine. IMS then accepts the request, and VTAM logically connects the logical unit to IMS. This logical connection is called a *session*.

Some of the functions performed by the remote application program include:

- Reading from and writing to associated terminals
- Editing and verifying the data that is received from a terminal
- Reading from and writing to disk storage within the remote LU
- Reading from and writing to the host in which IMS is running
- Editing and verifying data that is received from the host in which IMS is running

## Introducing the IMS TM Network

- Communicating with other network logical units
- Formatting display and printer devices
- Operating offline when the host, VTAM, IMS, or NCP is unavailable

Depending on the system type, each LU can consist of one or more terminals. An application program that controls an LU consisting of more than one terminal or component must be able to direct output to a specific device. Therefore the application program must be capable of some form of data interrogation in order to make the proper device selection. IMS assists the application program in this process by:

- Allowing LU components to be defined and addressed individually
- Providing, in the header of each output message, the identification of the component to which the message is directed

## Communications Controller and NCP

VTAM uses the facilities of the Network Control Program (NCP) which runs in the 37x5 Communications Controllers. VTAM uses NCP in order to:

- Control lines and devices that are attached to the controllers
- Transmit data between the logical unit and the host processor
- Perform error recovery
- Collect statistics about the network

## VTAM

VTAM controls the allocation of network resources, and enables these resources to be shared among VTAM users. To VTAM, IMS is a single VTAM user; VTAM is unaware of the IMS application programs.

The IMS application programs use the IMS CALL interface to request IMS services; IMS uses VTAM macros in order to activate VTAM facilities.

If APPC/IMS is active, VTAM regards it as a separate user.

**Related Reading:** For more information on VTAM and how it is used, see *Network Program Products General Information*.

## IMS

IMS comprises two main product features:

- The Database Manager (IMS DB), which can control your databases
- The Transaction Manager (IMS TM), which can control your data communications and application programs

IMS provides:

- Standard functions required by application programs
- An execution environment for concurrently running application programs that serve many online users
- Control of full-function, MSDB, and DEDB databases

IMS and the application programs it controls run under MVS/ESA.

## Message Processing Programs

**Definition:** A *message processing program (MPP)* is an IMS application program that is driven by transactions and has access to online IMS databases and message queues.

MPPs process the input transactions, perform database processing, send replies, and execute other application programs.

---

## Introducing IMS Data Communication Services

This section describes several specialized DC services.

### APPC/IMS and LU 6.2 Devices

APPC/IMS is a part of IMS TM that allows you to use CPI Communications to build CPI application programs. APPC/IMS supports APPC with facilities provided by APPC/MVS.

IMS supports implicit and explicit application program interfaces (APIs) for APPC support. The implicit API for APPC support allows the IMS application program to communicate with APPC devices using the same techniques employed with other remote devices. This support allows an application program, written by a programmer who has no knowledge of APPC LU 6.2 devices, to:

- Be started from an APPC LU 6.2 device
- Receive input messages from originating LU 6.2 devices
- Send output messages back to originating LU 6.2 devices
- Start transaction programs on remote LU 6.2 devices

The same application program can work with all device types (LU 6.2 and non-LU 6.2) without new or changed coding.

The explicit application programming interface (API) for APPC support is the CPI-C interface, and it is available to any IMS application program. The IMS application program can issue calls <sup>4</sup> to APPC/MVS through this interface.

**Related Reading:** For more information on APPC/IMS, see:

- “Chapter 17. CPI Communications” on page 359
- “Chapter 18. Administering APPC/IMS and LU 6-2 Devices” on page 367

### Fast Path

Fast Path is capable of performing transaction and database processing at high rates. When your system requirements include a high transaction volume, using Fast Path can be advantageous if you do not require all the facilities of full-function processing. Examples of such applications include teller transactions in banking and point-of-sale transactions (inventory update) in retail marketing. Fast Path input and output messages use expedited message handling (EMH) and bypass message queuing and the priority-scheduling process.

**Related Reading:**

---

4. You can also use MVS ATBxxx call services. For information on these call services, see *MVS/ESA Programming: Writing Transaction Programs for APPC/MVS*

## Introducing IMS Data Communication Services

- For more information on the Expedited Message Handler (EMH), see “Using the Fast Path Expedited Message Handler” on page 20.
- For more information on message queuing and message scheduling, see “IMS Messages and Their Scheduling” on page 27.

## Extended Terminal Option (ETO)

IMS Extended Terminal Option (ETO) provides dynamic terminal and local and remote logical terminal (LTERM) support for IMS TM.

**Definition:** A *logical terminal (LTERM)* is a user destination. Each logical terminal has a name that points to one physical terminal.

You can add terminal hardware and LTERMs (users) to the IMS system without first defining them. ETO gives you the option of eliminating macro statements in the IMS system generation of VTAM terminals and LTERMs. ETO enhances the availability of your IMS system by eliminating the need for you to bring the system down in order to add terminals and LTERMs.

ETO also enhances security for the IMS TM user by associating all output with a specific user, instead of with a device. ETO requires the user to sign on.

**Restriction:** The Security Maintenance Utility (SMU) does not support ETO.

ETO reduces the IMS system generation time for those systems where the terminal network is defined dynamically.

### **Related Reading:**

- For more information on logical terminals (LTERMs), see “Logical Terminals” on page 26 .
- For more information on ETO, see:
  - “Chapter 6. Overview of the Extended Terminal Option” on page 115
  - “Chapter 7. Administering the Extended Terminal Option” on page 123

## Intersystem Communication (ISC)

You can exchange data between your IMS system and other external subsystems.

**Definition:** An *external subsystem* is a subsystem that provides a set of database resources that IMS can use, but does not control. Examples of external subsystems to which your IMS system can connect include:

- CICS
- Other IMS subsystems
- User-written subsystems

The session that is created between IMS and an external subsystem is called an *application-to-application session*. The IMS subsystem that operates in this session uses a function called Intersystem Communication (ISC), which uses SNA protocols.

**Related Reading:** For more information on ISC, see “Chapter 10. Overview of Intersystem Communication” on page 221.

### Message Format Service (MFS)

IMS Message Format Service (MFS) is a facility that formats messages to and from terminals, so that application programs need not deal with device-dependent data in input or output messages. An application program can format messages for different device types using a single set of editing logic, even when device input and output differ from device to device.

IMS MFS formats messages to and from user-written programs in remote controllers and subsystems so that application programs need not deal with the transmission-specific characteristics of the remote controller.

**Related Reading:** For more information on MFS, see “Message Format Service” on page 93.

### Multiple Systems Coupling (MSC)

Multiple Systems Coupling (MSC) enables you to enter transactions in one IMS system and process them in another IMS system. The responses from IMS can be returned to the terminals that entered the transactions, or to other terminals.

**Related Reading:** For more information on MSC, see “Chapter 8. Overview of Multiple Systems Coupling” on page 173.

---

## Using DC Control

DC Control (DCCTL) allows you to use the IMS Transaction Manager independently of the IMS Database Manager. DCCTL provides improved system performance in terms of throughput, system availability, and integrity. DCCTL can coexist with current IMS application programs and installed terminals.

The following scenarios involving DCCTL do not require modifications to your existing environment:

- Application programs that access external database managers can use DCCTL without any modifications. For example, DCCTL can function as a front-end transaction manager for DB2. DB2 subsystems do not require modifications in order to run with DCCTL.
- IMS exit routines and IMS application programs that access external subsystem resources in a DCCTL environment do not require modifications.

The following procedures might require modifications for a DCCTL environment:

- Operational procedures
- The general procedure that is produced by the system definition
- Application programs that contain a mixture of calls that access both external subsystems and IMS databases do require changes. DL/I calls result in a status code of AD.

Your existing system generation procedures support the generation of a DCCTL system. DCCTL executes with a collection of control blocks that identifies your data processing environment. These control blocks describe the system, data communication, and transaction manager components.

Using a DCCTL environment class system generation, you can generate a TM batch environment. Using TM batch, you can either take advantage of the IMS Batch Terminal Simulator (BTS) or access DB2 systems. TM batch support only allows DBB and DL/I regions. It does not provide DL/I database capabilities.

## Using DC Control

### **Related Reading:**

- For more information on accessing DB2, see *IBM DATABASE 2 Application Programming and SQL Guide*.
- For more information on the DCCTL environment, see *IMS/ESA Administration Guide: System*.

---

## Operating an IMS Network

Operating a basic IMS network involves:

- Establishing a communication session between a logical unit and IMS
- Sending data between a logical unit and IMS
- Terminating the session between a logical unit and IMS
- Restarting the session after a failure

## Operating the Network with APPC/IMS

APPC/IMS supports IMS commands for network operation, but the LU 6.2 device handles normal operations such as session startup, transaction initiation, and error handling that does not require master terminal operator (MTO) intervention.

## Initiating a Session with IMS

A session can be initiated by a logical unit, by the VTAM network operator, by the IMS MTO, automatically by VTAM, or by IMS itself. After a logical unit connects to IMS, it remains connected until one of the following actions occurs:

- The logical unit itself requests disconnection.
- The IMS MTO requests disconnection.
- Another VTAM application program requests connection to the terminal.
- IMS, VTAM, NCP, the logical unit, or the entire network is stopped.

After the physical connection between the controller and VTAM is established, an LU-to-LU session is initiated. The LU that is requesting the session informs VTAM that it wants to communicate with IMS. VTAM notifies IMS of the request through the VTAM Logon exit routine. IMS indicates that it will accept the request, and VTAM then logically connects the LU to IMS. A session is required before communication between the LU and IMS can be accomplished. To open a session, all nodes in the communication path (IMS, NCP, line, and station) must be in active status. After a session is established, VTAM directs all data between the logical unit and IMS.

IMS also supports SNA communication links in an Extended Recovery Facility (XRF) complex.

**Related Reading:** For information on establishing sessions in an XRF environment, see:

- *IMS/ESA Customization Guide*
- *IMS/ESA Administration Guide: System*

## Logging On and Signing Onto IMS

The following definitions apply to logging onto IMS and signing onto IMS:

### **Definitions:**

- *Logging on* to a terminal establishes a session with IMS for that terminal.

- *Signing on* to a terminal identifies a user to IMS.

## Logging Off and Signing Off from IMS

The following definitions apply to logging off from IMS and signing off from IMS:

### Definitions:

- *Logging off* from a terminal ends a session with IMS for that terminal.
- *Signing off* from a terminal ends an identification of a user to IMS.

---

## Operating in a Shared-Queues Environment

Operating in a shared-queues environment allows multiple IMS subsystems in a 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 systems, it distributes processing loads between the IMS subsystems. Transactions that are entered on one IMS system can be made available on the shared queues to any other IMS system that is 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 the processing is as if they were operating in a single-system environment.

### Definitions:

- A *shared queue* is a collection of messages that are associated by the same queue name. A shared queue is managed by a Common Queue Server (CQS) and can be shared by multiple clients (IMS subsystems).
- A *Common Queue Server (CQS)* receives, maintains, and distributes data objects from a shared queue that resides in a coupling facility list structure for its client.
- A *CQS client* is an IMS DB/DC or DCCTL subsystem that accesses shared queues through its own CQS.
- A *coupling facility* is a special, logical partition that provides high-speed caching, list processing, and locking functions in a Sysplex environment.
- A *Sysplex environment* is a set of MVS systems that communicate and cooperate with each other through certain multisystem hardware components and software services in order to process workloads.

Figure 2 on page 16 shows a basic shared-queues environment.

In general, IMS handles messages in the following manner:

1. IMS subsystems register interest in those queues for which they are able to process messages.
2. When an IMS system receives a message and places it on the shared queue, all IMS systems that have registered interest in that queue are notified.
3. One IMS system retrieves the message and processes it.
4. The IMS system that processes the message places a response on the queue.
5. The IMS system that submitted the original message is notified that the response message was placed on the queue.
6. The IMS system that submitted the original message sends the response message to the originating terminal.

## Operating in a Shared-Queues Environment

**Related Reading:** For more information on queue types, see “Queue Types” on page 44 .

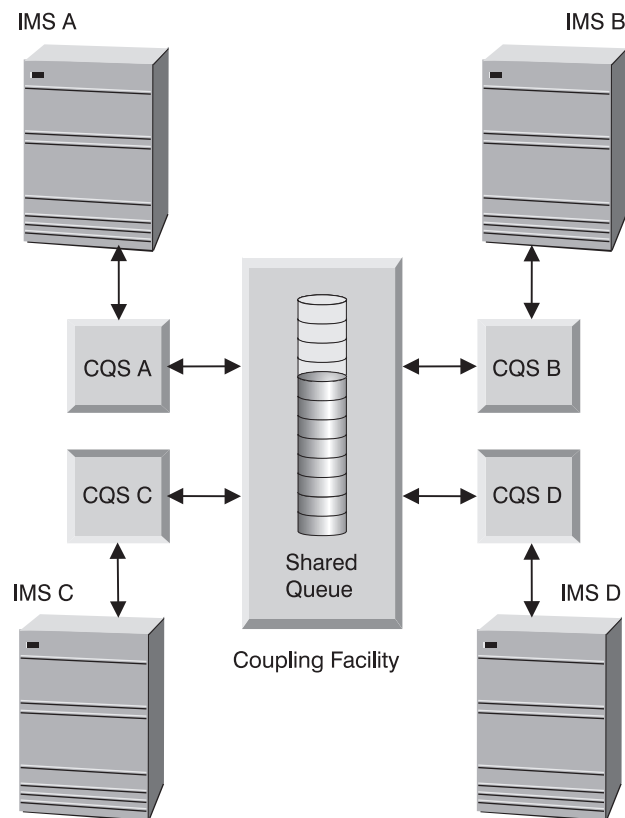


Figure 2. Basic Shared-Queues Environment

## Benefits of Using Shared Queues

The major benefits of operating in a shared-queues environment are:

### **Automatic workload balancing**

A message that is placed on a shared queue can be processed by any participating IMS system that is available to process 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.

### **Recommendations:**

- Enabling shared queues does not require the use of generic resource groups; however, IBM recommends that you use the two functions together.
- IBM recommends that all data in a Sysplex be shared across the Sysplex.

### **Related Reading:**

- For information on generic resource groups, see “Using Generic Resource Groups to Balance Sessions” on page 19.
- For information on data sharing, see *IMS/ESA Administration Guide: System*.



### Required Components of a Shared-Queues Environment

Although you can operate many different configurations of a shared-queues environment, the required components of shared-queues processing, shown in Figure 3 on page 18, include:

#### **Common Queue Server (CQS)**

One CQS is required for each client. Each CQS accesses the shared queues, which reside on coupling facility list structures.

**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.

##### **Definitions:**

- A *list structure* is an area of storage in a coupling facility that enables multisystem applications in a Sysplex environment to share information that is organized as a set of lists or queues. The list structure consists of a set of lists and an optional lock table.
- CQS maintains list structures in pairs, called *structure pairs*, with a primary list structure and an overflow list structure.
- The *primary list structure* contains the shared queues.
- The *overflow list structure*, if defined, contains shared queues that overflow after the primary list structure reaches a predefined threshold.

#### **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 so that shared queues on the structures can be recovered. The SRDSs maintain structure checkpoint information for the shared queues.

## Operating in a Shared-Queues Environment

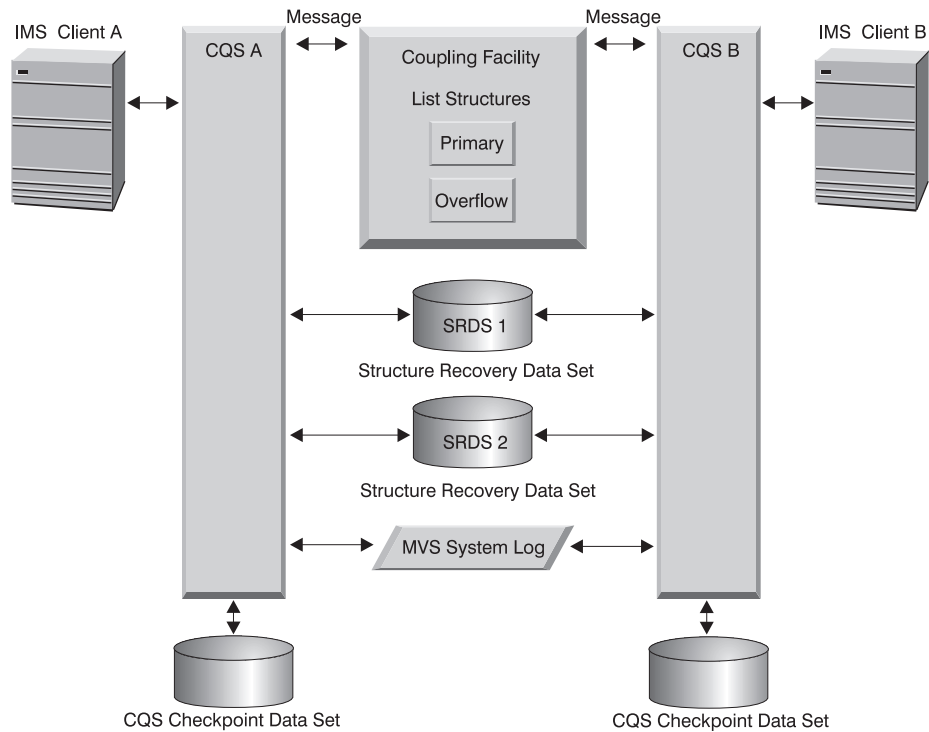


Figure 3. Components of a Shared-Queues Environment

**Related Reading:** For more information on CQS requests and other CQS topics, see *IMS/ESA Common Queue Server Guide and Reference*.

## Overview of the Common Queue Server

The Common Queue Server (CQS) is an internal interface by which IMS communicates 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 a 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 the following CQS client exit routines:
  - The Client CQS Event exit routine notifies the client of system events, such as CQS abnormal termination and CQS restart completion.
  - The Client Structure Event exit routine notifies the client of structure events, such as structure copy, structure recovery, structure overflow, structure checkpoint, and structure resynchronization.
  - The Client Structure Inform exit routine notifies the client when work exists on the shared queues.
- Drives the following CQS user-supplied exit routines:

## Operating in a Shared-Queues Environment

- The CQS Queue Overflow user-supplied exit routine allows the exit to approve or reject a queue that CQS selects for overflow.
  - The CQS Initialization/Termination user-supplied exit routine is notified when CQS initializes and when CQS terminates after disconnecting from all structures (under normal termination conditions)
  - The CQS Client Connection user-supplied exit routine participates in connecting clients to structures and in disconnecting clients from structures.
  - The CQS Structure Statistics user-supplied exit routine gathers structure statistics during CQS system checkpoints.
  - The CQS Structure Event user-supplied exit routine tracks structure activity, such as structure checkpoint, structure rebuild, structure overflow, and structure status change. It also tracks when CQS connects to a structure or disconnects from a structure.
- Provides the Log Print utility, with sample JCL that enables you to print log records from the MVS log.

### **Related Reading:**

- For more information on CQS, CQS client routines, and CQS utilities, see *IMS/ESA Common Queue Server Guide and Reference* .
- For information on IMS commands, see *IMS/ESA Operator's Reference*.
- For more information on enabling shared queues, see “Enabling Shared Queues” on page 81.

---

## Using Generic Resource Groups to Balance Sessions

If you are operating in a Sysplex environment and have multiple IMS systems, you can initiate a session by using the name of a 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 generic resource groups include:

### **Automatic session workload balancing**

Using generic resources is complementary to using shared queues; generic resources distributes network traffic among multiple IMS systems, while shared queues distributes back-end application workload.

### **Single-image resources**

You can use a single generic resource name to access multiple IMS systems, offering a single-system image while you are using the resources of many IMS systems.

### **Enhanced IMS system availability**

In general, if one IMS system fails, you can log on to another IMS system in that generic resource group.

**Exception:** If a terminal is ISC, SLU P, or Finance, you might not be able to log on to that terminal.

### **Share global messages**

You can obtain messages on shared queues from any IMS system in the generic resource group.

## Using Generic Resource Groups To Balance Sessions

**Recommendation:** If a terminal is in conversation or response mode, do not attempt to obtain messages from it.

### **Definitions:**

- A *generic resource group* is a set of IMS systems that have the same generic resource name, enabling VTAM to distribute terminal sessions among them.
- A *generic resource member* is an IMS system that belongs to a generic resource group.
- A *generic resource name* is the name by which IMS systems in a generic resource group are known to VTAM. Specify a generic resource name on the GRSNAME startup execution parameter.
- An *APPLID name* is the application program name by which VTAM knows an IMS system. The name is specified in a VTAM APPL definition statement and in the APPLID keyword of the IMS COMM system definition macro.

If you choose to specify the same APPLID name in COMM macros for the IMS systems in a generic resource group, you must specify a unique APPLID name in the IMS APPLID1 execution parameter override.

**Related Reading:** For more information on the APPLID1 execution parameter, see the COMM macro in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

When VTAM assigns your session to an IMS system in the group, the logical unit has an affinity to that IMS system.

**Definition:** An *affinity* is an association that a VTAM logical unit has with a specific IMS system in a generic resource group.

When you terminate the session by logging off, IMS terminates the affinity. In some cases, however, affinities remain after logging off.

### **Related Reading:**

- For more information on affinities, see “Determining When Affinities Are Terminated” on page 59.
- For information on planning for generic resources, see “Planning for Generic Resource Groups” on page 56.

---

## Using the Fast Path Expedited Message Handler

This section briefly describes the Fast Path expedited message handler (EMH) and how it processes Fast Path messages.

### Concepts for Understanding the Fast Path EMH

The following sections describe how the Fast Path EMH processes Fast Path messages.

#### **Fast Path Message Scheduling**

Fast Path schedules input messages by associating them with a load balancing group.

## Using the Fast Path Expedited Message Handler

**Definition:** A *load balancing group (LBG)* is a group of Fast Path input messages that are ready for balanced processing by one or more copies of a Fast Path program. One LBG exists for each unique Fast Path message-driven application program.

The messages for each LBG are processed by the same Fast Path program. The EMH controls Fast Path messages by:

- Managing the complete execution of a message on a first-in-first-out basis
- Retaining the messages that are received in the control program's storage without using auxiliary storage or I/O operations
- Supporting multiple copies of programs for parallel scheduling
- Requiring that programs operate in a wait-for-input mode

### Routing Codes and Balancing Groups

IMS places input messages on message queues by using the transaction code, and attempts to process one queue until that queue is exhausted, within limits specified by system definition. Message processing is grouped for load balancing and synchronized for database integrity and recovery.

#### **Definitions:**

- A *message queue* is a data set on which messages are placed before being either processed by an application program or sent to a terminal.
- A *transaction code* is a one- to eight-character alphanumeric code that invokes an IMS message processing program.
- A *routing code* is a user-defined code that the EMH uses to enable transactions to be routed to programs within a LBG. A message is assigned to a balancing group by a routing code. The routing code can be the same as the transaction code, or it can be any other name you choose.

Incoming messages have preassigned routing codes, or a routing code can be dynamically set for the input message. You declare routing codes during system definition, and they are then associated with individual application programs.

### Fast Path Input Edit/Routing Exit Routine (DBFHAGU0)

All Fast Path-exclusive or Fast Path-potential messages are sent to the Fast Path Input Edit/Routing exit routine so that the application program can examine, change, or edit the input message in the input area.

**Related Reading:** For more information on the Fast Path Input Edit/Routing exit routine (DBFHAGU0), see *IMS/ESA Customization Guide*.

### Message Buffering with a Fast Path-Capable System

IMS buffer management uses a single EMH buffer when the input message has all of the following characteristics:

- Source terminal is VTAM
- Terminal is not enabled for front-end switch (FES)
- Input is single segment
- Input is non-MFS edited
- Input is not an IMS command
- Execution of at least one transaction is through Fast Path

## Using the Fast Path Expedited Message Handler

An EMH buffer is dynamically allocated to the terminal when IMS receives the first message that has all of the characteristics listed above. The EMH buffer remains allocated to the terminal for use with subsequent input messages (that have the same characteristics) until the user signs off, or until the session terminates. The buffer is then released to the EMH pool.

### Fast Path EMH and Shared Queues

Using the Fast Path EMH in a shared-queues environment is very similar to using shared queues. Fast Path messages are placed on shared EMH queues, similar to how messages are placed on shared message queues.

**Related Reading:** For more information on shared queues, see “Chapter 4. Using Shared Queues in the Network” on page 81.

## Criteria for Fast Path Application Programs Using EMH

Application programs that use EMH must conform to all of the following criteria:

- Each transaction must be initiated by a single-segment, response-type message.
- Each input message must require only a single-segment response message or no response.
- The input and output message length cannot exceed the EMH buffer size.
- ETO and LU 6.2 users cannot access MSDBs with terminal-related keys.
- No IMS conversational processing is performed.
- Fast Path programs cannot act as automated operator programs or issue IMS commands.
- Fast Path transactions cannot be sent over any of the four types of MSC physical links for processing in another IMS system.

---

## Chapter 2. Planning the Network

This chapter describes information that will help you plan your IMS network.

### In this Chapter:

- “Planning for Network Administration”
- “Documenting Network and Terminal Requirements” on page 24
- “Understanding the IMS Terminal Network” on page 24
- “Logical Terminal Network Design” on page 32
- “Terminal Modes and States” on page 36
- “Planning for Security” on page 40
- “Planning for a Shared-Queues Environment” on page 43
- “Planning for Generic Resource Groups” on page 56
- “Terminals Used with Fast Path” on page 62
- “Extended Recovery Facility (XRF) and IMS” on page 64

---

## Planning for Network Administration

As an IMS network administrator, you should plan for each of the activities listed in Table 1.

Table 1. Network Administration Activities

Activity	Related Reading
Identifying the terminals and other devices required in the online system, verifying that IMS supports these devices, and identifying any incompatibilities.	For information on the terminals that IMS supports, the communication modes that IMS supports for each terminal, and the features for each terminal, control unit, or processor, see <i>IMS/ESA Release Planning Guide</i> .
Establishing appropriate security features in the network	For information on establishing security, see <i>IMS/ESA Administration Guide: System</i> .
Determining message editing requirements	For information on editing and formatting messages, see “Chapter 5. Editing and Formatting Messages” on page 93.
Determining requirements for exit routine	For information on using exit routines to customize your IMS network, see <i>IMS/ESA Customization Guide</i> .
Enforcing suitable naming conventions	For information on establishing naming conventions, see <i>IMS/ESA Administration Guide: System</i> .
Defining the network at IMS system definition, and coordinating the system generations for the host	For information on IMS system definition, see <i>IMS/ESA Installation Volume 2: System Definition and Tailoring</i> .
Defining ETO descriptors and MFS device characteristics table entries	For information on ETO and the MFS device characteristics table, see “Chapter 7. Administering the Extended Terminal Option” on page 123.
Defining LU 6.2 descriptors to APPC/IMS	For information on APPC/IMS and LU 6.2, see “Chapter 18. Administering APPC/IMS and LU 6-2 Devices” on page 367.
Testing the readiness of the network	For information on testing the network, see <i>IMS/ESA Administration Guide: System</i> .
Monitoring the performance of the network, and analyzing message loading, such as transaction routing and message queue size	For information on performance monitoring, see <i>IMS/ESA Administration Guide: System</i> .

# Documenting Network and Terminal Requirements

The size of your IMS installation and your terminal profile determine how you document your network requirements.

## The Terminal Profile

Creating a terminal profile is the best approach for gathering the information you need for your system definition. Your terminal profiles also assist you in coordinating the installation arrangements. They help you describe, to the system programming staff, how each terminal is to be used.

Begin by examining the application program specifications or the hardware plans to identify the terminals that are to be used.

## Terminals Attached Using VTAM

Terminals attached through VTAM are defined according to terminal type. Communication and attachment modes include:

- Binary synchronous communications (BSC)
- Synchronous data link control (SDLC)
- Local attachment (directly to a channel)

When you define your IMS system or select an ETO logon descriptor, your choice of terminal type determines the communication protocol and the MFS formatting that IMS uses for that terminal. You can use more than one terminal type for a particular physical terminal; however, your choice must be compatible with the application programs that execute on the IMS system.

---

## Understanding the IMS Terminal Network

An IMS network is characterized by physical terminals and other devices. IMS supports many terminals, including display devices, printers, and remote intelligent controllers.

IMS uses an access method, such as VTAM, to communicate with a physical terminal. Be sure to understand the difference between:

- VTAM's view of terminals that are connected to a host
- IMS's supervision of those connections

VTAM's responsibilities include:

- Receiving the input message and processing it
- Transmitting data
- Managing line and terminal communication status

IMS's responsibilities include:

- Monitoring the message traffic
- Routing messages to application programs
- Handling message storing, scheduling, and recording

Although IMS uses the network facilities of VTAM, it can also control devices that use BTAM and BSAM.



## Understanding the IMS Terminal Network

VTAM is the preferred access method for IMS. You can use OTMA for non-VTAM networks.

To describe your desired physical terminal configuration to IMS, you code system definition macros, such as the TYPE macro, that describe the terminal attachments and their characteristics (such as communication type, features, or options). You also define ETO and LU 6.2 descriptors.

### Terminal Connections to IMS

IMS supports various modes of communication or attachment, usually using VTAM. For BTAM, one major distinction is whether the attachment is local (through a channel) or remote (through a communication control unit). Remote attachments are either switched or nonswitched.

#### Definitions:

- *Switched* communication lines permit the attachment of only one remote workstation or terminal at a time to a line; they require that the terminal operator use a modem, which is attached to the remote terminal, in order to establish connection with the computer.
- *Nonswitched* communication lines are dedicated to the terminals that are physically attached to them. A nonswitched line can be either a contention or a polled line.
- A nonswitched line with *contention* allows only one contention-type terminal on the line at one time.
- A *polled* line permits one or more terminals to share the (multipoint) line.

Non-VTAM devices are described to IMS as line groups. Separate line groups are used for terminal configurations with the same communication mode, polling technique, or transmission code.

**Example:** A separate line group would be defined for a locally attached IBM 3270 Information Display System, as well as for a switched IBM 3270.

Each device in a line group is assigned a numeric identifier or address, called the *PTERM*. The order in which the terminals are physically defined to IMS during system definition determines the *PTERM* value.

**Example:** To make two physical terminals on a BTAM line available for use, an operator can enter the following command:

```
/START LINE 4 PTERM 1,2
```

Terminals that are attached by using VTAM are assigned a terminal type. The addresses of the lines and terminals that IMS uses are transparent to IMS, because they are not specified during IMS system definition. Instead, all VTAM resources have names assigned to them.

**Definition:** Each logical unit is assigned a *node name* that your installation defines.

The node name that is chosen for the VTAM definition is also used in IMS system definition and commands. Using VTAM enables IMS users to share network resources with other VTAM applications.

## Understanding the IMS Terminal Network

### XRF Terminal Support

XRF supports all terminals that IMS supports. XRF terminal support is divided into three categories:

- Class-1 terminal users receive the highest level of support from XRF, and they have backup sessions established with the alternate IMS system. When a takeover occurs, the Class-1 terminal sessions switch to the alternate IMS system without losing the sessions.
- Class-2 terminal sessions do not receive backup session support on the alternate IMS system, but they do receive automatic session restart after the takeover.
- Class-3 terminals must be reconnected to the new active IMS system using the same procedures that are in place for an IMS system that is not operating in an XRF complex. XRF cannot recover sessions for Class-3 terminals that were in use during the takeover.

Because APPC/IMS devices are Class 3, LU 6.2 conversations that are in progress at the time of failure are not preserved during a takeover. A new LU 6.2 conversation must be allocated from the new active IMS system after a takeover occurs. The new active IMS system processes APPC/IMS activity exactly the same as the old active IMS system did prior to takeover. The new active IMS system has the same LU name as the old active IMS system.

### Logical Terminals

#### **Definitions:**

- A *logical terminal (LTERM)* is a symbolic destination that is mapped to a VTAM node or a BTAM physical terminal by the IMS administrator.
- Each logical terminal has an installation-defined name, called the *LTERM name*.

**Related Reading:** For more information on LTERMs, see “Logical Terminal Network Design” on page 32.

Reserve one LTERM and identify it as the IMS master terminal. This logical terminal is the control point for the online IMS system. Commands associated with this LTERM start and stop the system, control the system resources, and display the status of those resources. The master terminal operator (MTO) can use the /ASSIGN command to alter the physical device that is associated with the LTERM. When ETO users sign on, altering the physical device that is associated with the LTERM is automatic. The device is identified by either the node name or the line number with the physical terminal identifier.

When a user enters a transaction, the logical terminal name is associated with the input message, and input messages are queued by transaction code. The output message queue designation is actually the LTERM name itself. Although this name is often the same for output as for input, an application program can cause the output to be directed to an alternate LTERM. An input terminal can also specify an LTERM name as the destination of a message. A message switching (input) edit routine can append the current input LTERM name to the message.

**Related Reading:** For more information on input edit routines, see *IMS/ESA Customization Guide*.

## APPC/IMS and LU 6.2 Terminal Support

APPC/IMS does not use an LTERM for input. APPC/IMS provides two facilities (that are similar to LTERMs) for determining output destinations:

- LU 6.2 descriptors allow 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 the same way as regular LTERMs in IMS application programs.
- Side information

**Definition:** *Side information* contains system-defined values provided by CPI Communications (CPI-C), and supplies LU 6.2 destination information. For APPC/IMS to establish a conversation with a partner program, CPI-C requires initialization information, such as the name of the partner program and the name of the LU at the partner's node. CPI-C provides a way to use system-defined values for these required fields. These system-defined values are specified in the side information, and are accessed by a symbolic destination name. The symbolic destination corresponds to an entry in the side information containing the partner\_LU\_name, the mode\_name, and the TP\_name.

**Related Reading:** For more information on side information, symbolic destination names, and the entries in the side information, see *OS/390 V1R1 MVS Planning: APPC Management*.

## IMS Messages and Their Scheduling

**Definitions:** An *IMS message* can be one of four types of data communication for which IMS controls processing:

### Transactions

*Transactions* are input messages that are destined for processing by application programs. Transactions are identified by a one- to eight-character *transaction code*. Transactions can be entered at terminals or generated by application programs. In a multiple-systems environment, a transaction can originate in a remote IMS system, or in another subsystem (like DB2).

### Messages sent to logical terminals (LTERMs)

Messages that are sent to LTERMs are identified by LTERM names. These messages can be either of the following types:

- Output messages sent from application programs to communicate with a logical terminal. They usually acknowledge or give results of work, and are sent to the originating terminal. Application programs can also send output to logical terminals other than the one that originated the input message.
- Input messages whose destination is a logical terminal. This message type is known as a *terminal-to-terminal message switch*. The text of the message becomes the output message to the destination LTERM.

### IMS commands

IMS commands are entered by terminal users who request that IMS display or alter the status of one or more IMS resources. Most commands originate from the master terminal operator (MTO), but other terminals, as well as application programs, can also enter them.

### DFSAPPC

DFSAPPC is an IMS system service that is used for exchanging messages

## Understanding the IMS Terminal Network

between LU 6.2 devices, or between any combination of LU 6.2 and non-LU 6.2 devices. Messages are delivered by allocating a new conversation with the designated LU 6.2 destination device. If the allocated conversation fails, the output is stored for future delivery. Messages are held on the IMS message queue until they can be successfully delivered.

**Related Reading:** For more information on DFSAPPC and message switching, see “Chapter 18. Administering APPC/IMS and LU 6-2 Devices” on page 367.

### Message Queuing

All input and output messages are queued in IMS-controlled virtual storage. Most command output is queued in virtual storage. Application programs handle messages serially if SERIAL=YES is specified on the TRANSACT macro.

If virtual storage is exceeded, messages are saved on direct-access storage. In this way, messages can be received as input or saved for output, although the resources that are necessary to process them might not be immediately available.

The maximum number of concurrent terminal I/O requests and the amount of virtual storage to hold the messages are specified during IMS system definition. Both variables can be altered when IMS is started.

### Message Segments

When a terminal is used to enter data into the IMS online system, the input can consist of either single segments or multiple segments. A sequence of several segments might be required to specify a message that can be interpreted by an application program. The device operation determines the end-of-segment.

For input, detection of the end-of-message condition by IMS indicates that a complete message has been received.

When the first end-of-segment condition is detected, IMS examines the beginning of the segment to find a destination. If a destination is declared to be a single-segment message, the end-of-segment signals the end-of-message.

After editing, the first (or only) segment of a message has the following structure:

<b>LL</b>	<b>ZZ</b>	<b>DEST-CODE</b>	<b>b</b>	<b>MMMM</b>
2 bytes	2 bytes	1 to 8 bytes	1 byte	message text

These parameters have the following meaning:

<b>LL</b>	The total length of the segment (including the LL and ZZ bytes)
<b>ZZ</b>	A reserved halfword
<b>DEST-CODE</b>	A one- to eight-character destination code (usually a transaction code)
<b>b</b>	A trailing blank
<b>MMMM</b>	The remainder of the message text

Each input message is uniquely identified by its destination code. The destination is normally a transaction code, but it can also be an IMS command or LTERM for message switching.

### Invalid Destinations

Even when the destination code meets normal resource naming conventions, the input destination can be incorrect:

## Understanding the IMS Terminal Network

- If ETO is not active, destinations that cannot be identified in IMS are considered invalid, and the input message is rejected.
- If ETO is active, IMS assumes these destinations are LTERM names and creates dynamic user structures. If the destination is invalid, a *dead-letter queue* is created. Several commands are provided, including the /DISPLAY command, for the MTO to process the dead-letter queues.

**Related Reading:** For more information on how ETO processes dead-letter queues, see “Asynchronous Output to an Invalid Destination” on page 164.

### Transaction Codes

The transaction code has optional attributes that affect the processing program’s scheduling eligibility by the IMS control program.

Application programs are defined in separate but related macro instructions. The application program that is designated to process a particular transaction code is considered by IMS to be another transaction attribute. An application program might process several transaction codes, but a transaction code can be associated with only one program.

### Message Scheduling

**Definition:** *Message scheduling* is the process by which a completely received input transaction is united with its associated application program for processing.

Some of the factors that affect the message scheduling process are:

- The variable attributes associated with the transaction code
- The number and relative importance of other transaction codes
- The number of received messages that are not yet processed
- The intent of associated application programs toward the data to be processed
- The amount of currently available space in control-block storage pools and buffers

In addition, each of the following activities affect the message scheduling process:

- Selecting system definition options
- Designing and using databases
- Specifying buffer sizes
- Declaring and selecting transaction code priorities

By properly combining these activities, you can manipulate the sequence in which messages are processed and enhance system performance.

**Related Reading:** For more information on monitoring the performance of the IMS network, see *IMS/ESA Administration Guide: System*.

### CPI-C Transactions

IMS resources are made available to the CPI Communications driven application program when the application program issues the APSB (Allocate\_PSB) call. The CPI Communications driven application program can use the SAA Resource Recovery Commit (SRRCMIT) and Backout (SRRBACK) calls to initiate an IMS synchronization point (referred to as a *sync point* throughout this manual) or backout. CPI Communications driven application programs should use the SAA Resource Recovery calls to initiate an IMS sync point prior to program termination.

## Understanding the IMS Terminal Network

**Recommendation:** Define CPI transactions with a different message class from that used for non-CPI transactions.

**Related Reading:** For more information on CPI Communications driven application programs, see “Chapter 18. Administering APPC/IMS and LU 6-2 Devices” on page 367.

## Message Flow within the IMS Online System

This section describes the events that take place as a message that is entered at a terminal flows through an IMS online system. Although some exceptions to the order of events and processing exist, the majority of transactions are processed this way.

IMS provides two ways to customize processing: basic edit (the default) and MFS.

**Related Reading:** For more information on basic edit and MFS, see “Chapter 5. Editing and Formatting Messages” on page 93.

### Message Flow from Terminal to Program

The initial entry from a terminal is a message that is processed asynchronously by IMS. After the message is received from the terminal, the message is formatted using basic edit, MFS, or exit routines.

The IMS message scheduler then uses the status of the message queues and a prioritizing algorithm in order to select the next transaction to schedule. When the program is selected for scheduling in the dependent region, the first segment of the message is made available to the program.

### Message Flow from Program to Output Terminal

While the application program is executing, it can use DL/I calls to access databases. The data communication support of DL/I is used by the program to request messages, using the GU function for the first message segment and the GN function to retrieve subsequent message segments. The program can then send a response to the entering terminal by inserting the reply to the I/O PCB. The entering terminal's LTERM is the I/O PCB after a successful GU call.

If the message is handled by MFS, it can be transformed from the program output format to the device output format. Otherwise, the message is passed unchanged to the device.

When the output message has been completely received by a terminal or by a program, it is dequeued. Any failure in message delivery causes IMS to keep the recoverable message queued for a later delivery.

### Message Flow from Program to Alternate Destination

The application program can also send output to an alternate destination. Using an alternate PCB, the program can insert a message to another LTERM. In this case, the message handling is the same, but the message queue on which it is placed has a different LTERM name.

### Message Flow from Program to Program

The application program can also use the alternate PCB to send a message to a program; that is, it can generate a secondary transaction that is placed on a

message queue. The processing of the secondary transaction is dependent on the selection of the transaction by the scheduling algorithm.

### Conversational Transactions

**Definition:** *Conversational transaction processing* allows you to retain message continuity from a given terminal, even when the program that processes the conversation is not retained in storage throughout that conversation.

When the transaction is defined as conversational, the application program can use a scratch pad area (SPA) in order to interrelate messages from a given terminal.

**Definition:** The *scratch pad area (SPA)* is a work area used in conversational processing in order to retain information from an application program across executions of the program. A unique SPA is created for each conversation.

#### Contents of the SPA

Typical contents of the SPA are data items entered at the terminal and data that is retrieved from databases to be saved between iterations of the conversation. To simplify operator procedure and application design, database updates are retained in the SPA until the last iteration of the conversation.

Any subsequent data entry from a terminal that is already operating in conversational mode causes a message processing program (MPP) to receive both the contents of the SPA and the input terminal data. Each input message is considered an individual unit of work. Each interaction can be with a different program, although the same program is typically used.

#### Message Processing for Conversational Transactions

When the message is a conversational transaction, the following sequence of events occurs:

- IMS removes the transaction code and places it at the beginning of a message segment. The message segment is equal in length to the SPA that was defined for this transaction during system definition. This is the first segment of the input message that is made available to the program. The second through the *n*th segments from the terminal, minus the transaction code, become the remainder of the message that is presented to the application program.
- When the conversational program has prepared its reply, it inserts the SPA to IMS. The program then inserts the actual text of the reply as segments of an output message.
- IMS saves the SPA and routes the message to the input LTERM.
- If the SPA insert specified that another program is to continue the same conversation, the total reply (including the SPA) is retained on the message queue as input to the next program. This program then receives the message in a similar form.
- A conversational program must be scheduled for each input exchange. The other processing continues while the operator at the input terminal examines the reply and prepares new input messages.
- To terminate a conversation, the program places blanks in the transaction code field of the SPA, and inserts the SPA to IMS.
- The conversation can also be terminated if the transaction code in the SPA is replaced by any nonconversational program's transaction code, and the SPA is inserted to IMS. After the next terminal input, IMS routes that message to the other program's queue in the normal way.

## Understanding the IMS Terminal Network

### ETO Conversations

ETO conversations maintain associations with users, rather than with terminals. The conversation is associated with the terminal only while the user is signed on. The conversation can be restarted on any other ETO terminal.

**Related Reading:** For more information on ETO, refer to “Chapter 7. Administering the Extended Terminal Option” on page 123.

### Program Switches for Conversational Programs

A conversational program can use a deferred program switch or an immediate program switch.

#### **Definitions:**

- During a *deferred program switch*, the program responds to the originating terminal but causes the next input from the terminal to go to another conversational program.
- During an *immediate program switch*, the program passes the SPA (and, optionally, a message) to another conversational program without responding to the originating terminal. In this case, it is the next program’s responsibility to respond to the originating terminal.

**Related Reading:** For more information on conversational programs and message switching, see *IMS/ESA Application Programming: Transaction Manager*.

## Message Switches

**Definition:** A *message switch* occurs when an input message specifies an LTERM name instead of a transaction code. IMS formats the message and inserts it into the output message queue for that LTERM.

---

## Logical Terminal Network Design

The IMS system definition describes the characteristics and relationship of:

- Communication lines
- Static terminals
- Logical terminals <sup>5</sup> (LTERMs)

When only one user operates a physical terminal, only one logical terminal is associated with that physical terminal. If multiple users operate a physical terminal, the terminal is associated with many logical terminals it.

You can structure the IMS system definition so that a separate logical terminal is assigned for each user of a particular static terminal.

**Definitions:** The information in this section uses *physical terminal* to represent both:

- A *node*, for VTAM devices
- A *PTERM*, for BTAM devices

---

5. For a nonswitched terminal, the relationship between a physical terminal and a logical terminal within IMS is a static relationship defined during system definition.



## Logical Terminal Network Design

Logical terminals can be assigned to physical terminals for both input and output. When a logical terminal is assigned to a physical terminal for output, all messages that are sent to that logical terminal are transmitted to its associated physical terminal. More than one logical terminal can be assigned to a given physical terminal for output. Only one physical terminal can receive the output for a given logical terminal. The relationship between physical and logical terminals for output is shown in Figure 4.

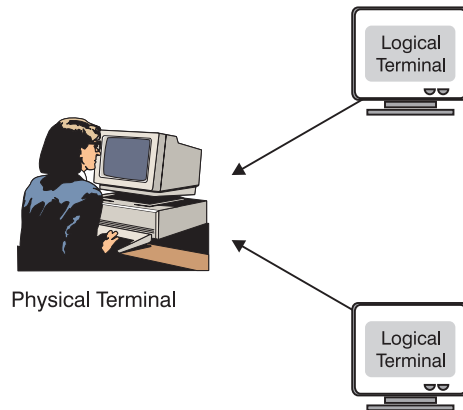


Figure 4. Relationship between Physical and Logical Terminals for Output from IMS

## Logical-Terminal Chains

When a logical terminal is assigned to a physical terminal for input, any message that is entered from the physical terminal is considered to have originated at the logical terminal. When more than one logical terminal is assigned to a physical terminal for input, a chain of input logical terminals is formed. Any input from the physical terminal is considered to have originated at the first logical terminal on the chain. Figure 5 shows the relationship between physical and logical terminals for input.

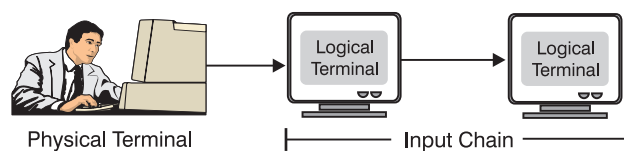


Figure 5. Relationship between Physical and Logical Terminals for Input to IMS

If the first logical terminal is not allowed to transmit a message (for example if it is not authorized or it is a stopped logical terminal), all logical terminals on the input chain are interrogated in chain sequence for their ability to transmit messages. If the physical terminal is a Finance, SLU 1, SLU P, or LU 6.1, only the logical terminals associated with the input component are scanned. The first appropriate logical terminal found is considered the originator of the message. If no appropriate logical terminal is found, the message is rejected with an error message.

## Logical-Terminal Queues

Using a message queue for received input messages or for pending output messages enables an application program to be independent of the arrival time and message transmission. Because this message queue is associated with a logical

## Logical Terminal Network Design

terminal, rather than with a physical terminal, the message queue can be moved from device to device, independent of the application program. The message queue can even be moved among device classes.

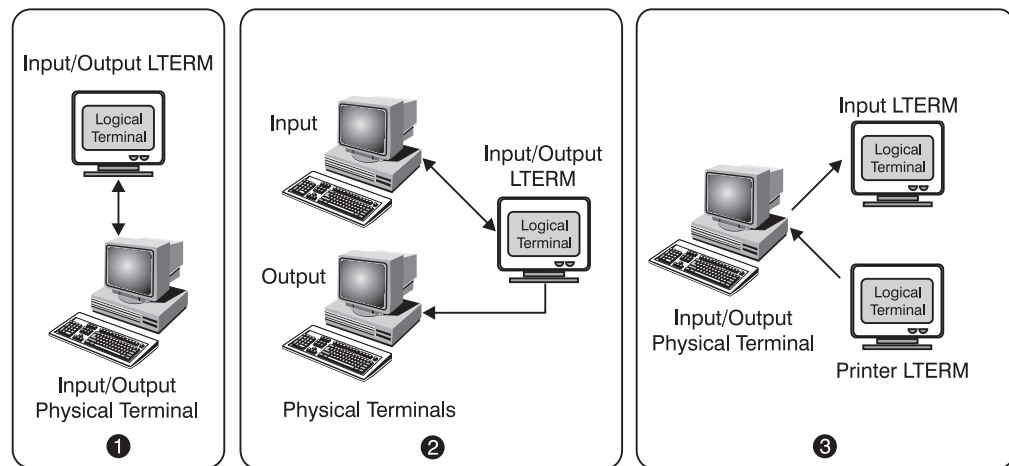
Logical terminals provide stability for application programs. Regardless of how the physical terminal network changes, the application programs remain insensitive to these changes.

The application program interface to a logical terminal is conceptually the same as that for the database system:

- GU calls retrieve message segments from a queue.
- ISRT calls insert message segments to a queue.

## Separating Input and Output Devices

In certain application programs, it might be necessary to associate a different physical device for output than the one that is used for input. If the physical terminal type is an input-only device and output is required, a different device must be associated for output. IMS system definition and commands support assignment of output devices that are different from the input device. For example, the application program that is processing a message from a display might need to send some of the output to a printer. The possible physical-to-logical relationships are shown in Figure 6.



### Notes To Figure Regarding Applications

1. Normal assignment of one or more logical terminals or physical terminals. Output is sent to input terminal. Application is insensitive.
2. Alternative assignment: Input and output sent using same logical terminal. Output is sent to different physical terminal. Application is insensitive.
3. Application uses specific logical terminal for output. Application is insensitive to input.

Figure 6. Possible Physical-To-Logical Terminal Relationships

## Logical and Physical Terminal Relationships

Figure 7 shows the communication flow between a terminal user, a physical terminal, a communication line, and a logical terminal in a nonswitched communications network.

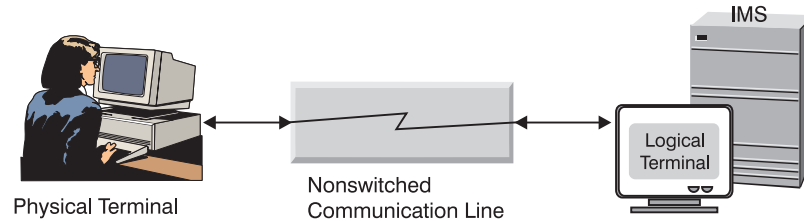


Figure 7. A Nonswitched Communications Network

IMS system definition describes the characteristics and relationship of physical terminals, communication lines, and logical terminals. On a nonswitched communication line, the relationship between a physical terminal at one end and a logical terminal within IMS at the other end is a stable relationship defined during system definition or during signon for ETO terminals.

Except for LU 6.1, terminals using VTAM have the same physical-to-logical terminal (PTERM-to-LTERM) relationship as a terminal on a BTAM nonswitched line.

The relationship that is established between a physical terminal and one or more logical terminals at system definition can be changed using commands or by creating a new system definition. The /ASSIGN command dynamically changes logical and physical relationships, and it is usually executable only from the master terminal.

## Master Terminal

The master terminal is the IMS control center. If a VTAM master terminal is defined, IMS establishes a connection with it during startup.

**Recommendation:** Although several device types are supported, a VTAM SLU 2 device is the preferred type for the master terminal.

**Restriction:** Neither an ETO terminal nor an LU 6.2 terminal can be defined as the primary or secondary master terminal.

The master terminal operator (MTO) must be familiar with all aspects of operating the system. The physical location of the master terminal in relation to the computer console is important. If, for security reasons, the MTO and master terminal are not close together, telephone communication should be provided.

## Master Terminals in an XRF Complex

In an XRF complex, each IMS system must have its own master terminal, and it can have a secondary master terminal. Typically, the secondary master terminal is a printer (output-only device type). System messages for the active and alternate subsystems go to their respective master terminals.

## Logical Terminal Network Design

During a takeover, the console operators must know the status of the takeover on both systems. The operator at the alternate subsystem must know when I/O prevention in the failed active subsystem is complete; terminal switching of Class–3 terminals and failed Class–2 switch attempts must be handled. If possible, the two master consoles should be close to each other physically.

**Related Reading:** For a list of devices that can be defined as primary and secondary master terminals, see *IMS/ESA Administration Guide: System*.

## NTO Terminals

When using the NTO licensed product, VTAM can support some start-stop and teletype devices, known as *NTO devices*.

The receipt of a message can be confirmed by using transaction-response mode to “lock” the keyboard following the input entry. When IMS responds with available output to a particular terminal for a specific transaction, it “unlocks” the keyboard and sends the data.

Even if the NTO devices are used primarily for inquiry transactions, the transactions can be made recoverable. Because input is not acknowledged, the application program should indicate in the output response the inquiry transaction that caused it.

Both input and output editing routines <sup>6</sup> can be made available for transactions using NTO devices.

Because NTO uses VTAM, IMS is not aware of whether a terminal user is logging on to a session or is logging off. Therefore, existing session status might cause problems when one terminal user ends a session and another terminal user resumes the session. This problem can be minimized by defining all NTO devices as Response mode.

To avoid the problem of continuing session status, a terminal user can clear any preset transaction codes that were previously set using /SET commands by entering the /RESET command either:

- Before issuing the /RCLSDST command when ending an NTO session
- Before issuing the /SIGN ON command when starting an NTO session

A terminal user who enters /RCLSDST or the MTO who enters /CLSDST terminates a session but does not reset conditions that were set during that session.

**Related Reading:** For more information on dynamic NTO terminals, see “NTO Devices” on page 130.

---

## Terminal Modes and States

IMS keeps two types of status information for the terminals in the network:

- The mode of operation
- The state of an inoperable terminal (such as temporarily unusable)

---

6. These routines are provided by your installation and are entered after MFS processing.

## Terminal and User Operating Modes

A terminal can be in more than one of the following modes at the same time.

### Response Mode

Response mode is established through static terminals, ETO users, or transaction specifications. Response mode can be used with full function or Fast Path processing. In response mode, after the entry of an input message, the terminal is locked until a reply is received, and no additional incoming data is accepted.

#### Definitions:

- For operator-driven terminals like 3270s, *locked* means that the keyboard is locked.
- For programmable terminals like LU Ps, *locked* means that IMS delays the acknowledgment of the input message until the output message is ready to be sent.

For LU 6.2, the originator of the transaction must issue a Receive command to get the response message (synchronous output). Failure to do so is treated as a protocol violation. Depending on the system, the application program could fail and IMS could send an error message.

Terminal-response mode is terminated when the response has been sent and dequeued. If Fast Path is used, terminal-response mode is automatically continued on an IMS restart. If full-function operation is used, terminal-response mode is terminated when IMS is restarted. The ETO user, not the terminal, is in response mode. If the user signs off, the user is still in response mode when signing back on. The user cannot enter another transaction until the response to the first transaction is received.

**Related Reading:** For more information on ETO terminals, see “Chapter 7. Administering the Extended Terminal Option” on page 123.

### Conversation Mode

After you enter a transaction that is defined using the SPA= parameter on the TRANSACT macro, the terminal is in conversation mode. While the terminal is in conversation mode, other transactions cannot be entered from that terminal. However, the terminal is not locked, in the same sense that response mode locks the terminal. The terminal remains in conversation mode until the conversation is terminated, when the message has been sent and dequeued, and the application program has placed blanks in the transaction code field in the scratch pad area (SPA).

A conversation can abnormally terminate under the following conditions:

- When an application program abends
- When the IMS MTO issues an /EXIT command, a /START NODE command, or a /START USER command
- When an inconsistent definition exists in the application program between IMS and MSC

## Terminal Modes and States

For LU 6.2, all iterations of the IMS conversation must use the same LU 6.2 conversation; each iteration of the IMS conversation is demarcated by using the LU 6.2 CMPTR (Prepare\_To\_Receive) call. If the LU 6.2 conversation ends prior to the end of the IMS conversation, the IMS conversation is abnormally terminated.

The ETO user, not the terminal, is in conversation mode.

### Exclusive Mode

A terminal is placed in exclusive mode when the /EXCLUSIVE command is issued. The Exclusive mode:

- Restricts the output received by the terminal.
- Remains with ETO users after they sign off. ETO users that sign off while they are in exclusive mode remain in exclusive mode when they sign on the next time.
- Terminates with an /END or /START NODE command.

### Lock Mode

Lock mode prevents a terminal from sending and receiving messages. A terminal, node, or logical terminal (LTERM) is placed in lock mode when the operator issues the /LOCK command. Lock mode is reset by issuing the /UNLOCK command or the /IAM command.

### Test Mode

Test mode ensures that any input message entered into a terminal is transmitted back to the terminal. A node is placed in test mode by the /TEST command. Test mode is reset by an /END command or a /START command. Test mode remains with ETO users after they sign off. ETO users that sign off while they are in test mode remain in test mode when they sign on the next time.

**Related Reading:** For more information on the /EXIT, /START NODE, /START USER, /EXCLUSIVE, /END, /LOCK, /UNLOCK, /IAM, and /TEST commands, see *IMS/ESA Operator's Reference*.

## Terminal and User States

The states in which a terminal is inoperable include:

### Stop State

The stopped state prevents the delivery of any output queued on a logical terminal that is associated with a physical terminal.

The /STOP NODE command results in the termination of the session between IMS and the node. This termination occurs immediately for most devices but only at the end of the message for 3270 devices and SLU 2 devices. The /STOP NODE command also prevents a new session until a /START command or /RSTART command is issued.

The /STOP command, the /PSTOP command, and the /MONITOR command also cause a terminal to enter the stopped state. This state is reset by the /START command or /RSTART command. The /STOP NODE or USER command allows ETO users to retain their status after signoff or logoff.

### QERROR State

A logical terminal is placed in a stopped state if an I/O error is encountered while attempting to read from or write to a message queue. This condition is reset when the MTO operator issues a /START command.

### **QLOCK State**

A logical terminal is prevented from sending any additional output until the locked state is reset by a specific request received in the LU 6.1 session. The LTERM is also prevented from receiving input that might create additional output. This condition is reset when the MTO issues a /START command. A /PSTOP or /PURGE command is ignored for LTERMs that are in QLOCK state.

### **INOP State**

The physical terminal is considered inoperable by IMS device support whenever an error is detected and put in the INOP state. All logical terminals associated with this physical terminal are also considered inoperable. The /START or /RSTART command resets the inoperable condition.

### **COMPINOP State**

Component inoperative can be set in one of two ways:

- When an error is detected that is isolated to a component of the terminal
- When the /COMPT or /RCOMPT command is issued for terminals that are defined to VTAM

All logical terminals associated with this component are ineligible for message output. The component inoperative state is reset when the operator issues a /START LINE PTERM command, a /START NODE command, another /COMPT command, or a /RCOMPT command. Special signals from the device, such as device end from a 3270 device or the “component available” status from a SLU 1 can also cause a reset.

**Related Reading:** For unique device considerations, see *IMS/ESA Operations Guide*.

### **PAGE, SCREEN, and COMPONENT PROTECTION State**

This is a state supported for video terminals, SLU P, Finance, and LU 6.1 devices. Logical terminals associated with these physical terminals are not eligible for output selection.

**Related Reading:** For more information on screen protection for the IBM 3270 devices, see *IMS/ESA Operations Guide* and *IMS/ESA Application Programming: Transaction Manager*.

### **SNA QUIESCE State**

When IMS sends output messages to a VTAM programmable node and the node wants to stop receiving, the node signals IMS to stop transmissions after an end-of-chain has been sent. IMS does not send any more output to the terminal until the terminal sends the SNA release-quiesce (RELQ) command.

### **Related Reading:**

- For more information on the quiesce-at-end-of-chain (QEC) function, see “Suspending Output from IMS” in *IMS/ESA Customization Guide*. This state is only for Finance and SLU P terminals.
- For more information on ETO and user states, see “Setting Special Processing Modes” on page 155.

### Planning for Security

To prevent unauthorized use of a terminal in the IMS network, you can use two types of security:

- RACF, which is a licensed program available under the MVS/ESA operating system (or an equivalent product)
- IMS Security Maintenance Utility (SMU)

RACF and SMU allow you to control access to:

- Physical terminals
- Logical terminals
- Transactions
- Commands

If you do not use either of these security options, IMS only allows certain commands to be entered at user terminals (excluding the master terminal). This is called *default terminal security*.

**Related Reading:** For a list of commands that cannot be entered at user terminals, see *IMS/ESA Operator's Reference*.

RACF and SMU provide two different ways to configure your IMS network security profiles:

- Using RACF, you can design security profiles based on user ID.
- Using SMU, you can design security profiles based on LTERM names.

You can define two levels of security for your network:

- You can control the use of the terminals connected to your network.
- You can control the resources that can be accessed from the terminal.

You control use of a terminal by signon verification security. For example, a terminal user enters an identifier as a parameter on a /SIGN command or in response to a DFS3649 message. You can use RACF, an exit routine, or both to validate the signon. The user ID is logged with each input and output message and with each database change.

The following sections describe the methods you can use to control access to resources from a terminal:

- "Transaction Authorization"
- "Command Authorization" on page 41
- "Transaction Command Security" on page 41
- "Password Security" on page 42
- "Security for APPC/IMS" on page 42
- "Security for ETO" on page 42

### Transaction Authorization

During transaction authorization, RACF or SMU checks to ensure that the terminal user or the LTERM is authorized to enter the transaction.



### **Using RACF and a Security Profile Based on User IDs**

To control which users can issue which transactions, define controlled transactions to RACF as TIMS class, and grant authorization to RACF-defined users or groups of users.

You can also use System Monitoring Facility (SMF) logging to track the successes and failures of transaction authorization. Using RACF, request auditing capabilities for your transaction security profiles.

### **Using SMU and a Security Profile Based on LTERM Names**

To control which transactions can be issued from a terminal, define authorized transactions using the SMU macro as input to the IMS Security Maintenance Utility.

### **Using the Transaction Authorization Exit Routine (DFSCTRN0)**

You can use the Transaction Authorization exit routine (DFSCTRN0) to check the validity of a transaction that is entered by a user who signed on using the /SIGN command. You can use the Transaction Authorization exit routine with both RACF and SMU.

**Related Reading:** For more information on using the Transaction Authorization exit routine (DFSCTRN0), see *IMS/ESA Customization Guide*.

## Command Authorization

For command authorization, RACF or SMU checks to ensure that the terminal user is authorized to enter a command.

### **Using RACF and a Security Profile Based on User IDs**

To control which users can issue controlled commands, define controlled commands to RACF as CIMS class, and grant authorization to RACF-defined users or groups of users.

You can use SMF logging to track the successes and failures of command authorization. Using RACF, request auditing capabilities for your command security profiles.

### **Using SMU and a Security Profile Based on LTERM Names**

To control which commands can be issued from an LTERM, define authorized commands using the SMU macro as input to the IMS Security Maintenance Utility.

### **Using the Command Authorization Exit Routine (DFSCCMD0)**

You can use the Command Authorization exit routine (DFSCCMD0) to restrict keywords and parameters by editing the input command buffer. The Command Authorization exit routine checks the validity of commands. You can use this exit routine with both RACF and SMU.

**Related Reading:** For more information on using the Command Authorization exit routine (DFSCCMD0), see *IMS/ESA Customization Guide*.

## Transaction Command Security

Transaction command security and password security provide another level of access control. Transaction command security is indirectly related to a terminal. If the terminal user enters a transaction to start a program that issues IMS commands, both the transaction code and the set of commands that program can issue must be authorized.

## Planning for Security

### Password Security

Password security requires that a transaction or a command that is entered at a terminal have a password. This allows you to implement secondary security to verify that the user who is issuing a specific transaction or command is authorized to do so.

#### Using RACF and a Security Profile Based on User IDs

RACF provides the REVERIFY option, which requires the user's password to be entered when a command or a transaction is entered. You need to use RACF for signon authorization and include the REVERIFY parameter as APPLDATA when defining the transaction or command to RACF. You must specify RVFY=Y as an execution parameter in order to use REVERIFY.

#### Using SMU and a Security Profile Based on LTERM Names

SMU's password security allows you to associate a password with controlled transactions and commands. Password security, if defined, is in addition to LTERM security.

### Security for APPC/IMS

APPC/IMS requires security using the System Authorization Facility (SAF) interface to RACF, or an equivalent security environment. RACF is optional for remote transactions from LU 6.2 application programs.

APPC/IMS supports both the Transaction Authorization exit routine (DFSCTRNO) and the Command Authorization exit routine (DFSCCMD0).

#### **Restrictions:**

- APPC/IMS does not support SMU security.
- APPC/IMS does not support the /SIGN command, because it is not required in order to validate the user ID. MVS validates user IDs when using RACF; therefore, each APPC/IMS message has a validated user ID.

**Related Reading:** For information on the AUTH call for APPC/IMS, see *IMS/ESA Application Programming: Transaction Manager*.

### Security for ETO

If you use SMU security for static terminals and RACF security for ETO terminals, you might be in the position of having two security profiles:

- One based on the LTERM name
- One based on the user ID

You only need one security profile when using RACF security for command and transaction authorization on both static and ETO terminals.

**Restriction:** SMU does not support ETO terminals.

**Related Reading:** For more information on ETO security, see "Planning a High-Security Environment with ETO" on page 134.

## Planning for a Shared-Queues Environment

To manage message queues, your installation can use the IMS queue manager with either message queue data sets or shared queues.

**Restriction:** If you use shared full-function message queues on a Fast Path-eligible system, you must define shared queues for EMH messages, as well.

This section provides an overview of a variety of concepts that are integral to operating in a shared-queues environment.

- “The IMS Role in a Shared-Queues Environment”
- “How IMS Registers Interest in Queues” on page 44
- “Message Flow in a Shared-Queues Environment” on page 46
- “Tracking Units of Work” on page 46
- “Terminal Autologon” on page 46
- “Message Switches” on page 47
- “Dynamic Control Blocks” on page 47
- “The IMS Queue Manager” on page 47
- “Serial Transaction Processing” on page 47
- “Conversational Transactions in a Shared-Queues Environment” on page 48

The section then describes planning and administrative tasks for operating in a shared-queues environment:

- “Configuring a Shared-Queues Environment” on page 48
- “Planning to Define List Structures” on page 50
- “Using Associated Printers” on page 52
- “Planning for IMS Front-End Switch” on page 52
- “Determining Eligibility for Sysplex-Wide Processing” on page 53
- “Managing the IMS Dead-Letter Queue” on page 54
- “Using the Expedited Message Handler” on page 54
- “Requeuing Suspended Transactions” on page 55
- “Scheduling AOI Transactions” on page 55
- “Planning for MSC in a Shared-Queues Environment” on page 55
- “Planning for RSR in a Shared-Queues Environment” on page 56

## Concepts for Operating in a Shared-Queues Environment

Before operating in a shared-queues environment, read the information in this section which explains how operating in a shared-queues environment differs from operating in a non-shared-queues environment.

### The IMS Role in a Shared-Queues Environment

In a shared-queues environment, IMS acts as a front end for terminals, as a back end for processing, or as both simultaneously.

#### *IMS as a Front End:*

As a front end, an IMS subsystem manages terminal resources and originates communications traffic, such as commands, transactions, and message switches.

## Planning for a Shared-Queues Environment

The message is typically the result of data that is entered at a terminal. A front-end IMS subsystem can also play the role of back end, if database, program, and transaction resources are defined.

### ***IMS as a Back End:***

As a back end, an IMS subsystem manages transactions, database resources, and schedules application programs. A back-end IMS subsystem can also play the role of front end, if terminal resources are defined.

***IMS as both a Front End and a Back End:*** As both a front end and a back end, IMS can process messages that are entered from one IMS subsystem terminal either on that IMS subsystem, or on another IMS subsystem within the Sysplex environment.

Three processing environments exist when IMS acts as a back-end processor: local, remote MSC, and remote shared queues.

### **Definitions:**

- *Local processing* occurs when the IMS dependent region that processes the message is on the same IMS system as the IMS control region that received the message.
- *Remote MSC processing* occurs when the IMS dependent region that processes the message is on a different IMS system than the IMS control region that received the message, and the message is sent to the remote IMS system using MSC.
- *Remote shared-queues processing* occurs when the IMS dependent region that processes the message is on a different IMS system than the IMS control region that received the message. In this case, however, the shared-queues facility contains the message that the remote IMS system processes.

## How IMS Registers Interest in Queues

An IMS system registers and deregisters its interest in queues according to the type of work it can process. Consequently, the CQS for this IMS system can notify the IMS system when work is present on a queue. The types of work include:

- Transactions
- LTERMs
- MSC resources (such as remote LTERMs, remote transactions, and MSNAMEs)

In general, IMS registers its interest as the resources are added (or started), and IMS deregisters its interest as the resources are deleted (or stopped).

### ***Queue Types:***

Various types of queues are managed in a shared-queues environment. Each queue type is used for a different type of work. An IMS system registers interest in only those queue types that it can process, based on the types of work you define for it.

## Planning for a Shared-Queues Environment

The queue types and the work that IMS places on them are shown in Table 2 .

Table 2. Queue Types Maintained in a Shared-Queues Environment

Queue Type	Description
Transaction-ready queue	Contains first qbuffer of messages that are destined for transactions
Transaction-staging queue	An internal queue that the IMS Queue Manager uses for holding those parts of messages that exceed a single qbuffer
Transaction-suspend queue	Contains first qbuffer of messages destined for suspended transactions
Transaction-serial queue	Contains first qbuffer of messages that are destined for serial transactions (transactions defined to IMS on the TRANSACT macro as SERIAL=YES)
LTERM-ready queue	Contains first qbuffer of messages that are destined for LTERMs or MSNAMEs
LTERM-staging queue	An internal queue that the IMS Queue Manager uses for holding those parts of messages that exceed a single qbuffer
APPC-ready queue	Contains first qbuffer of messages that are destined for APPC devices
REMOTE-ready queue	Contains first qbuffer of messages that are destined for remote transactions or remote LTERMs
OTMA-ready queue	Contains first qbuffer of messages that are destined for OTMA devices

### **Registering and Deregistering Interest in Transactions:**

IMS registers or deregisters interest in transaction queues when certain actions are taken:

- IMS registers interest in transaction queues when any of the following events occurs:
  - IMS is initialized.
  - Transactions are added by online change, after the /MODIFY COMMIT command.
  - An operator enters the /START TRAN command.
- IMS deregisters interest in transaction queues when any of the following events occurs:
  - IMS shuts down.
  - Transactions are deleted by online change, after the /MODIFY COMMIT command.
  - A /STOP TRAN command is entered.
- For IMS Fast Path:
  - IMS registers interest in a program queue during IFP region initialization for the name of the program that the region processes.
  - IMS deregisters interest in a program queue when the IFP region is terminated.

### **Registering and Deregistering Interest in LTERMs:**

IMS registers and deregisters interest in LTERM queues when certain actions are taken:

- IMS registers interest in LTERM queues when any of the following events occurs:

## Planning for a Shared-Queues Environment

- A user logs on to a static terminal
- A user signs on to a dynamic terminal
- An LTERM is assigned to an active user or node
- IMS deregisters interest in LTERM queues when any of the following events occurs:
  - A user logs off from a static terminal
  - A user signs off from a dynamic terminal
  - An LTERM is assigned to an inactive user or node

### **Registering Interest in MSC Resources:**

When an MSC logical link is started, IMS registers interest in the MSNAME that is defined with it. All remote transactions that are defined using the same SYSID pair are also registered.

When the logical link is stopped, IMS deregisters interest in the MSNAME and its associated remote transactions.

## Message Flow in a Shared-Queues Environment

In a shared-queues environment, when an IMS system retrieves a message from a shared queue, the message is locked on that shared queue until the IMS system unlocks or deletes the message. If IMS does not unlock or delete a message from the shared queue (for example, because of an IMS abend), the message remains locked. Locked messages are not available to other IMS systems for processing. If you cold start the IMS system after it locks messages on a shared queue, those messages remain locked, and they are moved to the cold queue.

**Definition:** The *cold queue* is a type of queue on which CQS places locked messages that it finds on its private queues after a client cold starts. Messages remain on the cold queue until they are unlocked or deleted.

## Tracking Units of Work

In a non-shared-queues environment, a unit of work is tracked by a recovery token and by a UOW (unit of work). In a shared-queues environment, a unit of work is tracked using only a UOW. The UOW consists of the following fields:

- Originating-system message ID: Message ID assigned by the IMS system that originates the message.
  - Originating IMSID
  - Time-stamp token
- Processing-system message ID: Message ID assigned by the IMS system processing the message.
  - Processing IMSID
  - Time-stamp token

Because the UOW has IDs for both the system that originates the message and the system (if any) that processes the message, all messages that are associated with an original message can be tied together by the UOW (specifically, the originating-system message ID in the UOW).

## Terminal Autologon

Application programs can run on any IMS back-end subsystem; therefore, output can be generated for the same user or terminal on any of these subsystems. In

## Planning for a Shared-Queues Environment

order to deliver output to all waiting users, an autologon node is often switched between IMS systems in the Sysplex environment.

### Message Switches

In a shared-queues environment, messages are queued globally.

If the destination of a message switch is not defined on the local IMS subsystem, the Output Creation exit routine is called to identify the destination. While ETO is active, if the exit routine indicates that the destination is an LTERM, the exit routine creates a dynamic user structure for the LTERM and then enqueues the message.

### Dynamic Control Blocks

To enable a user to enter transactions on IMS systems that do not have the transaction defined, a user can supply information in the DFSINSX0 exit routine to cause IMS to build a dynamic scheduler message block (SMB).

Input and output messages are enqueued to the shared queues, rather than to the control blocks.

### The IMS Queue Manager

The IMS Queue Manager manages the IMS message queues. The Queue Manager data sets, SHMSG and LMSG, are not used to back up messages in a shared-queues environment. However, in order to reduce necessary changes during IMS system definition, the MSGQUEUE macro is still required; the data sets are defined, but they are never allocated or opened.

The Queue Manager uses a number of real storage queue buffers to store the messages prior to placing them in the shared queues on the coupling facility. However, if the message needs to remain in the local IMS system rather than being placed on the shared queues, the message is placed on a local queue. The local queues are also managed by the Queue Manager.

Messages that are placed on the shared queues are recoverable. Because the message queue data sets are not used, a message on a local queue is only recoverable if the log record for the message was written to the IMS log after the oldest of the last two checkpoints were taken.

In an XRF environment, the local queue manager data sets (QBLKSL, SHMSG, and LGMSG) are used. While the XRF alternate subsystem is tracking, local messages are placed in the local message queue data sets (messages for the active subsystem are on the shared queues). When the XRF alternate subsystem takes over the active subsystem, any messages in the local message queue data sets are merged with those on the shared queues.

### Serial Transaction Processing

Messages for serial transactions are only processed by the IMS subsystem that places the messages on the queue. IMS keeps a transaction-serial queue for serial transaction messages in the coupling facility MSGQ structure.

**Related Reading:** For more information on the transaction-serial queue, see “Queue Types” on page 44.

AOI programs must define their transactions as serial (specified as SERIAL=YES on the TRANSACT macro).

## Planning for a Shared-Queues Environment

### Conversational Transactions in a Shared-Queues Environment

In a shared-queues environment, conversation status is local to the IMS system that initiates the conversation. When an IMS system initiates a conversation, the following events occur:

1. IMS places the conversational transaction on the shared queues.
2. A locally defined SMB (or RSMB) or the Output Creation User exit routine identifies the transaction as conversational.
3. Any IMS system that registers interest in the transaction can process it. Also, any IMS system can process any step of the conversation.  
**Exception:** Conversations initiated by APPC or OTMA devices are always processed locally.
4. One IMS system reads the message and schedules an application program to process the message.
5. The application program processes the message, saves data in the SPA, and responds to the initiating terminal.

The /EXIT, /HOLD, and /RELEASE commands apply only to the IMS system that initiates the conversation.

## Planning for Operating in a Shared-Queues Environment

This section describes the planning tasks that are associated with operating in a shared-queues environment.

### Configuring a Shared-Queues Environment

Figure 8 on page 49 and Figure 9 on page 50 illustrate two methods of configuring a shared-queues environment.

#### **Cloned Configuration:**

A *cloned configuration* involves creating identical IMS systems, with almost identical system definitions. Some differences in the definitions are required, such as MSC links. You can override the MTO definitions generated during system definition in PROCLIB member DFSDCxxx.

**Related Reading:** For more information on overriding the MTO definitions, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The LTERMs, databases, and transactions for all the IMS systems are defined identically. In this configuration, any IMS system is able to process a particular transaction. Cloned configurations are useful for automatic workload balancing.

**Recommendation:** To take full advantage of a cloned configuration, ensure that the databases are shared across the Sysplex.



## Planning for a Shared-Queues Environment

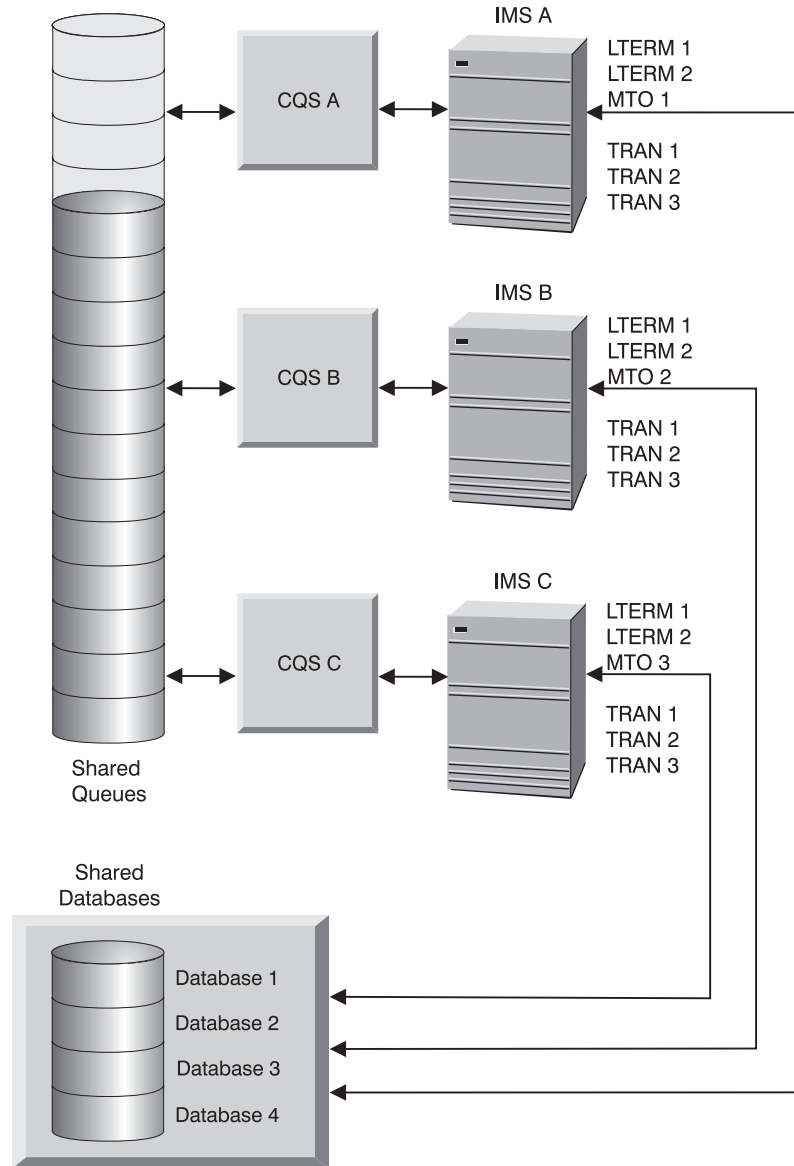


Figure 8. Cloned Configuration in a Shared-Queues Environment

### **Partitioned Configuration:**

A *partitioned configuration* separates the VTAM network responsibilities from the database work. Each resource is defined to only one IMS system.

**Example:** You can define an ES/9000 9021 to manage the VTAM network as an IMS front end, while multiple smaller 9672 CMOS processors manage the database work.

Partitioned configurations are useful as replacements for, or additions to, MSC networks.

## Planning for a Shared-Queues Environment

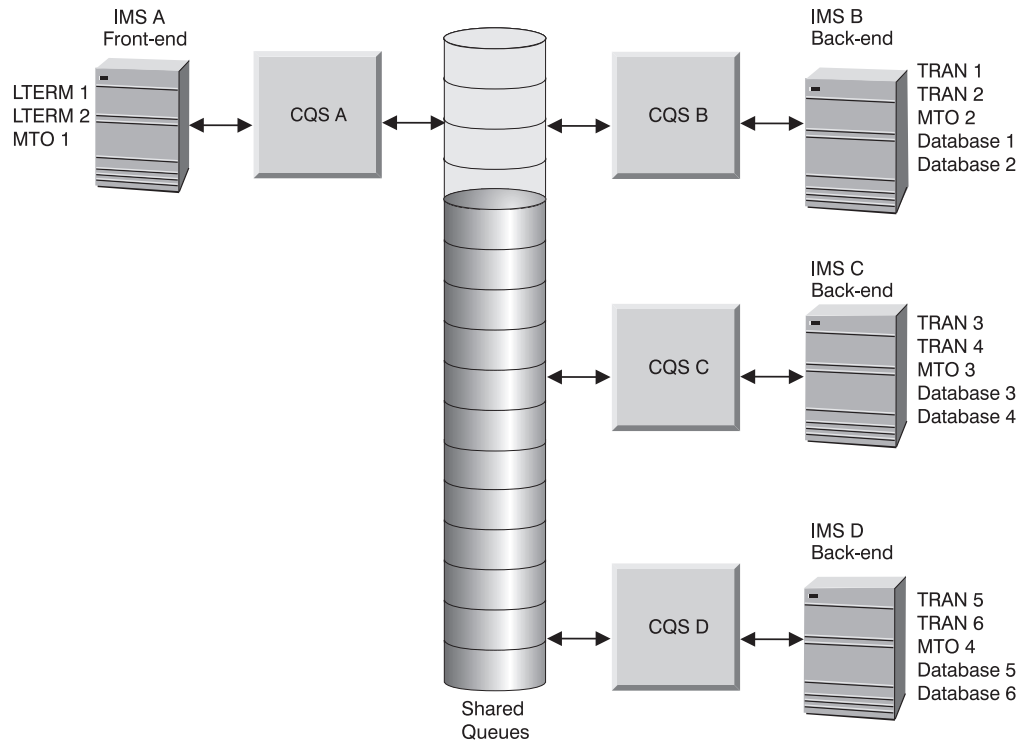


Figure 9. Partitioned Configuration in a Shared-Queues Environment

In the partitioned configuration in Figure 9, IMS A acts as the front end, and has LTERM 1, LTERM 2, and MTO 1 defined. IMS B, IMS C, and IMS D act as back ends, each with separate transactions and databases defined. Transactions in this configuration can run only on the IMS systems on which they are defined.

### Planning to Define List Structures

CQS manages message queues that are shared within a Sysplex environment by using coupling facility list structures. A primary list structure holds the message queues. An overflow list structure, if you define one, holds additional queues after the primary list structure reaches a predefined threshold.

Define the following list structures:

- For IMS systems that are not using Fast Path, define only MSGQ primary structures and optional MSGQ overflow structures.
- For IMS systems that use Fast Path, you must define both MSGQ primary (and optional overflow) structures and EMHQ primary (and optional overflow) structures.

Figure 10 on page 51 and Figure 11 on page 52 show shared queues for IMS clients A and B:

In Figure 10 on page 51, there is only the use of MSGQ structures, MSGQ structure recovery data sets (SRDSs), MSGQ checkpoint data sets, and a MSGQ MVS log stream.

## Planning for a Shared-Queues Environment

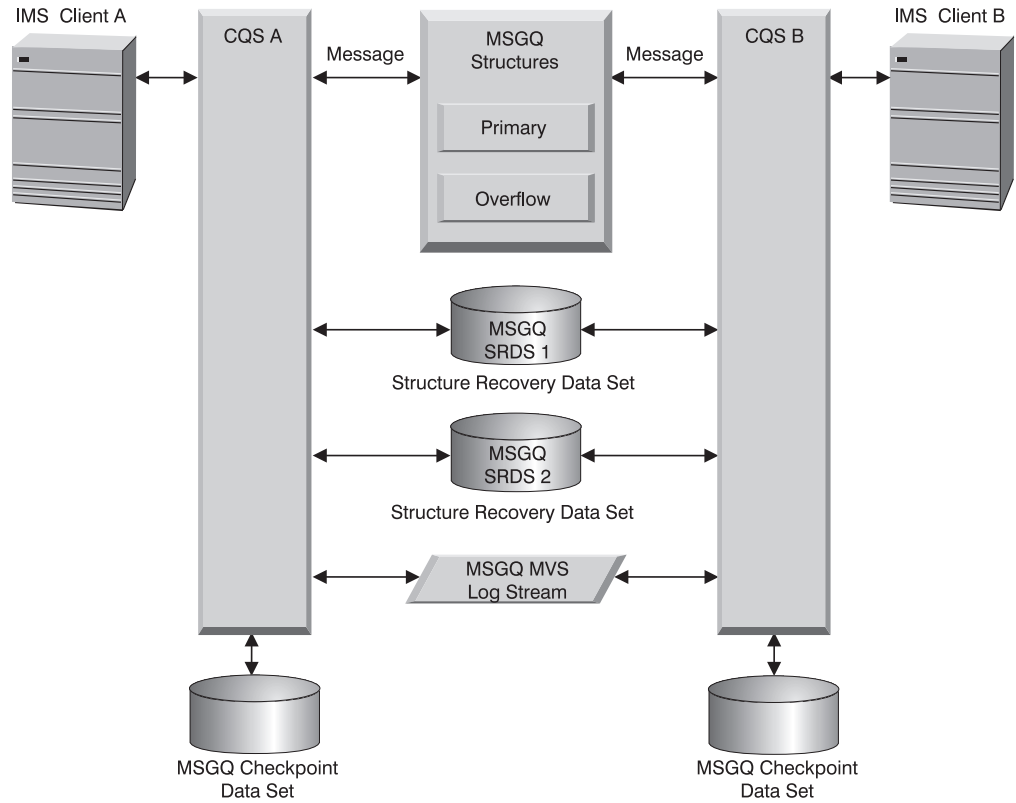


Figure 10. Shared-Queues Environment with its MSGQ List Structures

In Figure 11 on page 52, there is only the use of MSGQ (not shown) and EMHQ structures, along with their corresponding structure recovery data sets (SRDSs), checkpoint data sets, and MVS log stream.

## Planning for a Shared-Queues Environment

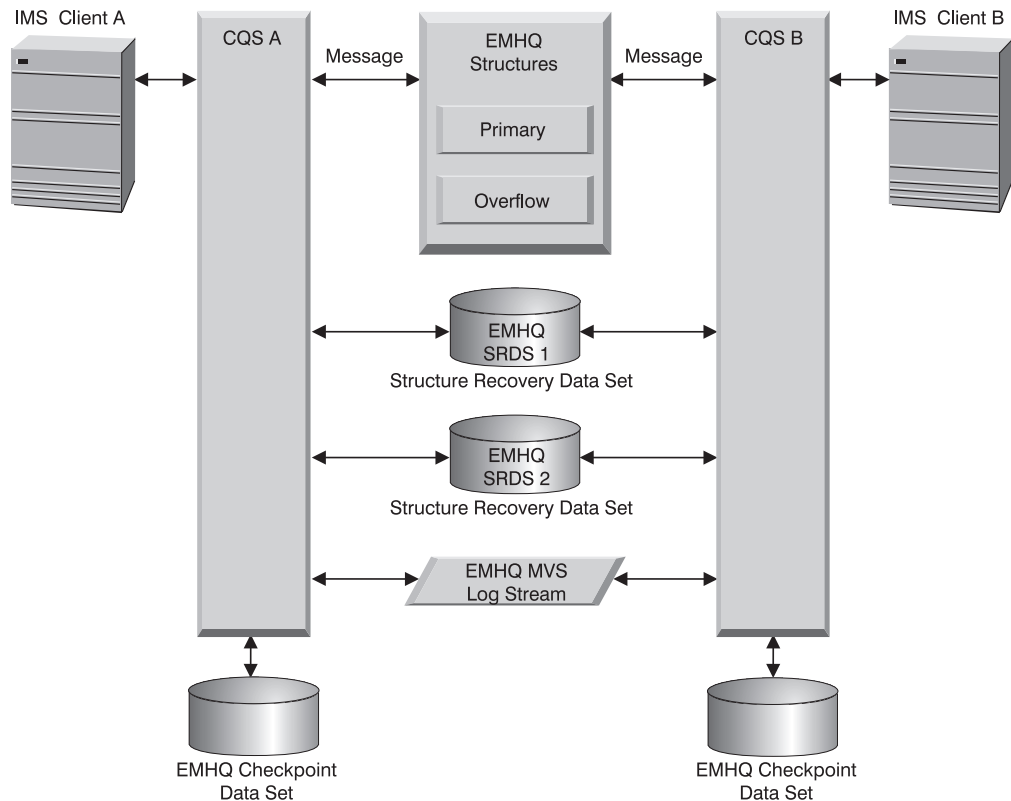


Figure 11. Shared-Queues Environment with its EMHQ List Structures

**Related Reading:** For information on how to define:

- MSGQ list structures, see “Defining Message Queue List Structures” on page 82.
- EMHQ list structures, see “Defining Fast Path Message Queue List Structures” on page 82.

### Using Associated Printers

In a shared-queues environment, when a user signs on and specifies associated printer information, the IMS front end registers interest for each associated LTERM. When the application inserts a message to one of these LTERMs, the IMS front end is notified and begins autologon for the printer.

**Recommendation:** Do not provide autologon parameters for these users in the IMS back end; doing so causes both the front end and back end to process autologon for the associated printers.

For shared printers in a shared-queues environment, shared printer terminals are often switched between IMS subsystems in the Sysplex environment. This enables the output to be sent to any IMS subsystem in the Sysplex environment and to be printed.

### Planning for IMS Front-End Switch

Using IMS Front-end Switch (FES) in a shared-queues environment requires either:

- Coding the FES exit routine (DFSFEJ0) to route the input messages to the correct LTERM (to deliver FES reply messages correctly).
- Establishing ISC sessions between IMS systems within the Sysplex environment (to ensure that FES input and reply messages are sent and received by any IMS).

### **Related Reading:**

- For information on the FES exit routine, see *IMS/ESA Customization Guide*.
- For information on establishing ISC sessions, see “Controlling the Session (Session Control Protocols)” on page 267.

### **Determining Eligibility for Sysplex-Wide Processing**

**Definition:** An *IMS message* can be one of the following statements:

- A transaction or transaction response
- A message switch
- A system message
- A message from LU 6.2 or OTMA
- A command or command response

Some IMS messages in a shared-queues environment are eligible for Sysplex-wide processing; others are not.

All messages (not including commands or command responses) are placed in shared queues.

***Messages Eligible for Sysplex-Wide Processing:*** Messages for the following types of communication are eligible to be processed on any IMS system in the Sysplex environment:

- Transactions defined locally or using the Output Creation exit routine (DFSINSX0)
- Fast Path transactions defined using the DFSHAGU0 exit routine with Sysplex processing option LOCAL FIRST or GLOBAL ONLY
- Fast Path messages for a program that is active in the Sysplex environment
- LTERMs or by using the Output Creation exit routine (DFSINSX0)

**Definition:** A message for an LTERM can be:

- A transaction response
- A message switch
- A system message
- A command response

***Messages Not Eligible for Sysplex-Wide Processing:*** Messages for the following types of communication must be processed locally and are therefore not eligible for Sysplex-wide processing:

- Serial transactions (transactions defined with SERIAL=YES specified on the TRANSACT macro)
- APPC LU 6.2 terminals
- OTMA clients
- Commands and command responses

***APPC and OTMA Messages in a Sysplex Environment:*** Because APPC and OTMA messages cannot be moved from one MVS image to another, APPC and OTMA messages must be processed by the IMS system that received them.

**Recommendation:** To help balance workload, spread your LU 6.2 workload across the Sysplex by using APPC generic resources.

## Planning for a Shared-Queues Environment

**Related Reading:** For more information on APPC and LU 6.2 processing, see “Chapter 18. Administering APPC/IMS and LU 6-2 Devices” on page 367.

For more information on APPC generic resources, see “Specifying APPC Generic Resource Names” on page 58.

For more information on OTMA messages, see *IMS/ESA Open Transaction Manager Access Guide*.

### Managing the IMS Dead-Letter Queue

IMS maintains a dead-letter queue only for local user structures. For shared queues on the coupling facility, use the `/DIS QCNT MSGAGE` command to display potential dead-letter queues. By examining the display output, you can determine whether a particular queue is a dead-letter queue and then take appropriate action.

**Related Reading:** For more information on the `/DIS QCNT MSGAGE` command, see *IMS/ESA Operator's Reference*.

### Using the Expedited Message Handler

The IMS Fast Path Expedited Message Handler (EMH) also uses the shared queues in a shared-queues environment. IMS calls the Fast Path Input Edit/Routing exit routine to determine if an incoming message should be processed by Fast Path. If Fast Path must process the message, either a local subsystem processes it, or IMS places it on the shared queues.

Fast Path uses EMHQ structures, which are defined in the same way as the MSGQ structures (in the DFSSQxxx member of PROCLIB).

If an IMS subsystem joins the XCF group, abends, or shuts down, it notifies all sharing IMS subsystems by sending a message with the program name table. Sending this message makes all IMS subsystems aware of which IMS subsystems are servicing particular programs. Likewise, when a program is started or stopped on an IMS subsystem, it notifies all sharing IMS subsystems, and each IMS subsystem updates its program name table.

If no active region is available in the Sysplex to service a program, all IMS subsystems that have enqueued messages for that program issue message DFS2529I to the inputting terminals. These terminals are then unlocked. Any new input for the program is rejected with message DFS2533I.

Because Fast Path input messages are not recoverable, they are not written to the SRDS data set during CQS checkpoint. You can recover EMHQ structures; however, no Fast Path input messages are retained on the SRDS data set, and only Fast Path output messages can be recovered from the EMH queues. For each terminal awaiting output from an input message that has been lost during EMHQ structure rebuild, IMS sends message DFS2766I and unlocks the terminal. In addition, IMS writes a X'5936' log record for the lost message.

The EMHQ structures can be copied, and all messages are copied from the old structure to the new one, regardless of whether they are recoverable.

For EMHQ rebuild, it is possible to specify that the rebuilt EMHQ structure be accessible while the rebuild is in progress. Use the `WAITRBLD=` parameter in the DFSSQxxx PROCLIB member.

## Planning for a Shared-Queues Environment

### Requeuing Suspended Transactions

When a transaction is suspended, it is placed on the transaction-suspend queue in the coupling facility. To make such messages available for processing, issue the /DEQUEUE SUSPEND TRAN command. Issuing this command moves the transaction to the transaction-ready queue or the transaction-serial queue.

The IMS system that places the transaction on the transaction-suspend queue can be different from the IMS system that issues the /DEQUEUE SUSPEND TRAN command. In this case, other IMS systems that share the MSGQ structure need to issue the /DEQUEUE SUSPEND TRAN command.

**Related Reading:** For more information on the transaction-suspend queue, the transaction-ready queue, or the transaction-serial queue, see “Queue Types” on page 44.

### Scheduling AOI Transactions

To ensure that the transaction runs locally, define all AOI transactions that are scheduled by the AOI exit routine as serial. Placing the AOI transaction on the shared queue allows any IMS system to process it, regardless of whether the command applies to that IMS system or not.

## Planning for MSC in a Shared-Queues Environment

IMS MSC subsystems can be part of a shared-queues environment, connected to IMS subsystems outside the shared-queues environment. Each IMS subsystem within the Sysplex environment can act as any of the following:

- A front end, receiving messages from terminals or across MSC links
- A back end, processing messages received from a front end
- An intermediate subsystem, receiving messages and passing them on to other IMS systems across MSC links

Although two IMS subsystems can have MSC links defined, messages are placed on the shared queues from one of these IMS subsystems and picked up for processing by another IMS subsystem within the Sysplex—these messages are not sent over the MSC links.

**Exception:** If one of these IMS subsystems is running without shared queues enabled, the MSC links are used. <sup>7</sup>

To plan for using MSC in a shared-queues environment, take each of the following actions:

- Specify all remote IMS subsystems on the MSNAME macro (SYSID and NAME), and assign a unique SYSID within the local IMS subsystem. These remote MSNAMEs remain stopped and are used only to route messages from any IMS subsystem within the Sysplex to a remote IMS subsystem.
- Define MSC remote system identifiers (SYSIDs), using the MSNAME macro, for each IMS system within the Sysplex.
- Within the local IMS subsystem, make all SYSIDs be unique across all IMS subsystems in the Sysplex and in the MSC network.
- Define remote transactions and LTERMs to the IMS subsystem that has the MSC link (the back-end subsystem). Also define them to the IMS front-end subsystem. If you do not define them, IMS calls the User Output Creation exit routine

---

7. This situation might occur when migrating to a shared-queues environment.

## Planning for a Shared-Queues Environment

(DFSINSX0) to determine whether the destination is local or remote, and to determine whether it is a transaction or an LTERM.

When an MSC link exists on the front-end IMS subsystem, you can use MSC for APPC and OTMA transactions in a shared-queues environment. The conversation is maintained with the local IMS subsystem but is not carried to the remote IMS subsystem. Therefore, the remote application program cannot issue the SET0 DL/I call, or the CPI-C verbs SEND\_ERROR and DEALLOCATE\_ABEND. Also, if the remote IMS subsystem allocates another APPC conversation, it receives the default user ID that is associated with the MPP region, and it does not use the original APPC user ID in order to verify security.

### Planning for RSR in a Shared-Queues Environment

Remote Site Recovery (RSR) allows you to quickly recover computer services in the event of an interruption. In a non-shared-queues environment, RSR recovers full-function databases, Fast Path DEDBs, IMS message queues, and the TM network.

RSR is supported in a shared-queues environment.

**Recommendation:** All IMS subsystems that share a structure should:

- Be part of the same RSR service group
- Send log data to the same IMS tracking subsystem

If an RSR takeover occurs, and the active subsystems are restarted at the remote site, the new active subsystems must be started with the shared-queues option.

**Restrictions:** For RSR in a shared-queues environment:

- You cannot recover the shared message queues or the TM network after a remote takeover, because message queues are not logged in the IMS log. After takeover, cold start the DC system using the /ERESTART command.
- The tracking subsystem cannot track messages in real time, because it cannot run with the shared-queues option. To handle local message activity, the queue manager message queue data sets are used.

**Related Reading:** For more information on RSR in a shared-queues environment, see *IMS/ESA Administration Guide: System*.

---

## Planning for Generic Resource Groups

This section describes the planning and administration tasks associated with using generic resource groups.

- “Requirements for Using Generic Resource Groups” on page 57
- “Restrictions on Using Generic Resource Groups” on page 57
- “Managing Generic Resource Affinities Using IMS” on page 57
- “Managing Generic Resource Affinities Using VTAM” on page 58
- “Specifying Generic Resource Names” on page 58
- “Specifying APPC Generic Resource Names” on page 58
- “Initiating Sessions Using GRSNAMEs or APPLID names” on page 58
- “Changing the Logon Procedure” on page 58
- “Overriding the APPLID Field” on page 59
- “Determining When Affinities Are Terminated” on page 59



## Planning for Generic Resource Groups

- “Terminating Affinities That Persist” on page 59
- “Dropping an IMS System from a Generic Resource Group” on page 60
- “Resetting Terminal Status” on page 60
- “Logging On after IMS Failure with Affinity” on page 60
- “Bypassing Affinity Management During the IMS ESTAE Process” on page 62
- “Controlling Generic Resource Member Selection” on page 60
- “Ensuring Consistency across IMS Systems” on page 61

## Requirements for Using Generic Resource Groups

To participate in session balancing using generic resource groups, ensure your installation does each of the following steps:

- Connects your VTAM APPN node to an MVS coupling facility structure named ISTGENERIC. Make sure that the ISTGENERIC structure is defined in the active CFRM policy for the Sysplex.
- Operates the members of the generic resource group in a Sysplex environment.
- Operates the minimum release of VTAM and OS/390 that are required for non-APPC generic resources and APPC/IMS generic resources.
- Identifies the IMS systems that belong to the generic resource group (by specifying the same IMS generic resource name in the GRSNAME execution parameter of each IMS system). You can also use the /START VGRS command. You can optionally 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 systems 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 defining the ISTGENERIC coupling facility structure and determining the storage size, see *VTAM Network Implementation Guide Version 4, Release 2*.
- For information on the /START VGRS command, see *IMS/ESA Operator's Reference*.

## Restrictions on Using Generic Resource Groups

When creating a generic resource group, system programmers should be aware of the following restrictions:

- The target of an MSC link cannot be a generic resource name.
- IMS XRF systems cannot participate as members of a generic resource group; however, they can be members of the same shared-queues group.

## Managing Generic Resource Affinities Using IMS

To manage generic resource affinities using IMS, set the GRAFFIN=IMS|VTAM execution option of the DFSDCxxx proclib member to GRAFFIN=IMS. GRAFFIN=IMS is the default setting. If you enter the keyword or parameter for this option incorrectly, IMS uses the default and does not generate an error message.

## Planning for Generic Resource Groups

**Note:** Unless indicated otherwise, information included in "Planning for Generic Resource Groups" describes management of generic resource affinities using IMS.

## Managing Generic Resource Affinities Using VTAM

To manage generic resource affinities using VTAM, set the GRAFFIN=IMS|VTAM execution option of the DFSDCxxx proclib member to GRAFFIN=VTAM.

When IMS is included in a VTAM generic resource group, VTAM resets generic resource affinities whenever a session outage occurs. Session outages include CEC, MVS, VTAM, and IMS failures. When the affinity is reset, a terminal session can be reestablished immediately with another member of the generic resource group.

**Note:** VTAM does not manage LU6.1 (ISC) affinity. LU6.1 (ISC) affinity will be processed as usual by IMS.

## Specifying Generic Resource Names

To create a generic resource group, specify the same generic resource name for all the IMS systems in a generic resource group. Specify this name on the GRSNAME execution parameter of the MVS START command (when starting the IMS control region).

### Restrictions:

- The generic resource name specified on the GRSNAME parameter must not match the name of an XRF USERVAR.
- A particular IMS system cannot be a member of more than one generic resource group.

## Specifying APPC Generic Resource Names

Specifying an APPC generic resource name enables LU 6.2 devices to participate in a generic resource group and session balancing. Specify the APPC generic resource name on the GRNAME parameter of the LUADD statement in the APPC/MVS APPCPMxx member.

**Restriction:** The APPC generic resource name must be different from the name you specify on the GRSNAME execution parameter.

## Initiating Sessions Using GRSNAMEs or APPLID names

You can initiate a session in either of two ways:

- You can initiate a session with any IMS system in a generic resource group by specifying the GRSNAME execution parameter.
- You can initiate a session with a specific IMS system by specifying the APPLID name of the system.

## Changing the Logon Procedure

You can provide the option of logging on using either a generic resource name or an IMS APPLID name; therefore, plan to make one or both of these names available in the logon procedure.

## Overriding the APPLID Field

In order to keep the IMS system definitions as similar as possible, you can specify the same name in the APPLID field in COMM macros for all the IMS systems in a generic resource group. You must then override this name by specifying a unique APPLID name on the APPLID1 execution parameter.

Similarly, you can override the PASSWD field in the COMM macro by using the PASSWD= execution parameter.

**Related Reading:** For more information on the APPLID1 and PASSWD execution parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

## Determining When Affinities Are Terminated

When you terminate a session by logging off, IMS terminates the affinity between your node and the IMS generic resource member. However, in some situations, affinities persist. You can use several methods in order to terminate these affinities.

To display the list of IMS systems and their affinities with specific nodes, use the /DISPLAY AFFINITY command.

**Related Reading:** For information on the /DISPLAY AFFINITY command, see *IMS/ESA Operator's Reference*.

For information on how to terminate affinities that persist, see "Terminating Affinities That Persist".

## Terminating Affinities That Persist

A terminal cannot participate in session balancing if it has a persisting affinity with a particular IMS system from a previous session. An affinity persists after an IMS failure or logoff in each of the following situations:

- The terminal status at logoff or IMS failure was one of the following modes:
  - Conversation mode
  - Exclusive mode
  - MFS-test mode
  - Preset-destination mode
  - Response mode
  - Test mode
- The terminal is a SLU P or 3600 Finance terminal. Use a /CHECKPOINT command with the LEAVEGR keyword, or cold start the IMS TM component in order to terminate the affinity.
- The terminal is an ISC single-session terminal that has not quiesced.
- The VTAM node is an ISC parallel session node, and one or more of the parallel sessions has not quiesced. Use a /CHECKPOINT command with the LEAVEGR keyword in order to terminate all ISC parallel session affinities.

You can terminate an affinity using any of the following methods:

- Enter the /CHECKPOINT command with the LEAVEGR keyword.
- Use the Logoff or Signoff exit routine to reset terminal status.
- Cold start the IMS TM subsystem, and specify the GRSNAME when restarting IMS.

## Planning for Generic Resource Groups

For ETO, VTAM maintains affinities at the LU or IMS node level, and all affinities are deleted when the session is terminated.

### **Related Reading:**

- For information on the /CHECKPOINT command and the LEAVEGR keyword, see *IMS/ESA Operator's Reference*.
- For information on the Logoff and Signoff exit routines, see *IMS/ESA Customization Guide*.

## Dropping an IMS System from a Generic Resource Group

You can use the /STOP VGRS command to drop an IMS system out of a generic resource group. Similarly, you can use the /START VGRS command to bring that IMS system back into the generic resource group.

**Related Reading:** For more information on the /STOP VGRS and /START VGRS commands, see *IMS/ESA Operator's Reference*.

## Resetting Terminal Status

You can use the Logoff (DFSLGFX0) exit routine to reset a terminal's status during a logoff. Similarly, you can use the Signoff (DFSSGFX0) exit routine to reset an ETO user's status during a signoff. Resetting terminal or user status enables VTAM to terminate an affinity with the terminal, enabling the terminal to log on again and participate in session balancing.

**Related Reading:** For more information on the Logoff and Signoff exit routines, see *IMS/ESA Customization Guide*.

## Logging On after IMS Failure with Affinity

Suppose you are operating in a shared-queues environment, and your terminal is logged onto an IMS system in a generic resource group. If that IMS system fails, the affinity between that IMS system and your terminal might persist. Consequently, you can have output messages waiting for you on that terminal. To obtain your messages, you can log onto another IMS system within the generic resource group, provided that both of the following conditions exist:

- You log on using the APPLID name of another IMS system within the generic resource group.
- Your output messages are not locked on the failed IMS system, and those messages are not responses to either an IMS conversation or a response-mode transaction.

## Controlling Generic Resource Member Selection

You can control VTAM's selection of a generic resource member for a terminal when the terminal logs on. To control the selection process, use either the VTAM Generic Resource Resolution exit routine (ISTEXCGR) or the MVS Workload Manager. Choosing between these facilities depends on the needs of your application program and on your installation requirements.

The criteria that VTAM uses to select a generic resource member are:

1. Existing affinity <sup>8</sup>
2. VTAM Generic Resource Resolution exit routine (ISTEXCGR)
3. MVS Workload Manager
4. Current session counts

### **Related Reading:**

- For more information on the VTAM Generic Resource Resolution exit routine (ISTEXCGR), see *VTAM Customization*.
- For more information on the MVS Workload Manager, see *MVS/ESA Planning: Workload Management* and *MVS/ESA Programming: Workload Management Services*.

## Ensuring Consistency across IMS Systems

Inconsistencies in definitions, operator commands and procedures, and exit routines can create problems within a generic resource group:

- At a minimum, they can confuse terminal users, who are unaware of being in session with different IMS systems from one logon to another logon.
- At worst, they can cause processing disruptions.

**Recommendations:** To ensure consistency across a generic resource group:

- Use equivalent specifications when defining the member IMS systems.  
**Example:** If one IMS system specifies that ETO be included (ETOFEAT=YES on the IMSCTRL macro), all other IMS systems in the generic resource group should specify that ETO be included.
- Specify execution parameters consistently across generic resource members.  
**Example:** IMS systems in a generic resource group should each specify that ETO be enabled or not be enabled.
- If ETO is included, specify the same information on the ETO descriptors for each IMS generic resource group member.
- Use the same naming conventions for LTERMs and transactions across a generic resource group. If the same LTERM name is assigned to different nodes on different IMS systems, unpredictable results might occur.
- Make all terminal definitions consistent across generic resource members.  
**Example:** LTERM names should be consistently defined, during IMS system definition, to the same physical terminals on all the IMS systems in the generic resource group.
- Ensure that master terminal operators (MTOs) on each IMS system in the generic resource group have procedures for notifying the other MTOs about the commands they issue.  
**Example:** If one MTO issues a /STOP command to stop a user on one IMS system in a generic resource group, that user is stopped only on the IMS system on which /STOP is entered.
- Make exit routines functionally equivalent across a generic resource group—especially the Output Creation exit routine, the OTMA Routing exit routines, the Logon and Signon exit routines, and the Logoff and Signoff exit routines.

---

8. VTAM maintains an affinity table in a list structure called ISTGENERIC on the coupling facility.

## Planning for Generic Resource Groups

### Bypassing Affinity Management During the IMS ESTAE Process

If you use IMS to manage generic resource affinities, you can control whether or not IMS uses or bypasses the VTAM generic resource logic in the IMS ESTAE exit. To use the IMS ESTAE process, indicate `GRESTAE=Y` in the `DFSDCxxx` proclib member. `GRESTAE=Y` indicates that IMS should follow the existing ESTAE logic to delete affinity for all nodes where no status remains before closing the ACF/VTAM ACB. `GRESTAE=N` indicates that IMS should close the ACF/VTAM ACB immediately to expedite IMS termination, and leave affinity for all nodes set.

IMS VTAM generic resource logic in the IMS ESTAE exit attempts to delete generic resource affinity if no terminal status remains. This action requires a serial and synchronous VTAM `CLSDST` loop for all active terminals in IMS. Under normal conditions, the amount of time that this action requires is not significant. However, each `CLSDST` issued by the IMS ESTAE can incur the maximum defined VTAM I/O timeout time, leading to a significant delay in the final termination and subsequent restart of IMS.

Table 3 summarizes how generic resource affinities will be managed using the `GRAFFIN` and `GRESTAE` options indicated in the `DFSDCxxx` proclib member.

*Table 3. Generic Resource Affinities Management Using GRAFFIN and GRESTAE Options*

<b>GRAFFIN Option</b>	<b>GRESTAE Option</b>	<b>Generic Resource Affinities Management</b>
IMS	Y	IMS manages generic resource affinities, including during ESTAE processing. IMS resets affinity during the ESTAE process that occurs during IMS failure.
IMS	N	IMS manages generic resource affinities except during ESTAE processing. IMS does not reset affinity during the ESTAE process that occurs during IMS failure.
VTAM	Y or N	VTAM manages non-ISC generic resource affinities. IMS does not reset ISC affinity because (at the very least) the failing (parallel) session will not be properly quiesced.

### Terminals Used with Fast Path

To have a Fast Path-capable system, specify Fast Path support on the `FPCTRL` macro. IMS systems with a very high transaction rate use Fast Path's expedited message handler (EMH) facility. EMH is a performance option that expedites message processing by imposing restrictions on message lengths and segmentation. LU 6.2 terminals can use EMH; no definitions or specifications are required.

With Fast Path, an EMH buffer is acquired from the EMH buffer pool when the first eligible input message is received for a Fast Path transaction. The buffer remains allocated to the terminal for future use until the session terminates or the user signs off. If an EMH buffer is allocated to the terminal, it is reused if it meets the requirements of the next input call. If the message requires a larger buffer, Fast Path swaps it for a larger one.

## Terminals Used with Fast Path

If, at entry to IMS, an input message satisfies all of the following criteria, it is edited in a Fast Path EMH buffer instead of in a full-function message queue buffer.

- The terminal is not FES capable.
- The terminal is not the MTO.
- The input is single segment.
- The input is non-MFS edited.
- The input is not an IMS command.
- Execution of at least one Fast Path transaction is scheduled using Fast Path.

**Exception:** If the message is not a Fast Path transaction, the message is moved to a full-function message queue buffer.

When a message with all of the above characteristics is received, an EMH buffer is allocated to the terminal. The EMH buffer remains allocated to the terminal until either the user signs off or the session terminates. The buffer is used repeatedly for Fast Path transactions until a larger EMH buffer is required for special applications. When a larger buffer is needed, the current EMH buffer is swapped for the larger-size EMH buffer. The smaller EMH buffer is released and returned to the EMH pool.

The EMHL parameter in the IMS control region initialization (also known as the startup parameter) specifies the default EMH buffer size for all Fast Path transactions for all Fast Path-eligible terminals. A Fast Path transaction can specify an EMH buffer that is larger than the default. Specify larger application EMH buffer sizes on the APPLCTN or TRANSACT stage\_1 macros by using the `FPATH=size` parameter when you generate the system. If the EMH buffer pool is exhausted, message DFS3971 is sent to the input terminal.

Buffers are extracted from the EMH buffer pool, which expands and contracts dynamically. The size of the pool depends on the number of terminals that are concurrently entering Fast Path transactions and the buffer sizes that are required to satisfy each request.

You can say `FPATH=No` or `Yes` on the APPLCTN and TRANSACT macros, or you can specify `FPATH=size`, where *size* is the EMH buffer size that is required to run the transaction. `FPATH=size` implies `FPATH=Yes`. The minimum EMH buffer size is 12 bytes, and the maximum is 30720 bytes.

MSDBs are available to ETO terminals, unless they have terminal-related keys.

The `OPTIONS` keyword on the `TERMINAL` macro or ETO descriptor is used to:

- Declare `FORCRESP` or `TRANRESP` so that Fast Path-eligible terminals operate in response mode
- Specify `PAGDEL` for automatic page deletion where appropriate

You can use non-VTAM terminals to enter Fast Path transactions. For non-VTAM terminals, the `LINE` macro must indicate response mode by using the `RESP=TERM` parameter.

**Related Reading:** For information on the communication devices that are supported by Fast Path, see *IMS/ESA Release Planning Guide*.

### Extended Recovery Facility (XRF) and IMS

**Definition:** An XRF complex enables you to rapidly resume end-user services when a failure occurs.

When you do not use XRF, a high level of availability with short interruptions in end-user service can be difficult to achieve when IMS fails. Termination processing must complete, and the cause of the failure determined before restart can take place. A restart can be time-consuming if IMS regions must be restarted and terminal sessions reestablished.

An XRF configuration consists of a primary system, called the *active IMS system*, and a secondary system, called the *alternate IMS system*. The alternate IMS tracks the activities of the active IMS in the same or in a different central processing complex (CPC). The active IMS processes the IMS workload in the same way that it would without XRF. It does not do any additional processing or logging. Using the log, the active IMS informs the alternate IMS of its activities. Surveillance mechanisms alert the alternate IMS to problems in the active IMS. The active IMS is not aware of the activities of the alternate IMS. The alternate IMS does not process any transactions; it maintains the status of the active IMS, and monitors it for any indications of trouble. The alternate IMS tracks resource changes in the active IMS, and updates its control blocks and buffers while continuously changing its status of readiness. In this manner, work can be quickly shifted from the active IMS to the alternate IMS without interruption. See Figure 12 for an example of a simple XRF complex.

You can use XRF to switch IMS to a different system for scheduled maintenance or during off-peak periods.

**Related Reading:** For more information on XRF, see *IMS/ESA Administration Guide: System*.

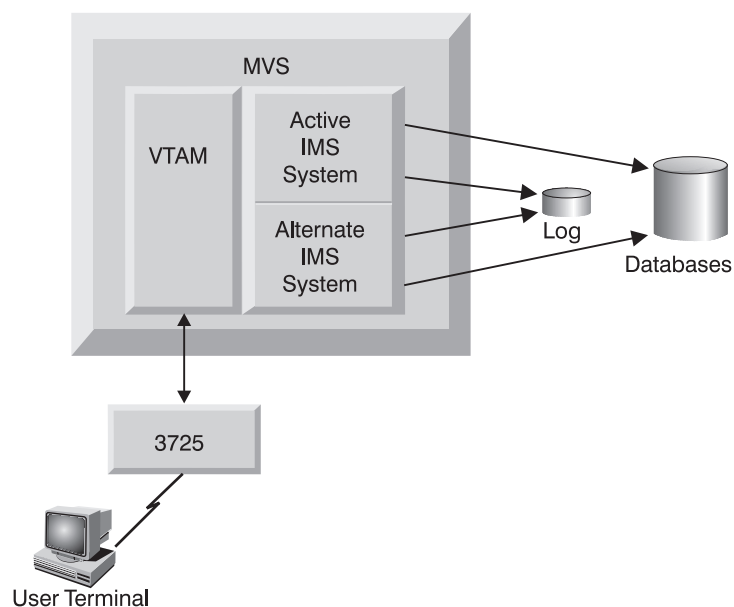


Figure 12. One XRF Complex in One Processor



## Terminals in an XRF Complex

XRF supports all terminals that your IMS users are currently using. XRF provides emergency support with a minimum of effort from you. XRF requires that you specify keywords on several system definition macros or ETO descriptors. Unless you want to change the switching priority of a terminal from the defaults, or to change the terminal type sessions, you need not change any terminal definition macros.

XRF automatically gives the terminals at your installation the best service that the terminal's characteristics allow. This section describes these characteristics.

The three terminal types and the service XRF provides them are:

<b>Terminal Grouping</b>	<b>XRF Support</b>
<b>Class-1</b>	For each active IMS terminal session that is opened, a backup session is opened on the alternate IMS. At takeover, IMS requests a VTAM SWITCH of the terminals from the active IMS to the backup session. The sessions are switched without losing the session.
<b>Class-2</b>	At takeover, the sessions end. The alternate IMS automatically tries to reestablish the sessions. The users are briefly without service.
<b>Class-3</b>	At takeover, the session ends. The operator or the end users can establish new sessions with the new active IMS if the terminals are physically connected to it.

### Class-1 Terminals

To take full advantage of XRF, most of the terminals in your complex should be Class-1 terminals. Class-1 terminals are those terminals defined on the UNITYPE keyword of the TYPE macro, as one of the following:

- SLUTYPE1
- SLUTYPE2
- SLUTYPE4
- SLUTYPEP
- FINANCE

Class-1 terminals have backup sessions on the alternate IMS. These sessions are established during the tracking phase, when a session on the active IMS is initiated. They are switched automatically after a takeover without losing the session.

From the terminal user's point of view, only one session with IMS exists. From the NCP point of view, two host sessions exist. The actual messages are only passed between the node and the active IMS. The alternate IMS tracks the status of the message activity by monitoring the IMS log.

When the active IMS establishes or terminates a terminal session with class-1 terminals, the alternate IMS establishes or terminates its backup session with the terminal. When the active IMS fails, the alternate IMS takes over the terminal session (without losing control from the viewpoint of the node) by changing the

## Extended Recovery Facility (XRF) and IMS

mode of the established backup session from BACKUP to ACTIVE. When the session is switched, the NCP sends the new active IMS its view of the terminal's status. IMS compares this with its own status record to decide what, if any, recovery action to take. When the REVERIFY operand is used with RACF, the user must sign on again.

The terminal types listed in Table 4 on page 67 are eligible for class-1 service if they are both:

- Owned by an XRF-capable VTAM
- Connected to an XRF-capable NCP in a 37x5

## Class-2 Terminals

Class-2 terminals do not have backup-session support on the alternate IMS, but they are restarted automatically by IMS after takeover. The interface to the terminal, including recovery procedures after session reestablishment on the alternate IMS, is exactly the same as in prior versions of IMS, except for the automatic session reestablishment at system takeover.

When a Class-2 terminal session is established on the active IMS, the alternate IMS tracks the session initiation and termination by using log records. When the active IMS terminates abnormally, the alternate IMS tries to establish a new session with the network resources that were active before the failure.

When IMS tries to reestablish a session with a terminal that has no available network path, the attempt fails. Before IMS can reestablish a session on a locally attached terminal, the operator must switch the terminals to the new active IMS. If the complex has many of these kinds of terminals, you might consider giving the operator control before the takeover actually proceeds. At this time, the operator could perform the switch, allowing a faster recovery for sessions on these terminals. To give the operator this control, use the AUTO parameter in member DFSHSBxx in IMS.PROCLIB.

See Table 4 on page 67 for terminals that qualify as class-2.

If you have an application terminal that communicates with a programmable controller in an XRF complex, you might need to add code to the communications portion in order to achieve the highest level of transparency available. Additionally, for the ISC terminal, you must define a session for the link that connects the ISC terminal to an XRF IMS.

**Related Reading:** For information on the additional code required, see “Chapter 10. Overview of Intersystem Communication” on page 221 and *IMS/ESA Customization Guide*.

## Class-3 Terminals

Class-3 terminals do not have backup session support on the alternate IMS. Their terminal sessions must be restarted manually on the new active IMS after takeover, either by the MTO or by user logon.

**Recommendation:** Switch Class-3 terminals in the same manner as in the past, when an operator brought up a new CPC after a failure.

## Extended Recovery Facility (XRF) and IMS

All IMS terminals are eligible for class-3 service. Class-1 and class-2 terminals that have indicated no switching should be done are eligible for class-3. This is done by specifying `BACKUP=NO` on the system definition macro or ETO logon descriptor.

Class -3 terminals use the existing procedures to connect with the new active IMS after takeover. If no path to the new IMS subsystems exists, reestablishing the sessions might not be possible. All LU 6.2 conversations on the failing IMS must be reallocated after a takeover.

## Specifying Terminal Support

All terminals that IMS supports can be used in an XRF complex. The terminal characteristics determine the type of support a terminal can have in an XRF complex, with class-1 having the best support and class-3 having no support. Table 4 summarizes the keywords you can use to change the priority or terminal type of various devices.

Table 4. Summary of Defining Class of Service

Device	Class	BACKUP Keyword on TERMINAL Macro or ETO Logon Descriptor
SLUTYPE1 (3286) SLUTYPE2 (3278,3279,...) SLUTYPE4 (6670) SLUTYPEP (4700,8100, SERIES 1,...) FINANCE 3601	Eligible for class 1,2,3	For class-1 service, <code>BACKUP=(1-7,YES)</code> . For class-2 service, <code>BACKUP=(1-7,NO)</code> . For class-3 service, <code>BACKUP=NO</code> .
LUTYPE6 (ISC) MSC (VTAM) NTO (leased line) 3270 (VTAM) SPOOL line groups BTAM (nonswitched line) 3270 (BTAM) Locally attached devices	Eligible for class 2,3	For class-2 service, <code>BACKUP=(1-7,NO)</code> . For class-3 service, <code>BACKUP=NO</code> .
LU 6.2	Eligible for class 3	Not Applicable

The primary and the secondary master terminals are special types of terminals that exist simultaneously on both the active and alternate IMS systems. They are not treated as class-1 or class-2 terminals. The master terminals on the alternate IMS are not disrupted by a takeover. After takeover, the master terminal on the alternate IMS processes the undelivered messages from the active IMS system's master terminal.

The `BACKUP` keyword on several macros in the system definition statement or ETO logon descriptor specifies the actual support a terminal is to receive. Your installation determines which terminals have backup sessions established and the switching priority of the terminals, with level 1 being the lowest priority and level 7 the highest. Although VTAM requests are ordered by priority by IMS, the requests can be completed in any order because of unpredictable factors in the network, such as pacing, line speeds, and congestion.

If `BACKUP=(1-7,YES)`, a backup session is established during tracking, and the terminal is automatically taken over without losing session control. This constitutes class-1 support, if the terminal is connected to a 37x5, XRF-capable VTAM, and NCP.

## Extended Recovery Facility (XRF) and IMS

The maximum number of terminal backup sessions that can be established on the alternate IMS is specified in the NCP BUILD statement with the BACKUP=n option. If the (n+1) class-1 terminal user tries to log onto the active IMS, the logon is denied.

If BACKUP=(1-7,NO), this terminal is automatically restarted after takeover, but no BACKUP session is established. This forces class-2 support.

For both class-1 and class-2 terminals, the default switching priority is 4.

If BACKUP=NO, this terminal must be restarted manually after takeover. This forces class-3 support.

If options on the BACKUP parameter are set incorrectly, IMS resets the incorrect option at session initialization. For example, do not specify BACKUP for an ISC terminal. The highest class of support that an ISC terminal can receive is class-2. Therefore, it might not have a backup session established on the alternate IMS, and the YES is reset to NO.

---

## Chapter 3. Defining the Network

Performing a system definition for an IMS terminal network includes each of the following:

- Defining the names of the terminals
- Defining the terminal device types
- Specifying optional parameters, such as buffer sizes

The IMS network consists of static and dynamic terminals (ETO and LU 6.2 devices). You do not need to define your ETO or LU 6.2 devices to IMS.

Although only static terminals in the IMS network are defined by system definition macros, both static and dynamic terminals must be defined to VTAM. You must coordinate several aspects of the IMS requirements with parameters that are specified during VTAM network generation. This chapter highlights those coordination activities, but does not attempt to relate the separate IMS system definition in a detailed way with the series of VTAM and NCP definitions.

### **In this Chapter:**

- “Preparing for the Operational Network”
- “Coordinating IMS Definition and Network Definition” on page 70
- “Network Operation” on page 75
- “IMS Transaction Types and Transaction States” on page 76
- “Restarting the Network after Remote Takeover” on page 78
- “VTAM Definition for Transport Manager” on page 79

**Related Reading:** For an example of an IMS system definition, including definitions for various types of remote logical units see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

---

## Preparing for the Operational Network

In establishing the network, you must incorporate the IMS system definition requirements into the corresponding VTAM generations. You should also track the overall progress and the hardware installation schedules. The principal activities involved in establishing the IMS network consist of:

- Gathering the IMS requirements for physical terminals.
  - In addition, gather the following information regarding terminal requirements:
    - Organize input data for stage–1 IMS system definition for static terminals.
    - Define ETO descriptors and display screen characteristics for dynamically allocated terminals.
    - Define LU 6.2 descriptors.
    - Use the terminal profiles as part of the documentation for the IMS system design.
- Matching the terminals that are specified for IMS with those available for wider use by the installation.

Multiple users can exist on high-function terminals, and the terminals might not be dedicated to the IMS online system. You need to be able to identify these terminals to the network generation personnel and to identify expected usage by the end users.

## Preparing for the Operational Network

- Matching IMS system definition parameters to their counterparts in network generation.

Correct device function might require the matching of IMS SYSGEN parameters with VTAM and NCP network parameters. Be especially careful when defining buffer sizes. IMS and network buffer sizes must be compatible. Although system definition in IMS for ETO and LU 6.2 devices are not necessary, you must define them to VTAM.

- Monitoring the hardware installation plan.

Create a plan for tracking the progress of hardware installation and network generation to ensure that the change to production mode is not jeopardized. This tracking might involve interacting with hardware specialists who install and maintain terminal hardware.

- Anticipating terminal installations.

You might want to predefine terminals to IMS in anticipation of their installation.

**Related Reading:** For more information on signing on or logging on to terminals using user data, see “Chapter 7. Administering the Extended Terminal Option” on page 123.

For more information on signing on or logging on to terminals using user data, see “Chapter 7. Administering the Extended Terminal Option” on page 123.

For the information necessary to include APPC/IMS in your network, see “Chapter 18. Administering APPC/IMS and LU 6-2 Devices” on page 367.

---

## Coordinating IMS Definition and Network Definition

Provide input to VTAM and NCP regarding the following network requirements:

- “Using IMS as a Host Subsystem”
- “Defining VTAM Nodes” on page 71
- “Estimating VTAM Storage Requirements” on page 71
- “Determining VTAM Buffer Pool Values” on page 71
- “Determining the NCP Buffer Pool Values” on page 72
- “Determining Static and Dynamic Terminal Signon Requirements” on page 72
- “Checking Requirements for LOGON MODE Tables” on page 73
- “Specifying Initial VTAM Configurations” on page 74
- “Using SON/COS Support in IMS” on page 74

## Using IMS as a Host Subsystem

You must define the IMS control region to VTAM as an application. Your installation needs to decide on a name. This name is used in the definition of application nodes that are added to the SYS1.VTAMLST system data set.

**Example:** The following is an example of a control region definition:

```
IMS  APPL AUTH=(ACQ),PRTCT=password
```

This example shows that a password is required in order to start communication with the subsystem. Use the `PASSWD` keyword on the `COMM` macro to specify, for IMS system definition, the corresponding password. If the VTAM definition does not use IMS as the name for the IMS subsystem, code the chosen subsystem name in the `COMM` macro for IMS, using the `APPLID` keyword. The default `APPLID` name is IMS.

### Defining VTAM Nodes

The local node names are also added into SYS1.VTAMLST. You need to take special care that the exact terminal names used in the IMS system definition are repeated in the local list members of that data set. Other characteristics of the terminal are specified in the VTAM entries, and you must ensure that no inconsistencies exist between IMS and VTAM parameters. Associated with the node name are:

- The use of program function keys or selector pen
- Keyboard characteristics
- Screen sizes
- Model number
- LU type
- Transmission service level

IMS verifies screen size, model, LU type, and transmission service for dynamic terminals. If these are not correct, a session is not established.

**Recommendation:** If you are a DC administrator, assure that the VTAM definitions are correct. These definitions are used to select ETO descriptors and MFS format characteristics. IMS system definition sets these definitions for static terminals and ignores VTAM definitions.

**Related Reading:** For more information on defining VTAM nodes, see “Identifying VTAM Device Types, Screen Sizes, and Models” on page 128.

### Estimating VTAM Storage Requirements

Examine the full set of terminal options that are planned for the IMS network and select those communication options that might have an effect on the VTAM storage requirements. For example, choose the appropriate:

- Protocol (BID or NOBID)
- Type of response patterns
- Number of active nodes

Using your terminal profiles, you can respond to the need for system programming information to estimate storage requirements for IMS support.

### Determining VTAM Buffer Pool Values

Provide the VTAM system programmer with IMS input and output message sizes. The values you select for the number and size of receive-any buffers, as specified on the RECANY keyword of the COMM macro, are of special importance. You must specify the largest number of receive-any buffers that are required in order to support the VTAM network. You can override the size with the execution parameter, RECASZ, and the number with the execution parameter, RECANY. Session initiation fails for terminals that are added to the system if they require a larger receive-any buffer size than the size that is currently specified. You must restart the IMS system, specifying the increased buffer size, before the terminal can establish a session with IMS.

## Coordinating IMS Definition and Network Definition

### Determining the NCP Buffer Pool Values

Work with the system programmer to assess the volume of predicted traffic for application programs. The NCP buffer pool sizes and thresholds are based on the VTAM buffer pool information. No specific IMS requirements exist for NCP system generation parameters.

### Determining Static and Dynamic Terminal Signon Requirements

All VTAM-defined terminals can sign on by providing user data with the session initiation request, regardless of whether signon is required for the terminal.

You cannot sign on a user to both a dynamic terminal and a static terminal simultaneously, unless you specify SGN=G, M, or Z on the EXEC parameter to allow multiple signons. Otherwise, IMS issues message DFS3649 or DFS2467.

A user cannot enter signon data from an output-only device through logon or a /SIGN command. When signon data is omitted at session initiation for an output-only device that requires a signon, message DFS2085 is sent to the MTO.

#### Multiple Signons

You cannot use the same name for dynamic LTERMs as for static LTERMs that are defined during system definition.

You can be currently signed on to one or more terminals, static or dynamic. For static terminals, include the EXEC parameter, SGN=M (which enables multiple signons).

**Related Reading:** For more information about dynamic terminals, see "Signing On Multiple Times" on page 152.

The IMS master terminal receives a security violation message, DFS286, for rejected signon attempts. If you do not want to receive the message, set SECCNT to 0 on the COMM, MSGEN, or SECURITY macro.

**Related Reading:** For more information on disabling security violation messages, see *IMS/ESA Customization Guide*.

#### Session Status Message—DFS3650

When signon is achieved or if signon is not required, message DFS3650 is sent to the user indicating the status of the session with IMS.

**Exception:** The following terminals do not receive these messages:

- Autologon terminals
- SLU-1 terminals running in unattended mode
- ISC
- SLU P
- 3600/Finance

In addition, if NOTERM is specified on the TERMINAL macro or on the ETO user descriptor, no DFS3650 message is received.

**Related Reading:** For more information about when message DFS3650 is issued, see *IMS/ESA Messages and Codes*.



## Coordinating IMS Definition and Network Definition

The information presented on the DFS3650 message panel shows whether the user is in conversation mode. This status panel also shows whether user output security exists for the terminal.

### Checking Requirements for LOGON MODE Tables

When a static VTAM terminal is connected to IMS and a session is initiated, VTAM needs to know the LOGON MODE identifier. A table, the LOGON MODE table, exists in VTAM; it contains the SNA bind parameters for the session and the identifier (the LOGMODE parameter value, which is used at logon) for that terminal. Also, a default identifier for each SDLC-connected terminal exists.

These considerations apply to static VTAM terminals.

**Related Reading:** For more information on dynamic VTAM terminals, see “Identifying VTAM Device Types, Screen Sizes, and Models” on page 128.

### Coordinating LOGON MODE Identifiers

When you are deciding on the LOGON MODE identifiers, you need to:

- Match the naming conventions that are being used for the VTAM network in your installation. Your choice of a name might need to follow this convention.
- Arrange for automatic starting of IMS-remote terminals, or have them controlled by the VTAM operator. Otherwise, choose names that are meaningful to a remote terminal operator, if in order to initiate a session the remote terminal operator (RTO) enters the VTAM LOGON command.
- Choose names that are helpful to the master terminal operator if sessions are to be started with the /OPNDST command. If a terminal can be operated with different SNA protocols, the name should help distinguish between the operating modes.
- Use the MODETAB keyword on the TERMINAL macro or ETO logon descriptor to specify the default identifier that is to be used in the following situations:
  - The default identifier is not used.
  - The terminal operator does not specify the identifier in the LOGON command.
  - The master terminal operator does not specify the identifier in the /OPNDST command.

**Examples:** The code below illustrates the nature of the LOGON MODE table entries that are described to VTAM for a 3270 SDLC connection or a 4730 device. IMS defines these as SLU 2 and SLU P respectively.

```
SE3270  MODETAB
IBMS3270  MODEENT LOGMODE=S3270,FMPROF=X'02',TSPROF=X'02',      X
          PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000'
MT4730  MODEENT LOGMODE=MT4730,FMPROF=X'04',TSPROF=X'04',      X
          PRIPROT=X'B1',SECPROT=X'B1',COMPROT=X'6080',      X
          RUSIZES=X'0000'
MODEEND
```

The operator can enter either the VTAM LOGON command or use the VTAM USSTAB command for installation-determined logon syntax.

**Example:** The following example specifies a KEYWORD=*value* format:

```
U3270  USSTAB
        USSCMD      CMD=IMS,REP=LOGON
        USSPARM     PARM=APPLID,DEF=IMS
        USSEND
        END
```

## Coordinating IMS Definition and Network Definition

### Logon Requirements for the Master Terminal

Because IMS automatically logs on the master terminal after the IMS START command is issued, you must code the MODETBL keyword on the LU statement for the physical device that is used as the MTO if the VTAM default is not to be used.

When the MTO is using the /OPNDST command, a MODE operand applies to all node names presented in the command.

## Specifying Initial VTAM Configurations

Based on your IMS requirements, start lists (ATCSTRyy) and configuration members (ATCCONxx) are placed in the SYS1.VTAMLST library. This information should reflect those terminals that are to be activated at VTAM startup, and those VTAM nodes that are known to VTAM at startup. For ISC (LU 6) devices and MSC VTAM links, if the session is started from another subsystem, the session is not automatically recovered. Session failure messages are sent to the MTO indicating that you must manually restart the session.

You should emphasize flexibility in the IMS requirements, and try to reduce the requirements for a large part of the network to be immediately available. This might entail more elaborate startup instructions for start lists and configuration members. This might also be reflected in your MTO instructions and the extent to which the /OPNDST command is used. The delayed start up of network sections can overlap with IMS processing in order to allow IMS to start up more quickly. Also, the network startup sequence can be executed by an automated operator program executing in IMS.

## Using SON/COS Support in IMS

Session outage notification (SON) and class of service (COS) are facilities of VTAM and SNA that allow IMS to recognize a session outage. IMS attempts to restart the session for recoverable session failures. Both SON and COS must be specified in VTAM to be available for IMS use. In addition, the ASR option must be specified for the IMS node using the IMS SYSDEF or the /CHANGE command.

**Related Reading:** For more information on using the /CHANGE command, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

The IMS SON and COS support are available for all the VTAM SNA terminals and MSC VTAM links. When multiple virtual routes exist between IMS and the other LU, use an alternate route to avoid a network outage.

The SON facility enables VTAM to inform IMS that a session failure has occurred. The SON facility gives installations the option of allowing IMS to automatically restart sessions (without end-user intervention) if a recoverable session failure occurs in the network.

You need to code SONSCIP=YES on the VTAM APPL definition statement for the IMS application in order to activate automatic session restart after a session outage. If you decide not to include the SONSCIP definition for IMS, VTAM and IMS proceed with session termination without automatic session restart.

The COS facility allows VTAM to control selection of actual routes by designating a set of virtual communication routes based on speed of traffic, data type, and

## Coordinating IMS Definition and Network Definition

security considerations. These sets of routes are defined in the VTAM COS table. The mode table entry that is used for session establishment can specify a COS entry name.

When the set of session BIND parameters associated with the terminal contains a COS name, these specified virtual routes are scanned to determine which one can be used for the session. If the selected virtual route fails, restart of the session causes the COS set to be scanned again for use of a different virtual route for session reestablishment.

For program LUs (as well as Finance, LU P, MSC, and ISC), in-flight messages can be recovered and retransmitted as necessary, thereby preserving the integrity of the message exchange across the restart.

For device LUs, in-flight messages might be lost or duplicated.

SON and COS support preserves the session setup options (BIND parameters), as well as the class of service associated with the failing session by using the same mode table entry name in order to reestablish a new session.

**Restrictions:** Because the boundary node NCP cannot distinguish an upstream route failure from a failure of a host processor, SON and COS support cannot be used for terminals in XRF configurations that have backup sessions.

---

## Network Operation

This section explains how to start an IMS network and how to initiate a session.

### Starting an IMS Network

Before a session with IMS can be established, VTAM and NCP must be active.

To make IMS ready to receive VTAM logon (session initialization) requests, use the IMS /START command with the DC keyword. The /START DC command tells VTAM to pass to IMS any queued VTAM logon requests to IMS, as well as logon requests for any logical unit that is defined to VTAM as belonging to IMS.

The /START DC command activates the following processes:

- Initiates IMS data communication processing
- Opens the VTAM access method control block
- Enables the IMS VTAM Logon exit routine

Any logon requests VTAM receives before the IMS /START DC command but after the IMS VTAM access method control block (ACB) has been opened are queued in VTAM until the /START DC command is completed. If VTAM is active when IMS is initialized, the IMS VTAM ACB is opened; otherwise, the ACB is opened by the /START DC command.

Use the /START APPC command to start APPC/IMS.

### Session Initiation

**Definition:** A *session* is the logical connection between a logical unit and a VTAM application program (such as IMS). A session must be established before data can be transmitted between a logical unit and IMS.

## Network Operation

Session initiation is requested in one of five ways:

- The terminal operator enters the LOGON sequence. VTAM verifies the command and passes the request to IMS.

To participate in session balancing across a Sysplex, the terminal operator can specify a generic resource name.

If the terminal is requesting a session with an IMS XRF complex, the IMS USERVAR name and the application program name should be the same. VTAM distributes this USERVAR value across domains in the network, eliminating the need for you to design, code, and maintain NetView CLISTs in order to propagate USERVAR changes.

- The MVS VTAM network operator requests session initiation on behalf of the logical unit by using the VTAM VARY command with the LOGON option. VTAM processes the request and passes it to IMS.
- VTAM passes a logon request to IMS for each logical unit that is defined to VTAM as belonging to IMS.
- The IMS master terminal operator requests session initiation for a logical unit by entering the IMS /OPNDST command.
- ETO autologon initiates the session and supplies the appropriate user data, based on user descriptors and exit routines.

**Related Reading:** For more information on logging on to your terminal and sharing printers, see “Logging onto ETO Terminals” on page 150 and “Sharing Printers Using ETO” on page 158.

For LU 6.2, session initiation occurs automatically as conversations are allocated. The IMS MTO can request delivery of queued LU 6.2 output by issuing the /ALLOCATE command. When using LU 6.2, remember that the IMS application name (which matches the LU name) for LU 6.2 conversations is different than the LU name IMS has for non-LU 6.2 types, and that the /START DC and /START APPC commands work independently.

Regardless of how session initiation is requested, identical processing occurs when IMS receives the request.

---

## IMS Transaction Types and Transaction States

This section describes transaction types and transaction states.

### Defining IMS Transaction Types

Transactions are the most common type of data that is sent from a logical unit to IMS. IMS supports two kinds of transactions—update and inquiry.

**Definitions:**

- An *update* transaction can modify a database.
- An *inquiry* transaction can look at data in a database, but it cannot change or update it.

You define transactions as update or inquiry during IMS system definition.

An additional attribute is defined for inquiry transactions—recoverable or irrecoverable.

**Definitions:**

## IMS Transaction Types and Transaction States

- *Recoverable-inquiry transactions* are always recoverable, regardless of which element in the network fails.
- *Irrecoverable-inquiry transactions* are not recovered after an I/O error condition occurs or at IMS system restart.

All update transactions are recoverable. All Fast Path transactions must be defined as recoverable, but they can be either inquiry or update.

An LU 6.2 transaction is recoverable only if the asynchronous protocol is used to initiate the transaction. IMS conversational transactions from LU 6.2 programs are not recoverable in the local system, but they are recoverable across the MSC link.

IMS treats a irrecoverable transaction in the same manner as a recoverable transaction, except that all processing that required to achieve recoverability is eliminated.

As a result, irrecoverable transactions require less processing time, but they could be lost in the event of a network failure (for example, line failure, processor failure, or queue failure).

## Determining Transaction States

The possible transaction states are:

**LOCK** A transaction in LOCK state does not receive messages. Scheduled messages containing this transaction code are stopped.

**Restriction:** The /LOCK TRANSACTION command cannot be used with Fast Path-exclusive transactions, but it can be used with Fast Path-potential transactions.

**MODSTOPPED** A transaction in MODSTOPPED state cannot receive input, because online change processing is in progress. A /MODIFY COMMIT command sets this status. A CPI Communications driven transaction cannot be marked MODSTOPPED.

**PSTOP** A transaction in PSTOP state cannot be scheduled; however, the transaction continues to be processed until the limit count is reached. If the limit count is large, the processing interval is long. To ascertain the status of the transaction, use the /DISPLAY command; to alter the status of the transaction, use the /ASSIGN command.

**PURGE** A transaction in a PURGE state has had its input messages stopped.

**QSTOP** A transaction in QSTOP state cannot be entered during the time between the completion of a /MODIFY PREPARE command and the completion of the corresponding /MODIFY COMMIT or /MODIFY ABORT command. Transactions that are known to be affected by the content of an online change are rejected; that is, the transactions are to be changed or deleted, or they could access databases or programs that are to be changed or deleted. By using the /DISPLAY MODIFY command, you can cause a list of transactions that are affected by a current online change to be displayed. At the terminal, such a transaction is rejected with message DFS3470. If the transaction uses a Fast Path routing code that is changed or

## IMS Transaction Types and Transaction States

deleted, the rejection message is DFS3471. A CPI Communications driven transaction is never in a QSTOP state.

<b>STOP</b>	A transaction in STOP state is stopped. The queuing and scheduling of messages that are destined for a transaction or class of transactions are stopped. However, output can still be queued if it originates from an application program.
<b>USTOPPED</b>	USTOPPED is a status value that is set when an application program attempts to use an IMS DL/I database resource that is not available, and the program terminates abnormally with abend U3033. The USTOPPED condition is not set for a CPI Communications driven program. The CPI Communications driven application program is abnormally terminated with an abend U0125.

---

## Restarting the Network after Remote Takeover

Unlike XRF, which automatically reestablishes data communications with class-1 and class-2 terminals, RSR remote takeover requires that you be involved in restarting the terminal network at the new active site. Your new active site (the old tracking site) should already have a proper system definition, including all ETO descriptors, terminal definitions specified in the TERMINAL macro, and all necessary VTAM definitions. Then, after remote takeover, you must restart the terminal network.

Before starting non-VTAM terminals, you need to physically switch all telecommunication lines and terminals to the new active subsystem. Because all non-STSN terminals have no knowledge of their work that was done immediately prior to the remote takeover, you receive no indications at the terminal or from IMS whether the remote takeover causes message data to be lost.

VTAM terminals must be disconnected from the old active site before they can log on to the new active subsystems at the new active site.

Finance terminals, SLU-P terminals, ISC terminals, and MSC links can cause IMS to recognize a lost-data condition after a remote takeover. If IMS recognizes this condition, the session resynchronization attempt fails, the MTO is notified, and IMS automatically forces the session or the MSC link to be cold started. Data loss is not recognized if information about the terminal's initial (cold start) logon is also lost. If IMS does not know the terminal was active at the old active site, it cold starts the session at the next logon. The terminal itself can recognize this action and issue messages (DFS2948 in a non-MSC environment, or DFS3311 and DFS3212 in an MSC environment), which you should save for possible manual resynchronization (recovery). From the messages and an associated CVCT log record, you can determine which input and output messages were lost, resulting in the resynchronization error.

**Related Reading:** For information on determining message loss following an RSR takeover, see *IMS/ESA Diagnosis Guide and Reference* .

An IMS ISC remote primary half session can detect an STSN problem from the RSR secondary half session after remote takeover, but it cannot perform the automatic forced session cold start. The remote primary half session terminates with a DFS2050 message, and the MTO must manually force a session cold start using the following commands:

- /STO NODE *node* USER *user*

## Restarting the Network after Remote Takeover

- /ASSIGN USER *user* TO VTAMPOOL
- /STA NODE *node*

VTAM MSC links can also be out of synchronization after a remote takeover if link cold start is not automated. The links and their sessions can be resynchronized in the same way as for unrecoverable data transmission errors when using VTAM in a non-RSR environment:

- Stop the link using the /PSTOP command
- Change the link using the /CHANGE command (SYNCSESS to FORCSESS)
- Restart the link using the /RSTART command, thus forcing a session cold start

MTM and CTC links cannot be restarted if significant data is lost, if the sessions are out of synchronization, and if the new active subsystem requires a cold start. If automatic restart fails for SLU-P or Finance devices, you must cold start IMS.

---

## VTAM Definition for Transport Manager

To use the IMS transport manager, some VTAM definitions are required, such as APPL definition and mode table definition.

### **Related Reading:**

- For information on VTAM definitions, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.
- For specific information on VTAM definitions, see the appropriate VTAM publications.

## VTAM Buffering and Pacing

When VTAM runs out of buffers, it returns errors to SEND operations, which result in transport manager deallocation. In other words, if pacing causes VTAM to stop accepting data and the transport manager buffers for the conversation fill up, log data transport is temporarily suspended. To avoid this transport manager deallocation, use pacing (APPL VPACING=) and allocate enough VTAM IOBUFs to support a number of bytes of data equal to:

$$VPACING \times \log\_block\_size$$

If session pacing is not used, a high-volume component (such as the IMS logger or IMS isolated log sender) will probably consume buffers faster than VTAM can empty them.

## VTAM Missing Interrupt Handler

You must be sure to include VTAM Missing Interrupt Handler (MIH) coding in the appropriate VTAM and MVS definitions. If you do not include MIH in these definitions, and if the CTC link becomes inoperative, MIH continues to appear active but is likely to be hung or unusable. And if the CTC link is inoperative, the transport manager subsystem is likewise inoperative.





---

## Chapter 4. Using Shared Queues in the Network

This chapter contains information for system programmers who are enabling shared queues and administering the IMS Transaction Manager network in a shared-queues environment.

### **In this Chapter:**

- “Enabling Shared Queues”
- “Using the Common Queue Server” on page 86
- “Tuning for Performance in a Shared-Queues Environment” on page 91

---

### Enabling Shared Queues

This section describes the tasks associated with enabling shared queues in your Sysplex environment.

### Defining the CFRM Policy

To enable shared queues, you need to define a coupling facility resource management (CFRM) policy that contains structure names and attributes for all message queues and EMH queues. Figure 13 on page 82 contains a sample CFRM policy with these definitions.

Be sure that the structure names you define in the CFRM policy for message queues (and EMH queues) are also defined in the following places:

- The shared-queues IMS.PROCLIB member (DFSSQxxx) parameters MSGQ= and EMHQ=
- The CQS local structure definition PROCLIB member (CQSSLxxx) parameter STRNAME=
- The CQS global structure definition PROCLIB member (CQSSGxxx) parameters STRNAME= and OVFLWSTR=

**Recommendation:** If possible, place high-use structures on separate coupling facilities. For example, place your IMS data-sharing structures on a different coupling facility from your shared-queues structures. Place your VSAM and DB2 structures on yet another coupling facility. You can place low-use structures (such as those for RACF) on any coupling facility where space is available.

### **Related Reading:**

- For more information on defining a CFRM policy, see *IMS/ESA Administration Guide: System*.
- For more information on definitions that are required to use CQS, see *IMS/ESA Common Queue Server Guide and Reference*.
- For more information on IMS.PROCLIB, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

## Enabling Shared Queues

```
//POLICY EXEC PGM=IXCM2APU
//STEPLIB DD DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(CFRM)
DEFINE POLICY NAME(CONFIG01) REPLACE(YES)

CF NAME(LF01) TYPE(ND01%%)
           MFG(%%)
           PLANT(%%)
           SEQUENCE(%%%%%%%%%)
           PARTITION(0)
           CPCID(00)
           DUMPSPACE(1400)

STRUCTURE NAME(MVSL0GMSGQ01)
           INITSIZE(1024)
           SIZE(1536)
           PREFLIST(LF01)

STRUCTURE NAME(MVSL0GEMHQ01)
           INITSIZE(1024)
           SIZE(1536)
           PREFLIST(LF01)

STRUCTURE NAME(STRMSGQ01)
           INITSIZE(1024)
           SIZE(1536)
           PREFLIST(LF01)

STRUCTURE NAME(STREMHQ01)
           INITSIZE(1024)
           SIZE(1536)
           PREFLIST(LF01)

STRUCTURE NAME(STRMSGQ010FLW)
           SIZE(256)
           PREFLIST(LF01)

STRUCTURE NAME(STREMHQ010FLW)
           SIZE(256)
           PREFLIST(LF01)
/*
```

Figure 13. Sample CFRM Policy Containing Definitions of Shared-Queues

### Defining Message Queue List Structures

You must define the primary list structure, MSGQ, to IMS before using shared queues. You can optionally define the overflow structure.

**Example:** In Figure 13, STRUCTURE NAME(IMSMGQ01) defines a message queue structure, and STRUCTURE NAME(IMSMGQ010FLW) defines an overflow structure for the IMSMGQ01 message queue.

### Defining Fast Path Message Queue List Structures

If you define the MSGQ primary list structure on an IMS system that has Fast Path defined, you must also define a primary list structure for shared Expedited Message Handler (EMH) queues. This structure is called the EMHQ structure. You can optionally define an EMHQ overflow structure.

**Example:** In Figure 13, STRUCTURE NAME(IMSEMHQ01) defines an EMHQ structure, and STRUCTURE NAME(IMSEMHQ010FLW) defines an overflow structure for the IMSEMHQ01 EMH queue.

## Defining an MVS Log Stream

You must define an MVS log stream. Create the following MVS log stream definitions:

- In the GRSTNLxx PARMLIB member, specify every IMS system in the Sysplex environment as being in the same global resource serialization complex. This allows global serialization of resources in the Sysplex.
- Define log stream structure names and attributes in the CFRM policy for message queues and EMH queues.

**Example:** In Figure 13 on page 82, STRUCTURE NAME (MVSLOGMSGQ01) defines a log stream structure name for the message queues. Similarly, STRUCTURE NAME (MVSLOGEMHQ01) defines a log stream structure for Fast Path EMH queues.

- Install DFSMS and change SMS DATACLAS, STORCLAS, and MGMTCLAS constructs.
- Format the inventory coupled data set and define log streams and structure names.
- Update the COUPLExx PARMLIB member.
- Define the logger inventory data set in SYS1·PARMLIB(COUPLExx).
- Define log stream and coupling facility structure names to the logger inventory coupled data set.

**Related Reading:** For more information on defining the MVS log stream, see *IMS/ESA Common Queue Server Guide and Reference*.

## CQS Definitions

To enable shared queues, define the following CQS parameters:

- CQS initialization parameters in PROCLIB member CQSIPxxx:

**ARMRST=Y|N** Specifies whether to use the MVS Automatic Restart Manager to restart the CQS address space after an abend occurs. ARMRST= is optional. The default is ARMRST=Y.

**CQSGROUP=** Specifies a one- to five-character identifier representing the XCF CQS shared-queues group name, which is concatenated to the letters CQS. The CQSGROUP= name can be the same as the SQGROUP= name.

**SSN=** Specifies the CQS address space subsystem name.

**STRDEFG=** Specifies the CQSSGxxx suffix.

**STRDEFL=** Specifies the CQSSLxxx suffix.

- CQS local structure definition parameters in PROCLIB member CQSSLxxx (for local structure definition):

**CHKPTDSN=** Specifies the checkpoint data set name.

**STRNAME=** Specifies the structure name.

**SYSCHKPT=** Specifies the number of log records that were written between system-generated checkpoints.

These definitions apply only to a single CQS.

- CQS global structure definition parameters in PROCLIB member CQSSGxxx (for global structure definition):

## Enabling Shared Queues

<b>LOGNAME=</b>	Specifies the name of the MVS log stream.
<b>OBJAVGSZ=</b>	Specifies the average message size.
<b>OVFLWSTR=</b>	Specifies the name of the overflow structure (optional).
<b>OVFLWMAX=</b>	Specifies the maximum overflow threshold level. OVFLWMAX is optional. The default is OVFLWMAX=70.
<b>SRSDSN1=</b>	Specifies the name of the first structure recovery data set.
<b>SRSDSN2=</b>	Specifies the name of the second structure recovery data set.
<b>STRMIN=</b>	Specifies the minimum structure size. STRMIN is optional. The default is STRMIN=0.
<b>STRNAME=</b>	Specifies the name of the structure.

You must define one of each of these parameters for each structure pair. All of the CQs that share queues must have the same values specified in the CQSSGxxx PROCLIB member. If they differ, only the first CQs connects to the structure, and the others fail.

- Execution parameters (optional):

<b>ARMRST=Y N</b>	Specifies whether to use the MVS Automatic Restart Manager to restart the CQS address space after an abend occurs. ARMRST= is optional. The default is ARMRST=Y.
<b>BPECFG=</b>	Specifies an 8-character name for the BPE configuration parameters PROCLIB member.
<b>CQSGROUP=</b>	Specifies a one- to five-character identifier representing the XCF CQS shared-queues group name, which is concatenated to the letters CQS. The CQSGROUP= name can be the same as the SQGROUP= name.
<b>CQSINIT=</b>	Specifies a 3-character suffix for the CQS initialization parameters PROCLIB member (CQSIPxxx). The default suffix is 000.
<b>SSN=</b>	Specifies the CQS address space subsystem name. If specified, this parameter overrides the value specified in the CQSIPxxx PROCLIB member.
<b>STRDEFG=</b>	Specifies a 3-character suffix for the CQS Global Structure Definition PROCLIB member (CQSSGxxx). The default suffix is 000. If specified, this parameter overrides the value specified in the CQSIPxxx PROCLIB member.
<b>STRDEFL=</b>	Specifies a 3-character suffix for the CQS Local Structure Definition PROCLIB member (CQSSLxxx). The default suffix is 000. If specified, this parameter overrides the value specified in the CQSIPxxx PROCLIB member.

If you do not specify ARMRST=, CQSGROUP=, SSN=, STRDEFG=, or STRDEFL=, CQS retrieves the values that are specified in the CQSIPxxx PROCLIB member.

**Related Reading:** For more information on these CQS definitions, see *IMS/ESA Common Queue Server Guide and Reference*.

### IMS Definitions

To enable shared queues, define the following IMS parameters:

- IMS control region execution parameters:

**LGMSGSZ=** Specifies the record length of the long message.

**QBUF=** Specifies the upper expansion limit of the queue buffer pool.

**QBUFHITH=** Specifies a one- to three-digit number with a value in the range of 1 to 100 representing the high threshold percentage at which the message queue buffer is dynamically expanded. The default percentage is 80%.

**QBUFLWTH=** Specifies a one- to three-digit number with a value in the range of 1 to 100 representing the low threshold percentage at which the message queue buffer is compressed. The default percentage is 50%.

**QBUFMAX=** Specifies the maximum number of message queue buffers allowed in the queue buffer pool.

**QBUFPCTX=** Specifies a one- to three-digit number with a value in the range of 1 to 100 representing the percentage of the originally allocated message queue buffers that are dynamically expanded when QBUFHITH is reached. The default percentage is 20%.

**QBUFSZ=** Specifies the size of the queue buffers.

**SHAREDQ=** Specifies the suffix of the shared-queues PROCLIB member, and is located in the PARMLIB member DFSSQxxx.

**SHMSGSZ=** Specifies the record length of the short message.

- IMS startup parameters in PROCLIB member DFSSQxxx:

**CQS=** Specifies the PROCLIB member containing the CQS procedure. CQS= is optional. The default is CQS=CQS.

**CQSSSN=** Specifies the name of the CQS address space subsystem.

**EMHQ=** Specifies the name of the EMH queues structure.

**MSGQ=** Specifies the name of the message queues structure.

**SQGROUP=** Specifies a one- to five-character identifier representing the XCF IMS shared-queues group name, which is concatenated to the letters DFS. All IMS subsystems that share the same set of structures must specify the same SQGROUP= name. The SQGROUP= name can be the same as the CQSGROUP= name, which is specified in the CQSIPxxx PROCLIB member.

**WAITRBLD=Y|N**

Specifies whether activity on the EMHQ structure should wait until CQS completes the structure rebuild process.

WAITRBLD=NO specifies that activity on the EMHQ structure should continue while CQS rebuilds the structure. WAITRBLD= is optional. The default is WAITRBLD=N.

**Related Reading:** For more information on these IMS parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

### Using the Common Queue Server

IMS uses the CQS interface to manage the shared queues in a Sysplex environment. From each IMS subsystem that acts as a CQS client, IMS uses this CQS interface to access the shared queues.

**Restriction:** When using the same LTERM name on multiple systems in a shared queues sysplex, all conversational and response mode reply messages must be processed by the originating terminal's system. This means that while a conversation or response mode operation is still in-progress on one IMS system, the same LTERM name cannot be used, simultaneously or serially, by a terminal on another IMS system. This Transaction Manager (TM) restriction applies to both Fast Path and non-Fast Path in a shared queues environment.

This section briefly describes how to use CQS in an IMS environment.

### Starting CQS

You can activate CQS in one of two ways:

- As an MVS task, using the MVS START command
- As an MVS batch job

In addition, IMS automatically starts CQS, when appropriate.

To customize and monitor your CQS environment, you can supply any of the following exit routines:

- CQS Initialization/Termination
- CQS Client Connection
- CQS Queue Overflow
- CQS Structure Statistics
- CQS Structure Event

**Related Reading:** For information on CQS initialization parameters and CQS user-supplied exit routines, see *IMS/ESA Common Queue Server Guide and Reference*.

### Restarting CQS

Depending on whether a CQS structure contains data, you can warm start CQS or cold start CQS:

- If the structure is empty, you must cold start CQS.
- If the structure contains data, you can either warm start or cold start CQS.

When you have completed restarting CQS, the CQS ready message is issued (CQS0020I).

#### CQS Warm Start

During a warm start, CQS reads the checkpoint data set to find the log token representing the last system checkpoint. When CQS finds this log token, it initiates a warm start. If CQS fails to find this log token in the checkpoint data set, it reads the log token from the structure. If CQS finds the log token, it issues a WTOR in order to allow you to confirm the use of this token.

At this point, you can:

- Confirm the log token
- Cold start CQS
- Cancel CQS
- Specify a new log token

Sometimes, CQS purges log tokens that are required for restart. In the following situations, CQS purges log records that are more than two structure checkpoints old:

- During a structure checkpoint.
- When the log becomes full, and no more data sets are available for logging.

**Recommendation:** If a CQS does not accept a log token during CQS restart, cold start the CQS. In cases where multiple CQSs are running, it is possible that log records for a CQS that previously failed and was not restarted are purged while another CQS performs a structure checkpoint.

### CQS Cold Start

When CQS cold starts after connecting to an empty structure, CQS purges all log records for the structure and performs a system checkpoint. After the system checkpoint is complete, the structure and CQS restart is complete.

## Using IMS Commands to Access CQS

You can use the following IMS commands to interact with a CQS:

### **/CQCHKPT SHAREDQ**

Initiates a CQS structure checkpoint.

### **/CQCHKPT SYSTEM**

Initiates a CQS system checkpoint.

### **/CQQUERY STATISTICS**

Displays statistics for primary and overflow structures.

### **/CQSET SHUTDOWN SHAREDQ**

Sets status to take structure checkpoint at CQS shutdown.

### **/DEQUEUE**

Dequeues one or more messages from one of the following:

- An outbound APPC queue
- A Fast Path program
- An LTERM
- An MSNAME
- An outbound OTMA queue
- A message queue
- A remote transaction or a remote LTERM
- A transaction

### **/DEQUEUE SUSPEND**

Dequeues one or more suspended transactions.

### **/DISPLAY CQS**

Displays CQS information that IMS tracks.

### **/DISPLAY MODIFY**

Displays local work in progress.

## Using the Common Queue Server

### **/DISPLAY OVERFLOWQ**

Displays a list of queue names that are in overflow mode.

### **/DISPLAY QCNT**

Displays global queue information for a particular resource type.

### **/DISPLAY STRUCTURE**

Displays structure status.

### **/DISPLAY TRACE TABLE QMGR**

Displays queue manager trace table status.

### **/DISPLAY TRACE TABLE SQTT**

Displays shared-queues trace table status.

### **/TRACE SET OFF TABLE QMGR**

Stops trace of queue manager activity.

### **/TRACE SET ON TABLE QMGR**

Starts trace of queue manager activity.

### **/TRACE SET OFF TABLE SQTT**

Stops trace of shared-queues activity.

### **/TRACE SET ON TABLE SQTT**

Starts trace of shared-queues activity.

These IMS commands apply to the local IMS system only. Commands and command responses are not placed on the shared queues; therefore, they are not recoverable.

**Related Reading:** For the complete syntax of these IMS commands, see *IMS/ESA Operator's Reference*.

## Using CQS User-Supplied Exit Routines

- The CQS Initialization/Termination exit routine is notified when:
  - CQS starts, before connecting to a structure
  - CQS terminates after disconnecting from all structures (under normal termination conditions)
- The CQS Client Connection exit routine is notified when clients connect to structures, and when clients disconnect from structures.
- The CQS Queue Overflow exit routine allows the client to approve or reject a queue that CQS selects for overflow.

When IMS reaches a predetermined threshold for a primary queue structure, CQS can take one or both of the following actions:

- Select queues for overflow mode
- Dynamically expand structure size

You can use the CQS Queue Overflow exit routine to approve or reject a queue that CQS selects for overflow processing. CQS is then able to do one of the following with the overflow queues:

- Move the selected queues to an overflow structure
- Reject future queue messages for selected queues if no overflow structure is available or if the overflow structure is full
- The CQS Structure Statistics exit routine is passed structure statistics that CQS gathers during CQS system checkpoints.



- The CQS Structure Event exit routine is notified of structure events, such as structure copy, structure recovery, structure overflow, and CQS connection to and disconnection from structures.

**Related Reading:** For more information on CQS exit routines, see *IMS/ESA Customization Guide*.

## Changing the Size of Coupling Facility Structures

The initial size of a structure on the coupling facility is determined by the value of the INITSIZE parameter in the CFRM policy. CQS allows you to dynamically reconfigure the size of a structure:

- Define the INITSIZE (initial size of the structure) and SIZE (maximum size of the structure) parameters in the CFRM policy.
- An MVS operator initiates the structure size change by entering the following XES command: `SETXCF START,ALTER,STRNAME=strname,SIZE=size`

The value of *size* must be within the range of values between INITSIZE and SIZE in the CFRM policy.

When the first CQS connects to a structure, the value specified for INITSIZE is the size of that structure. If enough free space does not exist for this INITSIZE value, the size of the structure becomes that of the available space in the coupling facility.

## Initiating System Checkpoint

This section introduces CQS system checkpoint, the checkpoint data sets that are used for recovery, and how CQS restarts after system checkpoint.

As a system checkpoint for recovering CQS information during restart, each CQS writes its own control blocks to the MVS log. To retrieve this restart information, CQS also writes information to the CQS checkpoint data set. CQS does not quiesce activity while the checkpoint is in progress.

You can initiate a CQS system checkpoint by taking either of the following actions:

- Code the SYSCHKPT= parameter in the CQSSLxxx PROCLIB member in order to initiate a system checkpoint by log volume.
- Issue the IMS command `/CQCHKPT SYSTEM`.

### Checkpoint Data Sets

For each structure pair, CQS maintains a checkpoint data set. CQS writes to a checkpoint data set and then uses it during restart.

Checkpoint data sets are dynamically allocated during CQS initialization. You define the checkpoint data set DSNAME for a structure in the CQS local structure definition PROCLIB member (CQSSLxxx), using the CHKPTDSN= parameters.

### How CQS Restarts after System Checkpoint

During CQS restart, CQS reads the log records from the last system checkpoint and rebuilds the work that was in progress. The frequency of system checkpoint affects this restart. CQS must read many more log records when checkpoints are infrequent than when the checkpoints occur more often.

## Initiating Structure Checkpoint

In order to recover message queues, CQS takes structure checkpoints. CQS copies the shared queues from a structure pair to a structure recovery data set (SRDS).

## Using the Common Queue Server

CQS quiesces activity while it performs part of this copy operation. CQS then performs a system checkpoint following each structure checkpoint.

CQS performs structure checkpoints in either of the following situations:

- When the MVS log becomes full.
- When you enter the `/CQCHKPT SHAREDQ` command.

CQS uses the SRDSs to recover structures after structure or coupling facility failure.

Taking more frequent structure checkpoints also frees space in the log stream.

## Recovering Structures

This section describes recovery procedures for coupling facility structures and the message queues on those structures.

MVS allows you to rebuild structures. You rebuild structures either by copying them or by recovering them.

### Rebuilding Structures

One or more CQSs must be running in order to copy or recover structures and the messages on those structures. When the new structure is allocated, policy changes (such as structure location) are applied. A structure rebuild is initiated by an MVS operator or by CQS in any of the following situations:

- While at least one CQS is running, an MVS operator can initiate a structure rebuild in order to copy or recover queues by using the following command:  
`SETXCF START,REBUILD,STRNAME=strname,LOCATION=NORMAL|OTHER`
- CQS initiates a structure rebuild if, during CQS initialization, it detects an empty structure and a valid SRDS (indicating a valid structure checkpoint in the SRDS).
- CQS initiates a structure rebuild if your installation defines a rebuild threshold for loss of connectivity, and that threshold is reached. The rebuild threshold is defined in the CFRM policy and the Sysplex failure management policy.

### Copying Structures

If at least one CQS has access to the structure when structure rebuild is initiated, one of the CQS systems that still have access to the structure copies all of the messages (both recoverable and irrecoverable) from that structure to the new structure.

### Recovering Structures

If no CQS has access to the structure when structure rebuild is initiated, the structure is recovered from the SRDS and the MVS log. Irrecoverable messages (such as Fast Path input messages) are lost. Messages are read from the SRDS and copied into a new structure. CQS then reads the log to bring the structure back to the point of currency.

Using the `WAITRBLD` IMS startup parameter, a client can specify whether work can be performed while the EMHQ structure is being recovered.

## Deleting Message Queues

You can delete message queues on a structure by deleting the structure. You can only delete a structure if no CQS is connected to it. To delete a structure and its queues:

1. Shutdown all CQSs connected to the structure.
2. Enter the `SETXCF FORCE,STRUCTURE,STRNAME=strname` command.

Ensure that the *strname* in this command is the same as the *strname* specified in the CQS global structure definition PROCLIB member and the CQS local structure definition PROCLIB member.

### CQS Logging and the MVS Logger

The MVS system logger records all information necessary for CQS to recover structures and restart. CQS writes log records for each coupling facility list structure pair that it uses to a separate log stream. The log stream is shared among all CQS address spaces that share the structure. The MVS system logger provides a merged log for all CQS address spaces that are sharing queues on a coupling facility list structure.

For CQS to use an MVS system log, you must first define the log stream and associated resources to MVS. CQS also provides a File Select and Formatting Print utility to read the log records.

**Related Reading:** For more information on defining the log stream, see *OS/390 MVS Programming: Sysplex Services Guide*.

For more information on the File Select and Formatting Print utility, see *IMS/ESA Customization Guide*.

### Shutting Down CQS

You can issue the /CHE FREEZE command to shut down CQS. However, in this situation, if any in-flight work is existing, CQS does not shut down.

**Recommendation:** When in-flight work exists and no clients are connected to CQS, use the MVS STOP command to shut down CQS.

**Related Reading:** For more information on the CQSSHUT request, see *IMS/ESA Common Queue Server Guide and Reference*.

---

## Tuning for Performance in a Shared-Queues Environment

The structures and parameters you define for a shared-queues environment determine how the system performs.

### Tuning Structures and Parameters

To optimize system performance, try adjusting the following structures and parameters:

- The size of the IMS execution parameters:
  - LGMSGSZ=
  - QBUF=
  - QBUFHITH=
  - QBUFLWTH=
  - QBUFMAX=
  - QBUFPCTX=
  - QBUFSZ=
  - SHMSGSZ=
- The value of the CQS local structure definition parameter SYSCHKPT=
- The values of the CQS global structure definition parameters:

## Tuning for Performance in a Shared-Queues Environment

OVFLWMAX=

STRMIN=

- The size of the shared-queues structures on the coupling facility
- The size of the MVS system log structure on the coupling facility
- The number of MVS system log data sets that back up the MVS system log structure
- The frequency of structure checkpoints

## Recommendation for Fast Path Transactions

For Fast Path transactions, if you specify LOCALONLY in the Fast Path Input Edit/Routing exit routine (DBFHAGU0), do not change the size of your EMH pools. The same buffer pool space is required for a shared-queues environment as for a non-shared-queues environment. Specify LOCALFIRST in the Fast Path Input Edit/Routing exit routine (DBFHAGU0). You can then analyze system usage, and reduce the EMH pool space that is required.

**Related Reading:** For more information on tuning for performance in a shared-queues environment, see:

- *IMS/ESA Common Queue Server Guide and Reference*
- *IMS/ESA Administration Guide: System*

---

## Chapter 5. Editing and Formatting Messages

IMS uses two methods to edit and format messages to and from terminals: Message Format Service (MFS) and basic edit routines. IMS also provides routines to edit:

- Input and output from a terminal
- Transaction codes
- Input message fields
- Input message segments
- Message switching

This chapter presents an overview of the advantages of MFS, introduces MFS control blocks for message formatting, and summarizes the characteristics of the different devices MFS supports and the responsibilities of the MFS administrator.

**Restriction:** MFS does not support LU 6.2 devices. DFSLUEE0, an Edit exit routine, is provided for both input and output messages from LU 6.2 devices when the implicit API support is used.

**Related Reading:** For guidelines in coding the Edit exit routine, see *IMS/ESA Customization Guide*.

### **In this Chapter:**

- “Message Format Service”
- “Creating MFS Formats with SDF II” on page 99
- “Basic Edit” on page 101
- “IMS Editing for Intersystem Communication (ISC)” on page 102
- “Transparency Option” on page 103
- “Unprotected Screen Option” on page 104
- “Bypassing MFS Editing” on page 104
- “IMS Sensitivity to Nongraphic Message Data” on page 104
- “Controlling Output Devices” on page 106
- “Small Buffer Devices” on page 106
- “Controlling Output” on page 107

---

## Message Format Service

Message Format Service (MFS) is an IMS facility that formats messages to and from terminals, so that IMS application programs need not deal with device-specific characteristics in input or output messages.

MFS formats messages to and from user-written programs in remote controllers and subsystems, so that host application programs need not deal with terminal-specific characteristics of the remote controller.

MFS uses control blocks that the user specifies to indicate to IMS how input and output messages are arranged.

- For input messages, MFS control blocks define how the message that is sent by the device to the application program is arranged in the program’s I/O area.

## Message Format Service

- For output messages, MFS control blocks define how the message that is sent by the application program to the device is arranged on the screen or at the printer. Data, such as literals that appear on the screen but not in the program's I/O area, can also be defined.

In IMS systems, data that is passed between the application program and terminals or remote programs can be edited by MFS or basic edit. The facilities provided by MFS depend on the type of terminals or secondary logical units (SLUs) your network uses.

MFS allows application programmers to deal with logical messages instead of device-dependent data; this simplifies application development. The same application program can deal with different device types using a single set of logic, whereas device input and output are varied for a specific device type. The presentation of data on the device or operator input can be changed without changing the application program. Full paging capability is provided by IMS for display devices. Input messages are created from multiple screens of data.

A program using MFS need not be designed for the physical characteristics of the device that is used for input and output messages unless it uses very specific device features. Even when these features are used, the program can request that MFS assist in their presentation to the program or the device.

MFS supports SLU-type devices SLU-1, SLU-2, SLU-P, Finance, and LU 6.1. MFS also supports older devices, including IBM 2740, 2741, 3270, and 3600.

For IBM 3270 or SLU-2 devices, device control characters or orders can be sent directly from or received by the program using the MFS bypass function. This gives the application program more direct control of the data stream. The program uses reserved format names that cause MFS to bypass the edit of:

- the output message
- the next input message that is received from the display terminal

Both logical- and physical-paging facilities are provided for the IBM 3270 and 3604 display stations; these facilities allow the application program to write large quantities of data that MFS can divide into multiple display screens on the terminal. The terminal operator has the capability to page forward and backward to different screens within the message.

## MFS Components

MFS has several components:

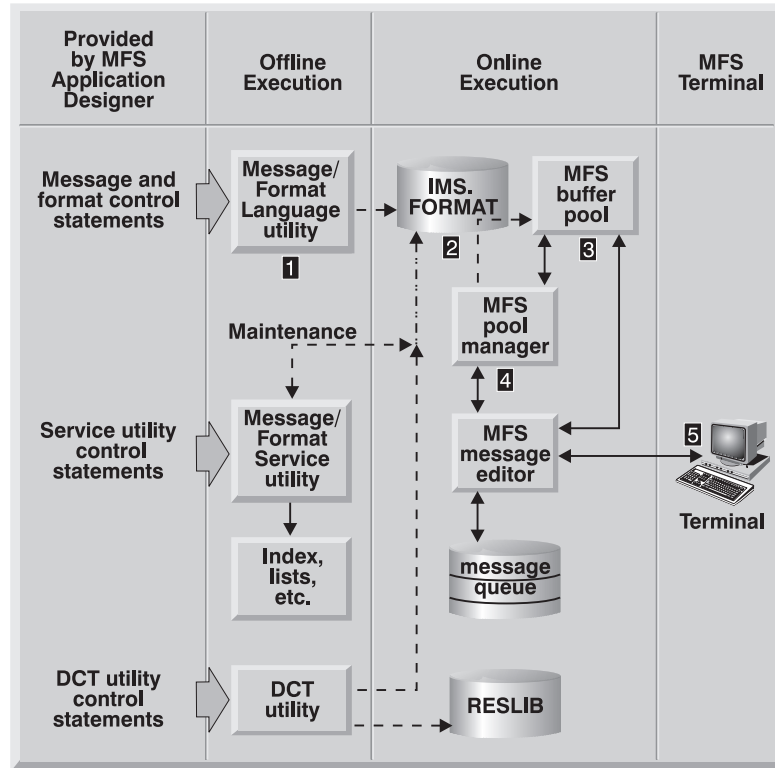
- MFS language utility
- Message editor
- MFS pool manager
- MFS service utility
- MFSTEST pool manager
- MFSDCT utility (DFSUTB00)

The MFS language utility is executed offline to generate control blocks and place them in a format control block data set named IMS.FORMAT. The control blocks describe the message formatting that is to take place during message input or output operations. They are generated according to a set of utility control statements specified by the MFS application program designer.

## Message Format Service

Figure 14 shows an overview of major MFS operations. The numbers refer to notes that indicate major distinctions in MFS processing when the MFSTEST facility is used.

**Related Reading:** For more information on MFS support, see *IMS/ESA Application Programming: Transaction Manager*.



### Note: MFS Test Distinctions

- 1** Can execute concurrently with the IMS online control region only in MFSTEST mode.
- 2** Replaced by IMS.TFORMAT in MFSTEST mode; IMS.FORMAT is available as a secondary source of control blocks in MFSTEST.
- 3** The communication-line buffer pool is used in MFSTEST mode.
- 4** Replaced by MFSTEST pool manager in MFSTEST mode.
- 5** Terminal operator must use /TEST MFS command in order to enter MFSTEST mode.

Figure 14. Overview of Message Format Service

The message editor and MFS pool manager operate online during the normal production mode of operation. The message editor performs the actual message formatting operations using the control block specifications.

Two other MFS components, an MFS service utility and an MFSTEST pool manager, are available to support optional MFS operations.

## Message Format Service

The MFS service utility provides a method for additional control of the format control block data sets. It executes offline and is able to create and maintain an index of control blocks for online use by the MFS pool manager.

The MFSTEST pool manager replaces the MFS pool manager in order to support the optional MFSTEST mode of operation. The /TEST command with the MFS keyword places a logical terminal into MFSTEST mode. For each terminal in MFSTEST mode, combining temporary format blocks with the use of other blocks that are already in production mode allows new applications and modifications to existing applications to be tested without disrupting production activity.

## Administering MFS

To take full advantage of the flexible message formatting options offered by MFS and to ensure efficient MFS operation, an MFS administrator should be appointed. The MFS administrator should be responsible for MFS implementation and administration and should coordinate MFS application design and programming for the installation.

The responsibilities of an MFS administrator include:

- Establishing procedures for the submission of MFS control block requests by application development personnel.
- Establishing procedures and controls for the application of changes to the IMS.TFORMAT library.
- Defining MFS control blocks most efficiently in keeping with the requirements of the specific application and the overall system.
- Minimizing device character transmission, sharing MFS control blocks, and ensuring the most efficient use of MFS without jeopardizing application requirements or operator considerations.
- Establishing and documenting operator guidelines and system standards for MFS. The many options that MFS offers can result in confusing practices, unless you establish and follow standard procedures. Be sure to standardize certain aspects of format design in order to minimize terminal operator training and error rates.
- Deciding if and how the optional index directory should be used and determining buffer pool requirements.
- Monitoring the use of the MFS control blocks and of the MFS buffer pool with the IMS /DISPLAY command and IMS Monitor report output, and modifying MFS parameters as required.
- Making end users aware of the operating characteristics of the different device types and terminal subsystems.
- Informing others about the differences between the various partition formats.
- Establishing and informing others about naming conventions and guidelines. In particular, the MFS administrator should be able to discuss naming conventions for the partition descriptor blocks and the sizes of the display screen, the viewports, and the display characters.
- Communicating information on conventions for and restrictions on MFS formats.
- Defining screen sizes and feature combinations that are not included in the IMS stage-1 system definition.
- Creating the MFS device characteristics table control statements for processing by the MFSDCT utility (DFSUTB00). The MFS device characteristics table entries and default format control blocks are used for ETO terminals.



## Message Format Service

- Defining input message field edit routines and segment edit routines. MFS and all MFS-supported devices are able to use message edit routines. You can use these edit routines for such common editing functions as numeric validation or conversion of blanks to zeros.

IMS provides a sample of both a field edit and a segment edit routine.

**Related Reading:** For information on the timing within the MFS editing processing when these routines are activated and the coding requirements, see *IMS/ESA Application Programming: Transaction Manager*.

The MFS administrator should be technically competent in all aspects of IMS relative to MFS:

- Online transaction processing
- IMS API for message processing
- Operation with remote controllers
- MFS implementation, device characteristics, and capabilities
- Interpretation of MFS statistics and related IMS Monitor report output

The administrator should also be familiar with the hardware and remote programs for SLU-P, Finance remote programs, or ISC subsystems if such programs are going to operate with MFS by using distributed presentation management.

In addition, because one administrative responsibility is minimizing device character transmission, the administrator should be familiar with the terminal hardware characteristics.

An MFS administrator must communicate with IMS system administrators and application developers, as well as programmable workstation developers and end users. The administrator must be able to enforce installation standards and to modify application specifications for MFS control blocks when necessary to benefit overall system performance. The procedures of related programming groups should recognize this authority of the MFS administrator.

## Advantages to Using MFS

Two primary advantages to using MFS are that it:

- Simplifies the development and maintenance of terminal-oriented application systems
- Improves online performance

To simplify IMS application development and maintenance, MFS performs many common application program functions and gives application programs a high degree of independence from specific devices or remote programs.

With the device independence offered by MFS, one application program can process data to and from multiple device types while still taking advantage of their different capabilities. Thus, MFS can eliminate or minimize the changes in application programs when new terminal types are added.

MFS makes it possible for an application program to communicate with different types of terminals without having to change the way it reads and builds messages. When the application program receives a message from a terminal, how the message appears in the program's I/O area is independent of the kind of terminal that sent it; the appearance depends on the MFS options specified for that program. If the next message that the application program receives is from a different type of

## Message Format Service

terminal, the user does not need to do anything to the application program. MFS shields the application program from the physical device that is sending the message in the same way that a database program control block (PCB) shields a program from the data in the database and how it is stored.

Other common functions MFS performs include left or right justification of data, padding, exit routines for validity checking, time and date stamping, page and message numbering, and data sequencing and segmenting. When MFS performs these functions, the application program handles only the actual processing of the message data.

**Related Reading:** For information on user-written exit routines and how to use them, see *IMS/ESA Customization Guide*.

Figure 15 shows how MFS can make an application program device-independent by formatting input data from the device or remote program for presentation to IMS, and by formatting the application program data for presentation to the output device or remote program.

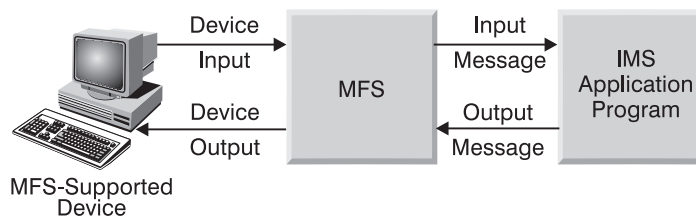


Figure 15. Message Formatting Using MFS

MFS also improves online performance of a terminal-oriented IMS system by using control blocks that are designed for online processing. The MFS control blocks are compiled offline, when IMS is not being executed, from source language definitions. MFS can check their validity and make many decisions offline to reduce online processing. In addition, during online processing, MFS uses look-aside buffering of the MFS control blocks in order to reduce CPU usage and the channel costs of input/output activity.

Because MFS control blocks are reentrant and can be used for multiple applications, online storage requirements are reduced. Optional main-storage indexing and anticipatory fetching of the control blocks can also reduce response time. IMS gains additional performance improvements, because multiple I/O operations can execute concurrently in loading the format blocks from the MFS format library.

In addition, MFS uses MVS paging services to reduce page faults by the IMS control region task.

Finally, MFS can reduce use of communication lines. Compressing and transmitting only the required data reduces line load and improves both response time and device performance.

## MFS Control Blocks

Users can specify four types of MFS control blocks to format input and output for the application program and the terminal or remote program.

### Definitions:

- *Message Output Descriptors (MODs)* define the layout of messages that MFS receives from the application program.
- *Device Output Formats (DOFs)* describe how MFS formats messages for each of the devices with which the program communicates.
- *Device Input Formats (DIFs)* describe the formats of messages MFS receives from each of the devices with which the program communicates.
- *Message Input Descriptors (MIDs)* describe how MFS formats messages so that the application program can process them.
- *Message descriptors* are both MIDs and MODs.
- *Device formats* are both DIFs and DOFs.

Because each MOD, DOF, DIF, and MID deals with a specific message, both a MOD and DOF must exist for each unique message a program sends, and both a DIF and MID must exist for each unique message a program receives.

## Overview of MFS Components and Operation

MFS has the following components:

- The MFS language utility, which generates control blocks from user-written control statements and places them in a library called IMS.FORMAT
- The MFS service utility, which is used for maintenance of the control blocks in IMS.FORMAT
- The MFS message editor, which formats messages according to the control block specifications generated by the language utility
- The MFS pool manager, which keeps the MFS control blocks that are required by the message editor in the main-storage MFS buffer pool
- The MFSTEST Pool Manager, which replaces the MFS pool manager when the language utility is being used in test mode

The IMS Online Change utility also plays an important part in updating the MFS libraries, even though it is not part of MFS. Briefly, online change allows the control block libraries to be modified while the IMS control region is executing.

**Related Reading:** For a complete description of online change, see *IMS/ESA Administration Guide: System*.

---

## Creating MFS Formats with SDF II

**Definition:** *SDF II* is an interactive tool for designing and generating MFS formats.

SDF II does not replace MFS, but it does make developing and maintaining MFS formats easier. Because SDF II uses a panel editor for designing and testing formats, it frees the MFS programmer from some of the tasks associated with coding MFS source statements.

With SDF II, application programmers and analysts who might not know the special requirements of MFS can perform a part of the programming job that would otherwise call for specialized knowledge. SDF II uses a panel editor, such as the one shown in Figure 16 on page 100, to define and test a panel.

## Creating MFS Formats with SDF II

```
DEFINE FORMAT
Format . . . . .Positions 1-75 or 80, Lines 1-24 of 24
Marks: V - C . L , S +                               Contents: FORMAT
001
002             *****
003             ** E M P L O Y E E   P A Y R O L L **
004             *****
005
006 LAST NAME:                               FIRST NAME:
007
008 EMPL NO:
009
010 SOC SEC NO:
011
012 RATE OF PAY:
013
014
015
016 INPUT:
017
018
019
020
021
022
023
024
PF1=HELP    2=SPLIT    3=END    4=RETURN    5=RFIND    6=CHANGE
PF7=UP      8=DOWN    9=SWAP   10=LEFT   11=RIGHT   12=CURSOR
```

Figure 16. Example of SDF II Panel Editor

After testing the panel, the SDF II user can automatically generate the MFS source code, shown in Figure 17 on page 101, that is needed for the format definition. Using MFS only, the programmer would need to code these statements manually.

```

DOF
PAYF      FMT
          DEV      TYPE=(3270,2),FEAT=IGNORE,DSCA=X'00A0'
          DIV      TYPE=INOUT
          DPAGE    CURSOR=((5,15))
          DFLD     '*****',POS=(1,21)
          DFLD     '* EMPLOYEE PAYROLL *',POS=(2,21)
          DFLD     '*****',POS=(3,21)
          DFLD     'LAST NAME:',POS=(5,2)
LNAME     DFLD     POS=(5,15),LTH=16
          DFLD     'FIRST NAME:',POS=(5,36)
FNAME     DFLD     POS=(5,48),LTH=16
          DFLD     'EMPL. NO:',POS=(7,2)
EMPNO     DFLD     POS=(7,11),LTH=6
          DFLD     'SOC SEC NO:',POS=(9,2)
SSN       DFLD     POS=(9,14),LTH=11
          DFLD     'RATE OF PAY: $',POS=(11,2)
RATE      DFLD     POS=(11,16),LTH=9
          DFLD     'INPUT:',POS=(16,2)
INPUT     DFLD     POS=(16,9),LTH=30
          FMTEND

MID
PAYIN     MSG      TYPE=INPUT,SOR=(PAYF,IGNORE)
          SEG
          MFLD     'PAYUP ' SUPPLIES TRancode
          MFLD     LNAME,LTH=16
          MFLD     FNAME,LTH=16
          MFLD     EMPNO,LTH=6
          MFLD     SSN,LTH=11
          MFLD     RATE,LTH=9
          MFLD     INPUT,LTH=30,JUST=R,FILL=C'0'
          MSGEND

MOD
PAYDAY    MSG      TYPE=OUTPUT,SOR=(PAYF,IGNORE)
          SEG
          MFLD     LNAME,LTH=16
          MFLD     FNAME,LTH=16
          MFLD     EMPNO,LTH=6
          MFLD     SSN,LTH=11
          MFLD     RATE,LTH=9
          MFLD     INPUT,LTH=30,JUST=R,FILL=C'0'
          MSGEND
    
```

Figure 17. MFS Source Statements

SDF II is designed for use on a 3270–display device; however, SDF II can create formats for all devices supported by MFS.

**Related Reading:** For more information on SDF II, see *Screen Definition Facility II General Information*.

---

## Basic Edit

If you do not use MFS, an IMS function called basic edit performs message editing.

For input messages, basic edit:

- Translates messages to uppercase, if specified by the EDIT=UC parameter on the system definition TRANSACT macro.
- Removes leading control characters from the first segment of each message. Leading blanks are also removed from the first segment if the message is not the continuation of a conversation or a message from a terminal in preset mode.

## Basic Edit

- Removes leading control characters from all subsequent message segments, if the message type is a transaction or a command (except the /BROADCAST command).
- Removes line control characters from all segments.
- Removes trailing carriage return and end-of-block characters from all segments of a transaction.
- Eliminates backspaces, on a one-for-one basis, from all segments when the entering or transmission of backspaces is a normal correction procedure on the entering terminal.
- Removes the password and replaces it with a blank when necessary to provide separation between the transaction code, logical terminal, or command verb, and data that follows.
- Inserts, in front of data that is entered in the first segment of a message, the transaction code or logical terminal name defined by the prior /SET command. A blank is inserted following the transaction code, if it is necessary to obtain separation between the inserted transaction code and the entered data.
- Adds a nonconversational transaction code to the first segment of the next input message, if a terminal is in conversation mode and the application terminates the conversation by inserting a nonconversational transaction code into the SPA.  
**Related Reading:** For more information on conversational transactions, see *IMS/ESA Application Programming: Transaction Manager*.
- Removes the function management header (FMH), if any, that appears at the beginning of the first transmission of a chain for VTAM-supported devices.
- Deblocks message segments at each interrecord separator (IRS) control character, and discards the IRS control character for input from a SLU-1 card reader, a transmit data set (TDS), or a user data set (UDS).
- Deblocks message segments at each new line or forms-feed control character if the optional MFS editing is not selected for SLU-1 consoles. This character can be discarded, depending on the criteria previously described.
- Treats the presence of a TRN (X'35') character immediately followed by a length in the next byte as transparent data.

For output messages, basic edit:

- Changes nongraphic characters in the output message before the data goes to the device.
- Inserts any necessary idle characters after new-line, line-feed, and tab characters.
- Adds line control characters for the operation of the communication line.

For basic edit support of SLU-1 transparent data, basic edit does not alter or delete characters following the destination and password fields if transparent processing has been requested. Indicate transparent processing by specifying:

- BINPDSB1=BINTRNDS on the bind image for VTAM-type SLU-1 terminals
- Edit option BASIC=TRN on the COMPT1 parameter of the IMS TERMINAL macro or ETO descriptors for SLU-1 terminals

---

## IMS Editing for Intersystem Communication (ISC)

IMS recognizes several options on a message-by-message basis when communicating with ISC. IMS recognizes the following options:

- Use basic edit.

## IMS Editing for Intersystem Communication (ISC)

The destination process name (DPN) is BASICEDT, which is a reserved IMS name.

- Treat the data as transparent data.

The IMS preset destination mode, which is established by the /SET command, is used. This option is used if a /SET command is in effect for a given session and a primary resource name (PRN) is not specified on the ATTACH or SCHEDULER FMH. While this option is in effect, IMS cannot recognize input commands within this input data stream.

- Do not edit data that follows the transaction code and input password fields.

The input destination and security checking is the same as in basic edit. A / as the first character indicates an IMS command and causes the IMS preset destination mode to be bypassed or terminated. No additional editing is done on the input message following the input transaction code or LTERM name and optional password field. This is the default input edit for ISC under each of the following conditions:

ATTACH or SCHEDULER FMH is not included in input

ATTACH or SCHEDULER FMH does not specify a DPN

ATTACH or SCHEDULER FMH specifies a DPN but not a PRN

- Do nothing.

IMS uses the input PRN ATTACH parameter as the transaction code or LTERM name for the duration of the single input message. Subsequently, the original IMS preset value can be used again. Basic edit treats the input as transparent with password security already checked.

The chosen option depends on the destination process name (DPN) and the data received by IMS. The DPN is specified on the ATTACH or SCHEDULER FMH.

If a DPN of ISCEDT or its alias is specified the second through fourth options are chosen. The alias for ISCEDT is defined using the EDTNAME keyword on the COMM macro. ISCEDT is also a reserved IMS name.

**Related Reading:** For more information on using basic edit for ISC, see *IMS/ESA Customization Guide*.

---

## Transparency Option

Transparent data includes programmed symbols, extended field attributes, and commands. These unprintable characters might appear on the 3270 screen as hyphens or another representation for unprintable characters.

**Related Reading:** For more information on the extended field attributes, see *IMS/ESA Application Programming: Transaction Manager*.

If IMS attempts to send transparent data to an IBM 3270 terminal that is unable to receive the data, an error is returned to IMS, and the terminal is taken out of service. To restart the terminal, the master terminal operator must issue the /START command.

For LU 6.1, input is processed using the transparency option of basic edit if the primary resource name (PRN) in the function management header (FMH) contains a transaction. Otherwise, the data stream is only scanned for the transaction code and optional password. The remaining data is treated as transparent data.

### Unprotected Screen Option

When the screen is in unprotected status, IMS can send output to the terminal at any time without requiring input from the terminal. This option can be used on a terminal-by-terminal or message-by-message basis.

---

### Bypassing MFS Editing

This section describes how to bypass MFS formatting for output messages.

#### When To Bypass MFS Editing

IMS enables an IMS application program to bypass MFS formatting of an output message destined to an IBM 3270 or to SLU-2 devices. This bypass is intended for subsystems or installation support programs that execute under IMS.

**Recommendation:** Do not use this bypass for IMS application programs, because the application program loses the productivity, device independence, and migration benefits associated with MFS.

When MFS is bypassed on output, the application program constructs the entire 3270 data stream, beginning with the command code and ending with the last data byte. The user might want to bypass the MFS edit function to allow application programs to receive the 3270 input data stream from the terminal without IMS editing.

#### Locking and Unlocking the Terminal Keyboard

When the application uses MFS bypass and the reserved format name DFS.EDTN, IMS enables the application program to lock or unlock the terminal keyboard. If the lock option is specified, the program is responsible for unlocking the keyboard after processing is completed. If the unlock option is specified, IMS controls the locking and unlocking of the keyboard.

**Related Reading:** For information on how to bypass MFS editing, see “Application Programming Using MFS” in *IMS/ESA Application Programming: Transaction Manager*.

---

#### IMS Sensitivity to Nongraphic Message Data

This section describes the sensitivity IMS has to specific characters when users attempt to send and receive nongraphic data in IMS messages.

#### Output Message Segment Editing

For output message segments that MFS edits, only graphic data (X'40' through X'FE') is contained in the output message that is presented to the device. Nongraphic characters, if present in the output message, are changed by MFS before the data is presented to the device. Device control characters HT, CR, LF, NL, and BS are changed to X'00' for 3270 data streams. For all other device types, all nongraphic characters are also changed to blanks.

If the Distributed Presentation Management (DPM) option of MFS is used for IBM 3600 controllers, the user can specify GRAPHIC=NO in the SEG statement. Nongraphic characters, if present in the output segment with GRAPHIC=NO specified, are presented unchanged to the remote program.



## IMS Sensitivity to Nongraphic Message Data

For programmable workstations that are supported through VTAM, IMS can insert function management headers (FMHs) and can perform additional editing for device control sequences when splitting a single IMS segment into multiple transmissions.

### Editing of Input Message Segments by MFS

If MFS is defined for a device, you should be aware of the following:

- Specify GRAPHIC=NO in the SEG statement to prevent uppercase translation on a segment if the destination requests it with the EDIT=UC specification on the system definition TRANSACT macro.
- If the first input record is from a 274x, 3600, SCS1, or SCS2 device, or from DPM-An, the segment is discarded if the final characters of the segment are:
  - Two asterisks (\*\*)
  - Two asterisks followed by NL (\*\*X'15')
  - Two asterisks followed by IRS (\*\*X'1E')
- The presence of two slashes (//) at the beginning of a message segment is considered an escape sequence.
- If the card feature is defined for an SCS1 device (with the CARD=operand in the DEV statement), all control characters are removed from magnetic card input before the data is presented to the input MFLD.

The definition of the MFS delete characters (LDEL=operand in the DEV statement) and field tab character (FTAB= operand in the DEV statement) for MFS-supported devices, excluding IBM 3270, can direct the editing of input message segments.

If the input is processed by MFS, the editing performed is dependent on the descriptions provided through the Message Format Service language utility. As input segments from the device might have no relationship to input message segments after MFS editing, the input segment from the device is not available to user-written edit routines. Input message segments after MFS editing are available to user-written edit routines.

**Related Reading:** For a description of MFS editing, See "Input Message Formatting" in "Message Formatting Functions" in *IMS/ESA Application Programming: Transaction Manager*.

### Editing of Input Message Segments by Basic Edit

The following editing is done if basic edit is used for nongraphic message data:

- For the first segment of an input message when a terminal is not in conversation mode, leading characters less than X'41' are removed. For other than the first segment or when a terminal is in conversation mode, leading characters less than X'40' are removed.
- If a terminal is not in conversation or preset mode, a left parenthesis within the first nine positions of the first segment indicates the presence of a password. The left and right parentheses and the password are removed, and the segment is compressed.
- For non-SNA devices, the X'26' character that appears as the final character in a segment is removed.
- Two asterisks (\*\*) or two asterisks followed by NL (X'15') that appear as the final characters of a segment cause the entire segment to be discarded.

## IMS Sensitivity to Nongraphic Message Data

- For unbuffered keyboard devices (for example, IBM 2740-1 and 2741), backspace (X'16') characters are treated as character-delete indicators. Each backspace character and the preceding input character are removed from the segment.
- If the destination of the input message is a transaction, an NL (X'15') character appearing at the end of a segment is removed.
- If a device is in preset mode, the transaction code is added to the first segment.
- For input from IBM 3270 devices, the attention identifier and cursor address are removed, and all start buffer address sequences are changed to blanks.
- If the first character of any segment is a slash (/), the entire input message is treated as a command.
- If an input message is received from an NDS device, or if it is using Intersystem Communication, the data stream is handled wholly as transparent data or transparent except for edit of the transaction code and password.

---

## Controlling Output Devices

You can control an output device by using control characters in the second byte of the ZZ field of the message prefix (Z2). Follow these rules:

- SLU-1:  
The Z2 field, bit X'80', should be set if the segment contains structured field data. The Z2 field, bit X'40', should be set if the segment is the first segment of a new LPAGE series.
- Switched devices:  
**Exception:** IBM 3270  
You can use the Z2 field in the message format to request that the line be disconnected after the present message is sent. This field is ignored if the output is physically sent to a device without this capability. The disconnect request is indicated using X'80' as the Z2 field value, this request is recognized if present in any segment.
- IBM 3270 printer (3270P device type):  
You can use the Z2 field, bit value X'80', to specify the presence of command code and WCC character in the data stream, which is to be used when MFS editing is bypassed.
- Printer components:  
The program can embed, in the text portion of the message, carriage return characters or new line symbols. If the output is going to an IBM 2780 or local printer (SYSOUT), the first two characters of the message can be carrier control characters.

---

## Small Buffer Devices

Some terminal devices have hardware limitations of the maximum buffer size that they can process. Additional limits can be imposed by software, either in the device or in the network. Exceeding these limits can result in an error and failure to deliver a message that is too large for the device to process. Be aware of what these limits are, and of the alternatives available to your application program.

Different device types specify the limits in different ways. For input, the SEGSIZE or BUFSIZE specification in IMS is relevant, and, for output, the OUTBUF specification. For VTAM terminals, the BIND RU size is based on these values. IMS

supports message chaining for both input and output, but some terminals do not support chaining. In this case, message length is limited by sizes.

**Related Reading:** For more information on the SEGSIZE and BUFSIZE, see the description of the LINE and TERMINAL macros in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Your application program has control over the length of the output message, either directly or through the MFS definition being used to format the message. If you need to send a message that is longer than the maximum length supported by the target device, you must break the message into multiple messages, each of which is shorter than the maximum length for the device. The DL/I PURG call must be issued by your application program to do this. IMS delivers messages in the order that your application program inserts them; this is a good solution for devices that are not sensitive to message boundaries.

**Related Reading:** For information on the DL/I PURG call, see *IMS/ESA Application Programming: Transaction Manager*.

---

## Controlling Output

Most outputs of an IMS online system goes to the input terminal. You have several choices exist for creating printer output.

- Use local printing through hardware-specific device support.
- Code the application program so that it directs response to the printer component of the input terminal.
- Use the /ASSIGN command to assign the response LTERM to a printer component.
- Code the application program to send output (SYSOUT) to an LTERM representing an IMS system printer.
- Code the application program to send output to an LTERM representing an alternative device for offline printing.

The choice is dependent on the application design. The use of the /ASSIGN command and the control of spool data sets are choices that might involve master terminal operator intervention.

With ETO, the output is associated with the user. The user can move from terminal to terminal. The output follows the user. When the output is available, use an autologon facility to generate a session for output delivery to a specified terminal.

## Using a Printer Component

The printer component for a terminal is often not in continuous operation. It might need a remote operator to supervise the paper supplies or special forms. The /COMPT command is used to make the device *ready* or *not ready* if the remote operator is a VTAM component. The operator can use the /ASSIGN command in order to direct output to a particular component.

**Related Reading:** For details on the /ASSIGN command, see *IMS/ESA Operator's Reference*.

## Controlling Output

### Spooled Output Control

You need to be aware of requirements for IMS spooled output. This is because the output might be needed immediately by the end user or additional output can be impeded if allocated space is exhausted.

Given a spool line group, you need to be aware of how many data sets are used for output. A recommended definition is at least two data sets, one data set printing while the other receives additional input. The line number and PTERM references are also required.

At any time, you can use the following command to call for the spooled output to be scheduled to a printer:

```
/STOP LINE n PTERM nn
```

The action of this command is to close the data set and direct additional output to the next spool data set. If this is the last in the set, output goes to the first. If only one output data set exists at the time the print utility is scheduled, all messages for the LTERM are queued. This could rapidly add to the contents of the message queues. Message queues can fill very quickly, specially when the spool is the secondary master console. The IMSWTnnn procedure to execute the print utility is scheduled automatically by the data-set-full condition. A write error for the data set also starts the procedure.

The printing procedure is tailored to the line group data sets by system definition. You can invoke this procedure at any time by using a /START REGION IMSWTnnn command. If the data sets are not closed, current line output is queued.

Restart can affect spooled output handling. If output data sets were not printed from a prior execution, the /NRESTART command prevents additional output messages. The action of a /START LINE automatically schedules the IMSWTnnn procedure and frees the data set for output from the current system execution.

## Using Printer Components of the IBM 3270 Information Display System

You can define printer components to be part of an IBM 3270 Information Display System. This allows for printed copy of the video output (or input) to be sent to the printer's component.

This support covers the following printer component devices: IBM 3284, 3286, 3287, 3288, or 3289.

Your system definition input indicates whether these are to be connected through a polled-BSC or an SDLC line. The printed copy can be automatically produced or be operator controlled. The input message definition or the control prefix of the output message can cause this automatic printing. The component must be attached to the same 3270 control unit as the display workstation containing the information to be copied—an IBM 3271/3274 or a 3275/3276.

**Restriction:** A locally attached 3270 does not support the copy function.

## Specifying Candidate Printers

With BSC 3270, using VTAM, requests for copying the screen content to a printer are directed to the first available printer on the same control unit as the screen.

**Definition:** *First available printer* refers to the sequence of TERMINAL macros for the devices. A predetermined sequence controls the order in which the printer components are selected for message output. Figure 18 shows two display workstation printer groups.

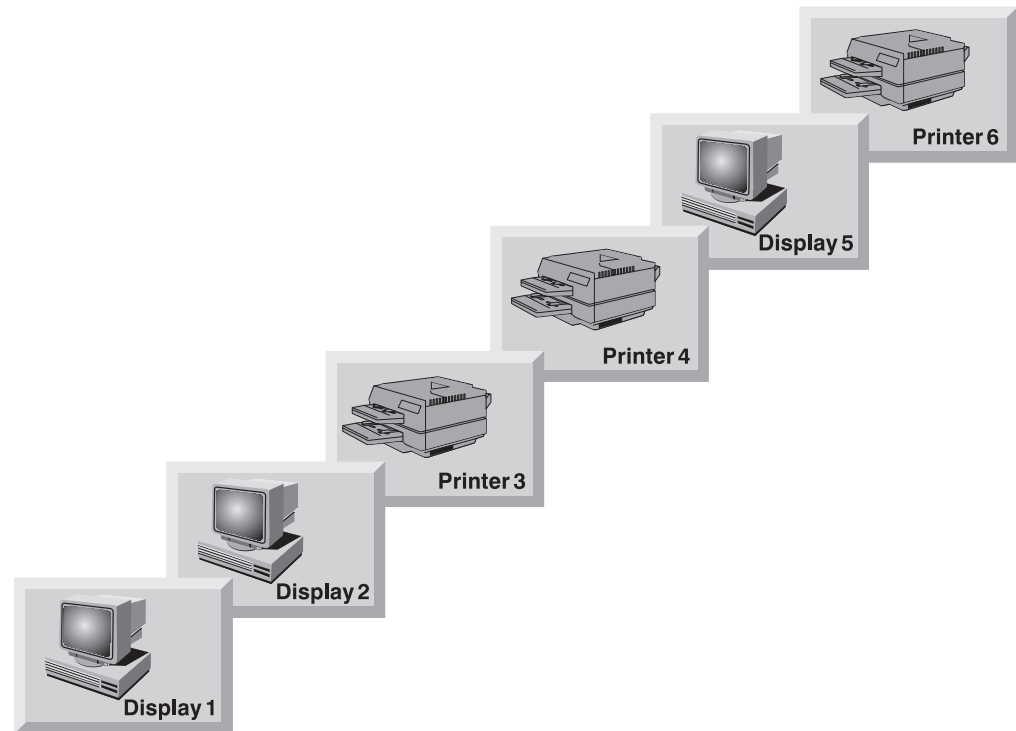


Figure 18. Sequence of IMS TERMINAL Macros

The first printer defined after a display is a candidate printer for that display. Any subsequently defined printer is also a candidate if no display is defined between it and the first candidate. In Figure 18, if a request is made for a printed copy of Display 1 or Display 2, IMS sends a 3270 copy command to Printer 3, if it is available. If Printer 3 is not available, IMS considers Printer 4. Printer 6 is not a candidate for Display 1 or Display 2, but it is a candidate for Display 5.

The copy function is not permitted to cross 3270 control-unit boundaries. For IBM 3274 and 3276 with SNA protocols, IMS sends the print request to the display, and the printer is selected by the controller. The printer need not be defined to IMS.

**Restriction:** The 3270R (non-SNA) copy function is not supported for ETO terminals.

## Operational Considerations

When a request for a copy operation is sent by the operator or an application program, the first available candidate printer is used for output. The search order corresponds to the system definition sequence. If a printer is stopped, already printing a message, in exclusive status, or not ready, the next candidate is chosen. If a copy request by an operator finds all printers busy, the keyboard is locked until a printer becomes available. Output messages that require the copy function are not sent if a candidate printer is unavailable. The display workstation receives an error message. A retry attempt is made for the message when the error message is cleared from the screen using the message advance function (PA2 key). The format

## Controlling Output

description for a message can also specify a copy action. MFS uses the DSCA operand in the DEV statement to specify a copy action.

**Related Reading:** For more information on the DSCA operand of the DEV statement, see *IMS/ESA Application Programming: Transaction Manager*.

## Sharing Printers between Systems

If you plan to share the use of a printer other than the master, IMS allows an online IMS system and another subsystem (possibly another IMS system) to alternate their use of the printer. The IMS support is for VTAM 3270 printers and, in the SNA environment, for LU-1 and LU-4 printers. The /OPNDST NODE Q command simulates a logon to IMS for the requested printer. The printer is automatically acquired when the owning subsystem frees the printer. You can queue output at any time independently of printer availability.

The system definition requirements are for the OPTIONS keyword of the TERMINAL macro. Specify:

<b>OPNDST</b>	so that the /OPNDST command is valid for this terminal
<b>SHARE</b>	so that the VTAM macro SIMLOGON can be issued by IMS in order to acquire the printer when data is queued for output
<b>RELRQ</b>	so that IMS releases the terminal to other VTAM subsystems upon their request

Specify RELRQ on the ETO logon descriptors to have the same features available for ETO terminals.

Each output message that is queued to a shared printer attempts a session.

As an alternative to immediate logon simulation and repeated failures while trying to acquire messages, you can use the Shared Printer exit routine (DFSSIMLO). This exit routine monitors terminal (or transaction) status, and takes one of three actions:

- Ignores the request
- Simulates the /OPNDST command
- Queues a transaction for an automated operator program

The Shared Printer exit routine is not called if the printer is connected. You specify the SIMEXIT keyword on the COMM macro to call the Shared Printer exit routine. Your installation must write the Shared Printer exit routine and add it to the library of user routines prior to stage-2 of system definition.

**Recommendation:** Do not share the secondary master terminal.

**Related Reading:** For more information on the Shared Printer exit routine, see *IMS/ESA Customization Guide*.

For more information on different ways in which multiple users can share output terminals, see "Sharing Printers Using ETO" on page 158.

---

## Part 2. Administering the Extended Terminal Option

<b>Chapter 6. Overview of the Extended Terminal Option . . . . .</b>	<b>115</b>
Benefits of Using ETO . . . . .	115
ETO Terminology . . . . .	116
Terminals . . . . .	116
Dynamic Users . . . . .	116
Structures . . . . .	116
Terminal Structures . . . . .	116
User Structures . . . . .	116
Descriptors . . . . .	117
Logon Descriptor . . . . .	118
User Descriptor . . . . .	118
MSC Descriptor . . . . .	118
MFS Device Descriptor . . . . .	119
ETO Concepts . . . . .	119
When Structures Are Created and Deleted . . . . .	119
Descriptors and Exit Routines . . . . .	119
How Descriptors Are Created and Used . . . . .	120
How Control Blocks Are Created and Deleted . . . . .	120
Summary of ETO Implementation . . . . .	121
<b>Chapter 7. Administering the Extended Terminal Option . . . . .</b>	<b>123</b>
Planning for ETO . . . . .	124
Identifying Your Requirements . . . . .	124
Restrictions on ETO . . . . .	125
Defining Physical Terminals . . . . .	125
Planning for Both Static and Dynamic Terminals . . . . .	127
Defining Terminals for Growth . . . . .	128
Identifying VTAM Device Types, Screen Sizes, and Models . . . . .	128
Defining Device Types . . . . .	128
LU2 and Non-SNA 3270 Screen Size and Model Information . . . . .	130
Planning a High-Security Environment with ETO . . . . .	134
Static versus Dynamic Terminals . . . . .	134
Using RACF versus SMU for Security . . . . .	134
Planning for MFS . . . . .	135
Planning User IDs . . . . .	135
Planning User Queue Names . . . . .	135
Planning Operations . . . . .	136
Planning for MSC Support with ETO . . . . .	136
Coding ETO Descriptors . . . . .	136
Creating Descriptors Using the System Definition Process . . . . .	137
Storing Descriptors . . . . .	137
Creating Logon Descriptors . . . . .	138
Logon Descriptor Format . . . . .	138
Creating Logon Descriptors at System Definition . . . . .	139
Criteria for Selecting Logon Descriptors . . . . .	139
Criteria for Selecting a Default Logon Descriptor . . . . .	140
Using NTO, 3600/Finance Terminals . . . . .	141
Recovering ETO Terminals Using XRF . . . . .	141
Creating User Descriptors . . . . .	142
User Descriptor Format . . . . .	142
Creating User Descriptors at System Definition . . . . .	142
Criteria for Selecting User Descriptors . . . . .	143
Using Installation-Created User Descriptors . . . . .	143

Using Node User Descriptors . . . . .	143
Using DFSUSER User Descriptors . . . . .	144
Creating MFS Device Descriptors . . . . .	145
MFS Device Descriptor Format . . . . .	145
Building the Device Characteristics Table . . . . .	146
Using the MFSDCT Utility (DFSUTB00) . . . . .	146
Creating MSC Descriptors . . . . .	148
MSC Descriptor Format . . . . .	148
Using Exit Routines. . . . .	148
Starting ETO . . . . .	149
Logging onto ETO Terminals . . . . .	150
Limiting Dynamic Logon To Specific Terminal Types . . . . .	150
Creating and Reusing LTERM Control Blocks . . . . .	150
Using Default CINIT or BIND User Data Formats . . . . .	150
Signing On and Queue LTERM Allocation . . . . .	151
Providing Signon Data. . . . .	152
Providing Signon Data for ISC, SLU-P, Finance, and Output-Only Devices . . . . .	152
Signing On Multiple Times . . . . .	152
Receiving the Signon Required Message—DFS3649A . . . . .	153
Receiving the Session Status Message—DFS3650 . . . . .	154
ETO Terminal-LTERM Relationship . . . . .	154
How IMS Determines Which Queues to Allocate . . . . .	155
Setting Special Processing Modes . . . . .	155
Using ETO with Printers . . . . .	156
Direct Printing. . . . .	156
Associated Printing . . . . .	156
How Associated Printing Works . . . . .	156
Identifying Printer Node Names . . . . .	157
Coding the Signon Exit Routine for Associated Printing. . . . .	157
Defining Your Printers . . . . .	157
Single-User Structure . . . . .	158
Multiple-User Structures . . . . .	158
Sharing Printers Using ETO. . . . .	158
Using Operator Commands . . . . .	158
Using the /OPNDST Command . . . . .	159
Using the /ASSIGN Command. . . . .	159
Using System Definition Parameters . . . . .	159
Using the DLQT Parameter To Set DEADQ Status Time . . . . .	159
Using Autosignoff (ASOT) . . . . .	160
Using Autologoff (ALOT) . . . . .	161
Using the Autosignoff and Autologoff Timer . . . . .	163
Using Autologon . . . . .	163
Assigning Output. . . . .	164
Asynchronous Output . . . . .	164
Asynchronous Output to a Valid Destination. . . . .	164
Asynchronous Output to an Invalid Destination. . . . .	164
Delivering Output Messages to Non-Originating Terminals . . . . .	165
Inadvertent Output Data Streams. . . . .	165
Signing Off . . . . .	165
Logging Off. . . . .	166
Improving Performance by Deleting ETO Control Blocks . . . . .	166
Using the IDC0 Trace Facility . . . . .	166
ETO and LU 6.1 (ISC) Terminals . . . . .	167
ETO and STSN Terminals . . . . .	167
SNA STSN Terminal Considerations. . . . .	167
ETO and 3600/Finance and SLU P . . . . .	168



/SIGN Support for ETO STSN Devices: ISC, Finance, and SLU P . . . .	168
ETO with Conversation Mode and Response Mode . . . . .	169
Conversation and ETO . . . . .	169
Response Mode . . . . .	170



---

## Chapter 6. Overview of the Extended Terminal Option

This chapter introduces the IMS Extended Terminal Option (ETO). ETO allows you to add VTAM terminals and users to your IMS system without predefining them during system definition. ETO is part of the IMS Transaction Manager (TM), and provides additional features for users, such as output security, automatic logoff, and automatic signoff.

This chapter provides system programmers with the conceptual information that is required to implement and administer ETO. Read the information in this chapter if you are unfamiliar with ETO. Then, proceed to “Chapter 7. Administering the Extended Terminal Option” on page 123 to understand the tasks associated with administering ETO in your environment.

### **In this Chapter:**

- “Benefits of Using ETO”
- “ETO Terminology” on page 116
- “ETO Concepts” on page 119

---

## Benefits of Using ETO

ETO adds essentially two major enhancements to the Transaction Manager environment. With ETO:

- Users can obtain IMS sessions with VTAM terminals that have not been defined to IMS during system definition.
- Output messages that are destined for particular users are secure, and they reach only those users.

In addition, by installing ETO, you can achieve each of the following:

- Improved system availability by reducing scheduled down time associated with adding or deleting VTAM terminals.
- Faster system availability to users, because they can establish an IMS session from any VTAM terminal in the network.
- Improved IMS security by relating output messages to users, rather than to terminals.
- Reduced number of macros required to define the terminal network. This reduces system definition time and storage requirements.
- Reduced checkpoint and restart time. For ETO terminals and user structures, resources are not allocated until they are actually required; similarly, when they are no longer required, they are deleted.
- Reduced number of skilled system programmer resources that are required for maintaining static terminal definitions.

### **Related Reading:**

For more information on user structures, see “User Structures” on page 116.

For more information on static terminals, see “Terminals” on page 116.

# ETO Terminology

The following terms have ETO-specific meanings that are important for understanding and administering ETO:

- Terminals
- Dynamic users
- Structures
- Descriptors

## Terminals

### Definitions:

- A *terminal* is a physical VTAM logical unit (LU) that establishes a session with IMS. A physical terminal is represented using a control block.
- When terminals are not built by ETO but are defined at system definition, they are called *static terminals*. A static terminal can be a VTAM node or a BTAM line.
- When messages are sent to a static terminal they are queued to a logical terminal (LTERM) message queue, where they await retrieval by the recipient.
- When a terminal is not defined at system definition and ETO builds a terminal, that terminal is called a *dynamic terminal*, or an *ETO terminal*. For dynamic terminals, the logical terminal (LTERM) is known as a *dynamic user message queue*, LTERM associates the messages with the user, rather than with the physical terminal. Associating messages with the users provides more security for these users, because they can access their messages only when they sign on using their unique user ID. In addition, all users in the network can access their messages from any physical terminal, instead of being restricted to using a particular physical terminal.

## Dynamic Users

**Definition:** An ETO *dynamic user* is a user who signs on to a dynamic terminal and who has a unique identification (user ID) that IMS uses for delivering messages. The user is usually associated with a person but can also be associated with another entity, such as a printer.

## Structures

**Definition:** A *structure* is an IMS control block that represents terminals and users in ETO. The two types of structures are: terminal structures and user structures.

### **Terminal Structures**

**Definition:** A *terminal structure* is a control block that represents a specific terminal that is known to IMS. A terminal structure is created when the individual terminal logs on to IMS. It is deleted when the terminal logs off with no remaining associated activity (such as status that must be retained for the next connection to IMS).

### **User Structures**

**Definition:** A *user structure* is a set of control blocks, including a user block and one or more LTERM blocks. The message queues are associated with the dynamic user, as opposed to the physical terminal, and they are queued to the user ID.

The dynamic user structure connects to the physical terminal only when the user signs on. This provides a secure environment, because different users accessing the same terminal cannot receive each other's messages.

IMS creates a user structure when either of the following events take place:

- A dynamic user signs on to IMS.
- Output messages that are destined for a dynamic user are sent to the user, but the user has not signed on to IMS.

Usually, a user structure represents a person who uses IMS. The user structure name is usually the same as the user ID. A user structure can also represent a logical destination, such as a printer. In this case, the user structure name can be the same as or different from the LTERM name that your installation uses in its application programs and its exit routines. For example, you can assign the same name to a user structure for a printer that you assign to its LTERM destination node name. However, output is then queued according to the terminal, and not to the user.

Figure 19 and Figure 20 show the differences between static resources and ETO dynamic resources.

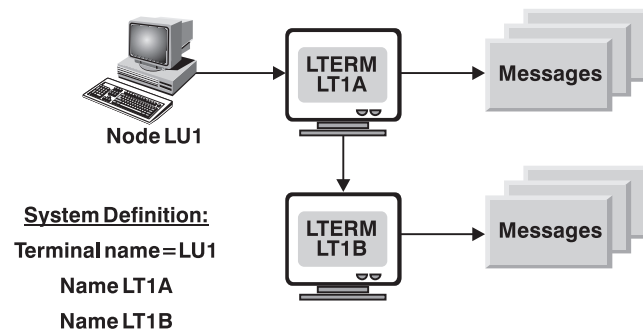


Figure 19. Static Resources

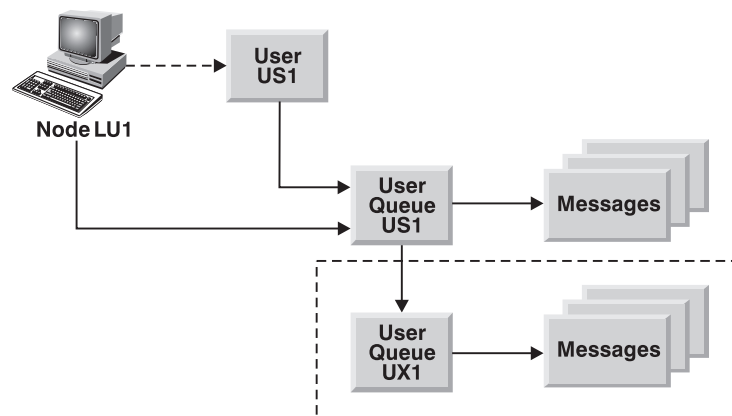


Figure 20. ETO Dynamic Resources

## Descriptors

**Definition:** A *descriptor* provides information to IMS when IMS builds a dynamic resource for a logon or a signon.

The four types of ETO descriptors are:

- Logon descriptors
- User descriptors
- MSC descriptors

## ETO Terminology

### MFS device descriptors

IMS stores descriptors in three IMS.PROCLIB members:

- DFSDSCMx** Contains the descriptors that are automatically generated during IMS system definition. The suffix of DFSDSCMx matches the suffix that your installation specifies on the SUFFIX= parameter of the IMSGEN system definition macro.
- DFSDSCTy** Contains customized device descriptors that your installation creates. Descriptors in DFSDSCTy override duplicate descriptors in DFSDSCMx, and the last descriptor that is defined is used.
- DFRSRxx** Specifies the RSR options used by the online active and tracking subsystems. IMS batch jobs do not use the DFRSRxx member.

### Logon Descriptor

**Definition:** A *logon descriptor* is a skeleton that IMS uses to build an ETO dynamic terminal. It provides information regarding a terminal's physical characteristics. IMS uses logon descriptors in conjunction with exit routines to create terminal structures.

The three types of logon descriptors are: generic, group, and specific.

#### **Definitions:**

- A *generic logon descriptor* provides characteristics for all terminals of a particular type. For example, all SCS printers might share a single generic descriptor. Similarly, all 3270 terminals might share a generic descriptor.
- A *group logon descriptor* provides characteristics for a collection of terminals, each of which has compatible hardware characteristics and is defined to IMS in the same manner. The actual characteristics for these terminals are usually identical, but they can differ; IMS uses the group descriptor to derive their characteristics.

**Example:** You might create separate logon descriptors for different groups of terminals that differ only in the setting for the autologoff (ALOT) time value.

- A *specific logon descriptor* provides characteristics for a single terminal, and these characteristics apply only to that terminal. In this case, the descriptor name matches the name of the terminal that it describes.

**Recommendation:** Although you might need to use specific logon descriptors during the actual migration to ETO, use generic or group logon descriptors after you have migrated to ETO; these kinds of descriptors ease network administration.

### User Descriptor

**Definition:** A *user descriptor* is a skeleton from which a user structure is built. A user descriptor can provide user options and queue names.

### MSC Descriptor

**Definition:** An *MSC descriptor* is used to create a remote LTERM, which is an LTERM that does not exist on the local IMS system. The physical terminal definition (either static or dynamic) for the remote LTERM is in the remote IMS system.

Each MSC descriptor for a remote LTERM is loaded during IMS initialization and tells IMS which MSC link to use for output destined for that remote LTERM.

**Related Reading:** For more information on MSC, see “Chapter 8. Overview of Multiple Systems Coupling” on page 173.

## MFS Device Descriptor

**Definition:** *MFS device descriptors* allow you to add new device characteristics for MFS formatting without requiring an IMS system definition. The MFSDCT utility (DFSUTB00) uses MFS device descriptors to update default formats in the MFS library.

IMS also uses MFS device descriptors to update the MFS device characteristics table. IMS loads this table only during initialization; therefore, updates are not effective until the next IMS initialization.

## ETO Concepts

This section describes important ETO concepts. If you are not familiar with ETO, be sure to read this section before proceeding to “Chapter 7. Administering the Extended Terminal Option” on page 123.

### When Structures Are Created and Deleted

Structures are created when you log on, when you sign on, and when output is queued to your LTERM. In all cases, IMS searches for an existing structure (terminal or user) before creating a new one.

IMS creates and deletes structures in the following sequence:<sup>9</sup>

1. When you establish a session between IMS and an undefined terminal, IMS selects a logon descriptor.
2. Using the information in the logon descriptor, the customization defaults, and VTAM information, IMS builds a VTAM terminal control block that describes the new terminal.
3. When you sign on, if a user structure does not exist, IMS builds one, using information from a user descriptor that it selects, and then connects this user structure to the terminal structure.
4. IMS deletes structures when they are no longer needed to maintain sessions. User structures are typically deleted when you sign off, if no special status needs to be maintained and if no messages remain queued. IMS deletes terminal structures when no terminal status exists (such as trace mode), no user is signed on, and the terminal is idle.

**Exceptions:** The following terminal structures and user structures are not deleted:

- SLU P and Finance terminal and user structures are only deleted during IMS coldstart.
- ISC terminal and user structures are only deleted following a cold session termination.

### Descriptors and Exit Routines

The main purpose of ETO is to dynamically define terminals to IMS. Using descriptors and exit routines, you can assign characteristics to these dynamic terminals and assign user structures to be associated with those terminals.

9. This sequence applies only to terminal logon and logoff and to user signon and signoff. When asynchronous output is queued to a user, IMS creates the user structure, as needed.

## ETO Concepts

A descriptor provides the basic information for the dynamic terminal. An exit routine completes or changes this information. Two methods of using descriptors and exit routines are:

- You can use many descriptors and code little or no processing logic in exit routines.
- You can use few descriptors and code exit routines to perform much of the processing.

## How Descriptors Are Created and Used

All descriptors are created during IMS initialization, prior to IMS startup. You must specify that you want ETO support and ensure that the ETO initialization exit routine (DFSINTX0) does not disable ETO support.

During IMS initialization, IMS reads and validates all ETO descriptors. IMS initialization then continues, and the descriptors remain in storage for the duration of IMS execution. Any changes you make to descriptors become effective after the next initialization of IMS.

IMS uses descriptors to create both terminal and user structures. IMS rebuilds structures during an IMS restart, if appropriate. For example, if messages are queued for a structure and IMS goes down, the structures are rebuilt when IMS restarts. IMS rebuilds these structures to be the same as they were before the IMS restart. IMS does not use the descriptors or exit routines to rebuild these structures. Therefore, any changes you make to descriptors are only reflected in new structures that are built after IMS restart, and the changes are not reflected in structures that are rebuilt during IMS restart.

**Example:** USERA signs on using descriptor DESCA which specifies ASOT=20. USERA starts an IMS conversation, and then IMS abnormally terminates. The system programmer changes DESCA to ASOT=10. After the IMS restart, USERB signs on using DESCA. USERA was rebuilt during the IMS restart. USERA still has ASOT=20, and USERB has ASOT=10.

## How Control Blocks Are Created and Deleted

IMS session control blocks are created for an ETO terminal when a VTAM session is established between the terminal and IMS. Similarly, user control blocks are created for the user during signon or when messages are queued to the user.

After a user signs off, IMS frees the storage associated with the user's control block, unless this user:

- Has output messages waiting on the user local message queue.
- Has specified a special status that is maintained across signoff and subsequent signon (such as conversation mode or response mode).
- Is allocated to an STSN terminal (Finance, SLU P, or ISC).

IMS frees session control blocks when the session terminates, unless:

- You have set a status, such as trace.
- The terminal is an STSN terminal (Finance, SLU P, or ISC) undergoing a warm session termination.

**Related Reading:** For information on managing storage requirements, see "Improving Performance by Deleting ETO Control Blocks" on page 166, and *IMS/ESA Administration Guide: System*.



## Summary of ETO Implementation

Figure 21 illustrates the concepts covered in this chapter and shows an overall view of an ETO implementation.

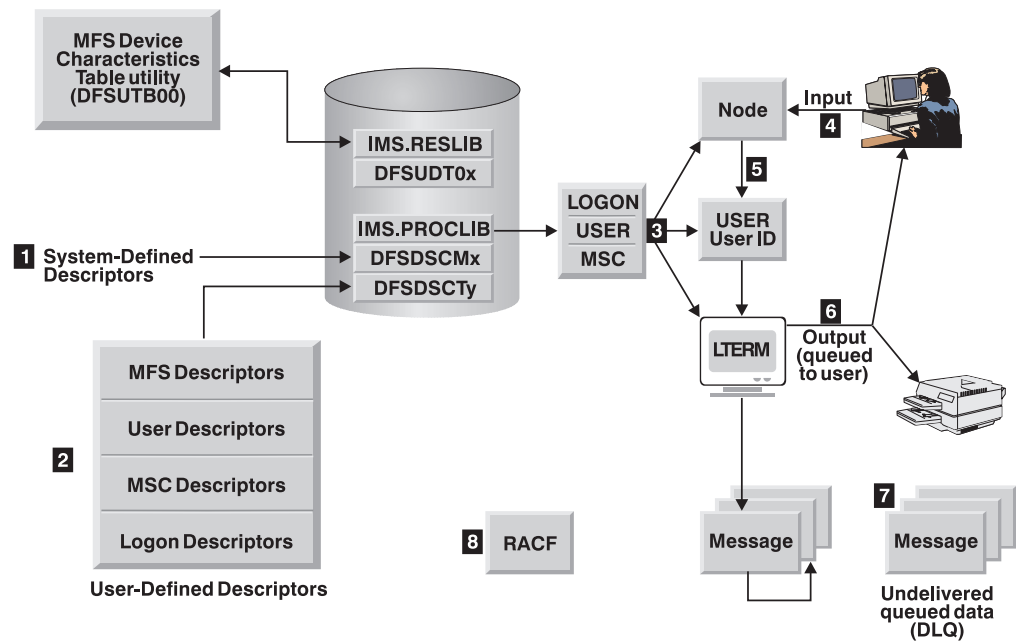


Figure 21. Summary of ETO Implementation

- 1** The system-defined descriptors that are built during system definition are stored in IMS.PROCLIB as member DFSDSCMx.
- 2** Your user-defined descriptors that are written to override the system definition defaults are stored in IMS.PROCLIB as member DFSDSCTy. MFS descriptors that are processed by the MFS Device Characteristics Table utility (DFSUTB00) are stored in the device characteristics table.
- 3** Logon, user, and MSC descriptors are loaded at IMS initialization using the input from IMS.PROCLIB.
- 4** The Logon and Logoff exit routines are called during logon and logoff.
- 5** The Signon and Signoff exit routines are called during signon and signoff.
- 6** Output is delivered to the destination specified in the Output Creation exit routine, unless the user is created during signon.
- 7** If IMS is unable to determine where output should be delivered, the messages are added to the dead-letter queue. Messages might not be delivered because:
  - The user name is not a valid user ID.
  - The user signon is rejected.
  - A physical-to-logical relationship does not exist between the device and the LTERM.
- 8** RACF (or an equivalent product) manages security in the ETO environment.

## ETO Concepts

---

## Chapter 7. Administering the Extended Terminal Option

The IMS Extended Terminal Option (ETO) allows you to dynamically add VTAM terminals and users to your IMS system without having to first define them during system definition. With ETO, IMS dynamically accepts user requests to sign on without requiring these users to be defined to IMS at system definition. ETO dynamically creates user structures for a terminal session when:

- The user signs on to IMS.
- Output messages are sent to the user and await retrieval by the user.

Some of the administrative advantages of using ETO include:

- You do not need to code the following macros for the system definition stage-1 input stream:

MSC remote NAME macros

VTAM macros: TYPE, TERMINAL, NAME, VTAMPOOL, SUBPOOL

Removing these macros reduces the complexity of network management.

- You need to perform fewer system definitions.
- You schedule fewer planned outages for new system definitions.

### **In this Chapter:**

- “Planning for ETO” on page 124
- “Planning a High-Security Environment with ETO” on page 134
- “Coding ETO Descriptors” on page 136
- “Using Exit Routines” on page 148
- “Starting ETO” on page 149
- “Logging onto ETO Terminals” on page 150
- “Signing On and Queue LTERM Allocation” on page 151
- “Using ETO with Printers” on page 156
- “Using Operator Commands” on page 158
- “Using System Definition Parameters” on page 159
- “Assigning Output” on page 164
- “Signing Off” on page 165
- “Logging Off” on page 166
- “Improving Performance by Deleting ETO Control Blocks” on page 166
- “Using the IDC0 Trace Facility” on page 166
- “ETO and LU 6.1 (ISC) Terminals” on page 167
- “ETO and STSN Terminals” on page 167
- “ETO with Conversation Mode and Response Mode” on page 169

Using ETO, you can ensure that all terminals and users are able to establish sessions with IMS, even if these terminals and users are not defined to IMS during system definition. You can use execution-time parameters and exit routines to authorize users to access some or all of the functions that ETO provides.

### **Related Reading:**

- For more information on implementing ETO, including an implementation checklist and example, see *IMS/ESA Version 4 Extended Terminal Option (ETO) Installation and Planning Guide*.

## Administering the Extended Terminal Option

- For information on specific device types that IMS supports, see *IMS/ESA Release Planning Guide* .
- For information on using execution parameters, see *IMS/ESA Administration Guide: System* .
- For information on using exit routines, see *IMS/ESA Customization Guide*.

---

## Planning for ETO

This section provides planning information to help you migrate your current static-terminal environment to an ETO environment.

Although you can continue to define VTAM terminals and LTERMs to IMS during system definition, if you do so:

- You cannot take advantage of the ETO features that exist for those terminals.
- You must fully define the terminal. You must supply all TERMINAL macros, NAME macros, and parameters.

ETO terminals must be VTAM terminals.

**Restrictions:** The following VTAM terminals cannot be ETO terminals:

- IMS master terminal (MTO)
- IMS secondary master terminal
- MSC physical and logical links
- ISC sessions that are used by XRF for surveillance
- LU 6.2 terminals (dynamically created and managed by APPC/IMS)

## Identifying Your Requirements

ETO is considered fully implemented when no static VTAM terminals exist in the system and when the majority of terminals and users are defined using the default logon descriptor and default user descriptor. However, because installations vary in application program dependencies, the cost of fully implementing ETO also varies.

Your installation should determine the extent to which full ETO support is required, based on the following requirements:

### **Full user-message security**

Full implementation of ETO is required for full user-message security. In this environment, no node-name user descriptors exist. Any requirements for user structures that the default user structure does not provide must be defined by user descriptors or by the Signon exit routine (DFSSGNX0).

### **Dynamic terminal support only**

Only partial implementation is required for dynamic terminal support. You can move network definition statements from the system definition to ETO descriptor PROCLIB members. Benefits of this implementation include:

- Fewer system generations are required in order to maintain network definitions, because you can change descriptors between IMS warm starts.
- Shorter run times are required for system generation, because you do not need to define VTAM terminal networks.

- Improved performance exists for IMS checkpoint and restart, because dynamic terminals and user resources are allocated only when they are used.

With partial implementations, however, you do not achieve improved user-message security, because each LTERM has a fixed relationship with a physical terminal.

## Restrictions on ETO

Before implementing ETO, be aware of the following restrictions:

- ETO dynamic terminals do not support the Security Maintenance utility (SMU).
- Dynamic terminals are not supported for terminal-related MSDBs or for non-terminal-related MSDBs that have LTERM keys.
- Application programs that use specific LTERM names sometimes require particular ETO customization.

**Related Reading:** For more information on how to customize ETO for application programs that have dependencies on LTERM names contained in the I/O PCBs, see “Using DFSUSER User Descriptors” on page 144.

## Defining Physical Terminals

When implementing ETO, ensure you achieve your desired VTAM terminal network by taking each of the following actions:

- Assess how often IMS application programs depend on specific terminal characteristics.
- Check the accuracy of each VTAM terminal definition. For each dynamic terminal, ETO builds a terminal structure that relies on the VTAM definition for the characteristics (such as the LU type, screen size, and model) for that terminal. Terminal characteristics that are specified in your IMS system definition might differ from (and override) those in the VTAM definitions. If these terminal characteristics in the IMS system definition are compatible with those of the actual terminal, the discrepancy is not apparent.<sup>10</sup>
- Either provide specific node-name logon descriptors or use the Logon exit routine (DFSLGNX0) for terminals that are not adequately defined using the default logon descriptor.
- Code one of the following two exit routines to determine the device type:
  - The Logon exit routine (DFSLGNX0) can determine the device type by examining the default logon descriptor.
  - The Signon exit routine (DFSSGNX0) can determine the device type later in the process by examining the terminal control blocks.

When receiving a request to establish a terminal session, IMS relies on the following information from VTAM session parameters:

- **UNITYPE**

The unit type of the node that is attempting to log on. IMS determines the UNITYPE by using the following fields:

---

10. If the actual terminal characteristics do not match those in the IMS definition or the VTAM definition, it is possible that the terminal can function with IMS.

## Planning for ETO

- The LUTYPE field of the CINIT<sup>11</sup> request provides part of the UNITYPE information. The LUTYPE value is usually found in the first byte of the PSERVIC operand of the MODEENT macro, which is used to generate the VTAM mode table entry that is used for a logon. Figure 22 shows the fields of the PSERVIC operand in the VTAM MODEENT macro.

Table 5 shows the mapping for the LUTYPE value and the IMS UNITYPE.

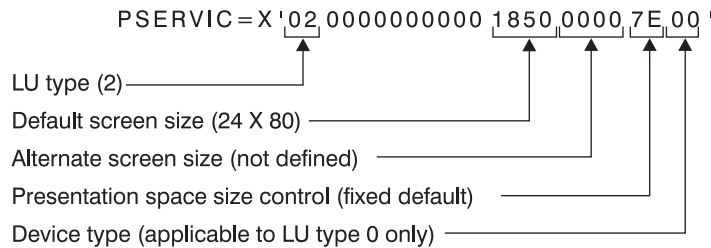


Figure 22. VTAM MODEENT Macro PSERVIC Operand Fields that IMS Uses

Table 5. Mapping for VTAM LUTYPE Value to IMS UNITYPE

VTAM LUTYPE	IMS UNITYPE
LUTYPE X'06'	LUTYPE6
LUTYPE X'04'	SLUTYPE4
LUTYPE X'02'	SLUTYPE2
LUTYPE X'01'	SLUTYPE1 (default) or NTO
LUTYPE X'00'	3270, SLUTYPEP, or 3600/Finance

- If the LUTYPE field is X'00' (indicating that the terminal is 3270 non-SNA, Finance, or SLU P), IMS must check the transmission services profile specification in the TS field of the CINIT request. The TS value is usually found in the TSPROF operand of the MODEENT macro that is used to generate the VTAM mode table. This LUTYPE value must match the value in the logon descriptor that IMS selects for a logon.

Table 6 shows the mapping for the TSPROF specification and the IMS UNITYPE.

Table 6. Mapping for TSPROF Specification to IMS UNITYPE

TSPROF Specification	IMS UNITYPE
X'02' or X'03'	3270
X'04'	SLUTYPEP (default) or 3600/Finance

- **Input RU size**

The input RU (request unit) size in the BIND must be less than or equal to the RECANY buffer size for the IMS system (also required for static terminals).

- **Output RU size**

The output RU size in the BIND must be greater than or equal to the OUTBUF size that IMS determines for the terminal from the selected logon descriptor. This parameter is also required for static terminals.

- **Screen size and model number**

For non-SNA 3270 and SLUTYPE2 devices, IMS retrieves both screen size (row and column) and model number from the BIND:

11. CINIT is a network services request sent from a system services control point (SSCP) to a logical unit (LU), asking that LU to establish a session with another LU and to act as the primary end of the session.

- For static terminals, the screen size is the value that was specified in the system definition.
- For dynamic terminals, IMS determines the screen size from the VTAM definition or from the Logon exit routine (DFSLGNX0).

**Recommendation:** Until the ETO feature was available, IMS ignored the screen size and model number values in the BIND, because the IMS system definition held this information. Therefore, check to ensure this definition is accurate.

If you determine that a VTAM definition is inaccurate, you can use the Logon exit routine (DFSLGNX0) to override the VTAM-provided screen size and model number. For example, use a terminal naming convention or MODETAB definition convention. The Logon exit routine can also assign USER=NODE as a name, when appropriate.

IMS uses the values in the PSERVIC operand of the MODEENT macro, which is used to generate the VTAM mode table entry that is used for a logon. MFS formats must be available for all screen sizes that IMS dynamically builds.

**Restriction:** IMS does not use the 3270 Read Partition Query (RPQ) command to determine the screen size from the device controller.

### **Related Reading:**

- For more information on the PSERVIC operand values for screen size and model numbers, see “Identifying VTAM Device Types, Screen Sizes, and Models” on page 128.
- For more information on the Logon exit routine (DFSLGNX0), see *IMS/ESA Customization Guide* .

## **Planning for Both Static and Dynamic Terminals**

Static and dynamic terminals can coexist in the same IMS system. However, if users move between static and dynamic terminals, plan for the following situations:

- IMS maintains separate queues for static and dynamic terminals. A static terminal has one or more LTERMs associated with it, as controlled by the IMS system definition (or the /ASSIGN command to move an LTERM to a different terminal). Some users become accustomed to having their output queue follow them from ETO terminal to ETO terminal. Static terminals do not provide this feature. Users need to be able to differentiate between static and dynamic terminals, or confusion can result.
- Given one static terminal and one dynamic terminal, separate IMS conversations can exist at the same time. The dynamic terminal conversation belongs with the user structure and follows the user from terminal to terminal.<sup>12</sup> The static conversation belongs to the static terminal and can only be released to a static terminal. This situation is user friendly and predictable only when the user is certain of the terminal type (static or dynamic).

Normally, the terminal operator is able to determine whether a terminal is static or dynamic by checking the security information that is provided at the end of the DFS3650 (SESSION STATUS) message:

### **OUTPUT SECURITY AVAILABLE**

Indicates that the terminal is dynamic, and output is associated with the signon ID.

---

12. This assumes that the Signon exit routine (DFSSGNX0) sets the same user name each time, as is usually the case.

## Planning for ETO

### NO OUTPUT SECURITY AVAILABLE

Indicates that the terminal is either statically defined or that ETO created it by using one of two methods:

- Using a node user descriptor
- Using the Signon exit routine to assign the node name to the user structure

In either case, the output is associated with the terminal, rather than with the user.

**Exception:** Message DFS3650 might be suppressed if you use the NOTERM option.

**Recommendation:** To ease migration and limit possible confusion, convert to dynamic ETO terminals by using logical groupings within your organization, such as departments or floors.

### Defining Terminals for Growth

When designing your ETO implementation, be sure to plan for growth in your network.

#### Recommendations:

- To increase future growth potential and system availability, minimize the use of descriptors. The Logon, Signon, and Output Creation exit routines should provide enough customization, thereby eliminating the need for unique descriptors beyond those that are required for specific terminal types.
- To ensure user data is correctly handled, design exit routines carefully. In particular, carefully plan for user data that is specified during the logon process. Exit routines should work correctly, regardless of whether user data is specified. Some terminal types, such as Finance and SLU P terminals, require user data that specifies signon information. If this data is missing, the equivalent information must be provided in the Logon exit routine (DFSLGNX0).

**Related Reading:** For more information on the Logon exit routine (DFSLGNX0), Signon exit routine (DFSSGNX0), and Output Creation exit routine (DFSINSX0), see *IMS/ESA Customization Guide*.

For more information on migrating static VTAM terminals to ETO terminals, see "Identifying VTAM Device Types, Screen Sizes, and Models".

## Identifying VTAM Device Types, Screen Sizes, and Models

VTAM logon CINIT user data provides IMS with information to build session control blocks. This information must be accurate to ensure that the correct terminal-related control blocks is built from information in the logon descriptors. Carefully define your VTAM PSERVIC parameters to ensure that IMS selects the appropriate logon descriptors and establishes screen sizes using specific terminal characteristics.

The BIND is rejected if the input and output RU sizes in the BIND are incompatible with the IMS RECAN and descriptor OUTBUF sizes.

### Defining Device Types

IMS dynamic terminal control blocks are built from a combination of the following:

- The logon descriptor



- Information that VTAM passes to IMS during logon
- The MFS device characteristics table

If the definitions are coded incorrectly, IMS chooses the wrong MFS format and device characteristics, possibly causing screen format errors.

VTAM passes the physical terminal characteristics to IMS. The following fields have information that IMS uses to determine device types:

- LUTYPE field from the VTAM PSERVIC parameter of the VTAM MODEENT macro
- TS profile from the TSPROF parameter of the VTAM MODEENT macro

**Non-SNA 3270 Printers and Displays:**

The default descriptor names for non-SNA 3270 printers and displays are DFS327P and DFS3270.

Rules required for defining non-SNA 3270 devices are:

- The TSPROF parameter of the VTAM MODEENT macro must be 2 or 3.
- The PSERVIC parameter of the VTAM MODEENT macro must specify LU Type=0.
- Byte 12 in the VTAM PSERVIC parameter must be modified so that it distinguishes a printer from a video.

```
bits 0-1 00 unspecified device type
          01 printer device
          10 display device
          11 display/printer device (3275)
bits 2-7 reserved
```

If you specify nothing (B'00'), the default is B'10', which is a display device. <sup>13</sup>

**Related Reading:** For more information on defining non-SNA 3270 devices, see *ACF/VTAM System Programmer's Guide*.

**LU Type-2 Devices:**

The following is a list of rules required for defining SLU-2 devices:

- The TSPROF parameter of the VTAM MODEENT macro must be 3.
- The PSERVIC parameter of the VTAM MODEENT macro must specify LU Type=2.

**Related Reading:** For more information on defining LU type-2 devices, see *ACF/VTAM System Programmer's Guide*.

**3275 Devices:**

VTAM definitions alone cannot identify a 3275 device as it is logging on. Additional information (UNIT=3275) must be specified in a logon descriptor. Define this descriptor in the CINIT user data or in the Logon exit routine. The descriptor itself must be identified as a 3275 device type.

Static 3275 devices are defined in the IMS system definition as follows:

---

13. B'00' means bits 0 and 1 equal 00. B'10' means bits 0 and 1 equal 10.

## Planning for ETO

```
TYPE          UNITYPE=3270
TERMINAL UNIT=3275,TYPE=3270-An,SIZE=(24x80),COMPT=PRT1
```

The 3275 has only one buffer. This forces the display and printer components to be the same model. The VTAM definitions for a dynamic 3275 as statically defined above are:

```
PSERVIC=X'000000000000185000007EC0'
```

### ***NTO Devices:***

The terminal must identify itself as an NTO device in one of the following ways:

- LU presentation services profile in the BIND image must specify LU1.
- When the LUTYPE is NTO, IMS uses the data stream compatibility byte to precisely define the specific NTO device type. If the LUTYPE specified is SLUTYPE1, the data stream compatibility byte is ignored. The terminal is assumed to be an actual SLUTYPE1 as defined on the logon descriptor.
- Logon exit routine, if available, must accept the DFSNTO (default) logon descriptor or specify an NTO logon descriptor that is defined by your installation using the node name or LOGOND user data parameter.
- The data stream compatibility byte in the CINIT specifies the device type:

<b>2740</b>	2740-1 NTO device
<b>2741</b>	2741 NTO device
<b>WTTY</b>	TTY NTO device

If none of the above is specified, the device type is LUNS NTO.

### **LU2 and Non-SNA 3270 Screen Size and Model Information**

After the Logon exit routine approves the logon, BIND image data (screen-size) and feature information from the logon descriptor are used to search the MFS device characteristics table for the appropriate MFS device information. If the model is specified in the screen size control byte the MFS device characteristics table is not searched. A DFS3646 error message is issued if no match exists on screen-size and feature. The information from the proper MFS device characteristics table entry is then used for that device. This information in the MFS device characteristics table comes from the IMS system definition or the MFSDCT utility (DFSUTB00).

If the screen-size control byte is X'7F' and both the default and alternate screen-sizes are specified, a search of the MFS device characteristics table using alternate screen size commences. If no match is found, another search begins using the default screen size. If no screen size is found, message DFS3646I is issued to the operator.

The screen size (the product of the lines and columns) must be in the range 80-16384. The lines and columns must each be in the range 1-255.

If the screen size and features of the 3270 device that is logging on maps into two or more MFS device characteristics table entries, the first entry in the table that matches the screen size and features is selected.

If the screen-size control byte (the 11th byte of the PSERVIC on the VTAM MODEENT) is X'00' and both default and alternate screen-sizes are specified, a search of the MFS device characteristics table using the default screen-size occurs. If no match is found, another search begins using the alternate screen-size.

Use the Logon exit routine, DFSLGNX0, to override the screen-size or model for the device during logon.

**LU2 Screen-Size and Model Information:** Screen-size and model information applies to LU type–2 devices. The following is an algorithm IMS uses to determine the model or screen size for ETO terminals:

1. If you want to override the model value, use the Logon exit routine (DFSLGNX0). Valid values are X'01' and X'02', corresponding to model 1 and model 2 in the logon descriptor. Screen-size and model specifications in the VTAM PSERVIC are ignored. Model=X'01' represents a 12x40 screen-size. Model=X'02' represents a 24x80 screen-size. The MFS device characteristics table is not searched for MFS device information. The features are obtained from the logon descriptor.
2. If you want to override the screen-size value, use the Logon exit routine, DFSLGNX0. Be aware that if you override the model, the screen-size override is ignored. The features from the logon descriptor and the screen-size are used to search the MFS device characteristics table for the MFS device information.
3. Screen-size is established based on the model specification in the VTAM PSERVIC. IMS determines the model by checking the screen-size control byte in the VTAM PSERVIC field for an X'01', X'02', or X'03'. The model is established accordingly. Screen-size specifications are ignored when the model is specified. X'01' represents a model 1 with a 12x40 screen-size, and X'02' and X'03' represent a model 2 with a 24x80 screen-size. If the model is specified, the MFS device characteristics table is not searched for MFS device information. The features are obtained from the logon descriptor.
4. Screen-size is established based on the screen-size specifications in the VTAM PSERVIC. A value of X'7E' in the screen-size control byte causes the default screen size in the VTAM PSERVIC to be used. A X'7F' causes the alternate screen-size to be used first to search the MFS device characteristics table. If no match is found, the default screen-size is used to search; the first match is the screen-size. If no match is found, message DFS3646I is sent to the operator.
5. If the screen-size control byte is X'00', the default and alternate screen-size specifications in the VTAM PSERVIC are used to search the MFS device characteristics table for MFS device information. If a match is made on the default size, the default is used. If a match is made on the alternate size, the alternate is used. If no match is made, the logon is rejected.
6. If the screen-size control byte is X'00' and no screen-size is specified, the device is defaulted to a model 2 device, and the screen-size is established (24x80).
7. The screen-size is established in the BIND parameter in the default screen-size field. The erase write (EW) command is always used.

**Non-SNA 3270 Screen-Size and Model Information:** Model information applies to VTAM 3270 Record-Mode devices (non-SNA 3270s). To determine the model or screen-size for ETO terminals IMS uses the following algorithm:

1. If you want to override the model value, use the Logon exit routine (DFSLGNX0). Valid values are X'01' and X'02', corresponding to model 1 and model 2 in the logon descriptor. Screen-size and model specifications in the VTAM PSERVIC are ignored. Model=X'01' represents a 12x40 screen-size. Model=X'02' represents a 24x80 screen-size. If the model is specified, the MFS device characteristics table is not searched for MFS device information. The features are obtained from the logon descriptor. Device type and screen-size are determined from the model value.

## Planning for ETO

2. If you want to override the screen-size value, use the Logon exit routine (DFSLGNX0). Be aware that if you override the model, the screen-size override is ignored. The features from the logon descriptor and the screen-size are used to search the MFS device characteristics table for the MFS device information. Device type and screen-size are implied by the model value.
3. Screen-size is established based on the model specification in the VTAM PSERVIC field. IMS determines the model by checking the screen-size control byte in the VTAM PSERVIC field for X'01', X'02', or X'03'. The model is established accordingly. Screen-size specifications are ignored when the model is specified. X'01' represents a model 1 with a 12x40 screen-size, and X'02' and X'03' represent a model 2 with a 24x80 screen-size. If the model is specified, the MFS device characteristics table is not searched for MFS device information. The features are obtained from the logon descriptor. Device type and screen-size are determined from the model value.
4. Screen-size is established based on the screen-size specifications in the VTAM PSERVIC field. A value of X'7E' in the screen-size control byte causes the default screen size in the VTAM PSERVIC field to be used. A X'7F' causes the alternate screen-size to be used first in searching the MFS device characteristics table. If no match is found, the default screen-size is used to search; the first match is the screen-size. If no match is found, message DFS3646I is sent to the operator. Device type and screen-size are implied by the model value.
5. If the screen-size control byte is X'00', the default and alternate screen-size specifications in the VTAM PSERVIC field are used to search the MFS device characteristics table for MFS device information. If a match is made on the default size, the default is used. If a match is made on the alternate size, the alternate is used. If no match is found, the logon is rejected. Device type and screen-size are implied by the model value.
6. If no model information and screen-size are specified in the VTAM PSERVIC field, IMS uses the CINIT's model byte. Model type is established by the VTAM definition statement FEATUR2. The model information applies only if the screen-size control byte is X'00' and if the screen-size is also X'00'. This applies to printers and displays. X'00' is the default and corresponds to a model 1. X'01' corresponds to a model 2. Device type and screen-size are implied by the model value.
7. The screen-size is used to determine the type of write command used.
  - If the screen-size is equal to 960 or greater than 1920, IMS uses erase write alternate (EWA).
  - If the screen-size is less than or equal to 1920 and not equal to 960, IMS uses erase write (EW).The device type and screen size are determined from the model value.

**Screen Definitions:** You can define the same screen in three different ways:

- To VTAM for ETO terminals
- To IMS system definition for statically defined terminals
- To IMS Message Format Service (MFS) DEV statements for ETO and static terminals

**Examples:**

- Example 1. LU0 (non-SNA 3270 video)

```
PSERVIC=X'00000000000000000000200'
SysDef:  UNITYPE=3270  UNIT=3284/86  MODEL=2
MFS:  TYPE=(3270P,2)
```

Byte 12 defaults to a video X'80'.

- Example 2. Model 2 non-SNA 3270 Printer

```
PSERVIC=X'00000000000000000000240'
SysDef:  UNITYPE=3270  UNIT=3284/86  MODEL=2
MFS:  TYPE=(3270P,2)
```

- Example 3. Non-SNA 3270 Display (model specified)

```
PSERVIC=X'00000000000000000000080'
SysDef:  UNITYPE=3270  MODEL=1
MFS:  TYPE=(3270,1)
```

Model information can come from the FEATUR2 parameter for non-SNA 3270; if this parameter is not specified, this is a model 1 (screen-size 12x40). Assume that FEATUR2=1 (specified or default).

- Example 4. Non-SNA 3270 Display (model specified)

```
PSERVIC=X'00000000000000000000080'
Sysdef:  UNITYPE=3270  MODEL=2
MFS:  TYPE=(3270,2)
```

Model information can come from the FEATUR2 parameter for non-SNA 3270; if this parameter is not specified, this is a model 2 (screen-size 24x80). Assume that FEATUR2=2.

- Example 5. Non-SNA 3270 Display (screen-size specified)

```
PSERVIC=X'000000000000185000007E80'
SysDef:  UNITYPE=3270  TYPE=3270-A2,SIZE=(24,80)
```

For a SLU-2 device, UNITYPE=SLUTYPE2 would be specified (also note the change to the PSERVIC field).

```
MFS:  TYPE=3270-A2, where A2=24x80
```

The screen-size comes from the default screen-size field (24x80). (For a SLU-2 device, the first byte of the PSERVIC field would be X'02'. The last byte would be X'00').

- Example 6. Non-SNA 3270 Display (screen-size specified)

```
PSERVIC=X'00000000000020500000080'
SysDef:  UNITYPE=3270  TYPE=3270-A3,SIZE=(32,80)
MFS:  TYPE=3270-A3, where A3=32x80
```

The screen-size comes from the default screen-size field (32x80). (For a SLU-2 device, the first byte of the PSERVIC field would be X'02'. The last byte would be X'00').

- Example 7. Non-SNA 3270 Display (model specified)



You can continue to use the SMU to define signon and other SMU security options for static terminals and for user structures that are defined during system definition.

**Recommendation:** To simplify your security profile, use only one security solution, such as RACF or an equivalent security product.

## Planning for MFS

After you implement ETO dynamic terminal support, the number and diversity of terminal types is likely to increase. When a terminal dynamically establishes an IMS session, MFS formats might not be available for the device type that is requesting the session. You can solve this problem by:

- Restricting device types to only those that are used in the MFS definitions for your application programs.

**Recommendation:** Use the Logon exit routine (DFSLGNX0) to select the desired logon descriptor.

- Extending the MFS device output formats to include the new terminal types that connect to the IMS system.

**Recommendations:**

- Use the MFS Generation utility (MFSGU) facilities, such as the STACK statement, to make creating these additional formats easier.
- Use the MFSDCT utility (DFSUTB00) to avoid needing to perform an IMS generation.

Often, multiple device types have similar or identical capabilities, so the distinction of device type might not be important to your environment. Although IMS application programs are usually not sensitive to device type, such a dependency can exist. For example, MFS can create different input messages from the same input data stream, due to differences in the format specification for device input.

## Planning User IDs

When planning user IDs for use with ETO:

- Ensure that dynamic user structures have unique names in IMS. Support for unique user IDs depends on whether user-based security is important. For user-based security, user IDs must be unique to the user. Otherwise, user IDs can be the same as the terminal LU name.
- Code the Signon exit routine (DFSSGNX0) to provide user ID suffixing, if users need to sign on to more than one terminal with the same user ID at the same time. Review and select the SGN and RCF EXEC keyword parameters.
- Analyze how application programs use the user field in the I/O PCB.

## Planning User Queue Names

When planning user queue names:

- Ensure that dynamic user queue names are unique in the IMS system. Support for unique queue names depends on whether user-based security is important. For user-based security, user queue names must be unique to the user. Otherwise, user queue names can be the same as the terminal LU name.
- Specify user descriptors or provide logic in the Signon exit routine (DFSSGNX0) if the user queue name cannot be the same as the user name.
- Analyze how application programs use the LTERM field in the I/O PCB.

## Planning for ETO

### Planning Operations

When planning for operations:

- Update your MTO procedures to reflect the concept of dynamic resources. The MTO needs to become familiar with the command input fields and response formats.
- Update the help-desk procedures to reflect the IMS terminal and user resource structures, as well as the commands that are required to diagnose user's problems.
- Review and update automated-operator procedures, if necessary.
- Develop procedures for handling the dead-letter queue.
- Identify requirements for autosignoff and autologoff.
- Develop standards for the conditions under which system-wide values should be used and when they should be overridden.

### Planning for MSC Support with ETO

Although MSC physical and logical links are predefined during system definition, you can dynamically create MSC remote LTERMs during IMS initialization. You can identify a remote MSC NAME to IMS using an MSC descriptor. The descriptor relates each remote resource to the link path name of a generated MSNAME macro.

**Recommendation:** Define remote LTERMs with MSC descriptors to maintain consistency between remote and local systems.

If you choose not to use MSC descriptors in order, the MSC Verification utility, DFSUMSV0, recognizes the remote LTERMs but not the corresponding local LTERMs in the target system. In this case, IMS issues warning message DFS2331W.

IMS processes MSC descriptors that are associated with MSC links defined within the IMS system. IMS ignores other MSC descriptors. You can maintain a single network definition for remote LTERMs in IMS.PROCLIB for multiple interconnected IMS systems that use MSC. In this case, each logical link path name must be unique throughout the entire network.

---

## Coding ETO Descriptors

IMS uses ETO descriptors to dynamically build terminal structures and user structures. The four types of ETO descriptors are:

- Logon
- User
- MFS device
- MSC

The basic format for all ETO descriptors is:

Column	Description
1	One of the following descriptor types:
L	Logon descriptor
U	User descriptor



<b>D</b>	MFS device descriptor
<b>M</b>	MSC descriptor
<b>*</b>	Comment line (ignored by IMS)
<b>3-10</b>	Name of the descriptor, using the following conventions: <ul style="list-style-type: none"> <li>• Must be one to eight alphanumeric characters.</li> <li>• For logon and user descriptors, characters are limited to A-Z, #, \$, and @.</li> <li>• For MSC descriptors, use the link name.</li> </ul>
<b>12-72</b>	One or more keywords and their parameters, separated by blanks. Use commas to separate multiple parameters for a single keyword.
<b>73-80</b>	Optional sequence numbers (ignored by IMS).

**Related Reading:** For more information on defining ETO descriptors and ETO descriptor formats, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

## Creating Descriptors Using the System Definition Process

To ease migration, you can create a starter set of ETO descriptors (except MFS<sup>14</sup> descriptors) using the ETOFEAT keyword on the IMSCTRL macro at system definition. You can add to or modify the starter set by creating an additional IMS.PROCLIB member.

**Restriction:** An LGEN system generation does not create ETO descriptors.

**Related Reading:** For more information on using the ETOFEAT keyword and creating the IMS.PROCLIB member with the ETO descriptors, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Creating descriptors during the system definition process saves you time and ensures that the descriptors are correct. IMS generates an ETO descriptor report, which provides information on the relationship between ETO descriptors and the IMS system definition resources that they represent.

## Storing Descriptors

Descriptors that are created during system definition are stored in the IMS.PROCLIB member, DFSDSCMx. Subsequent system definitions of the same stage-1 input deck overwrite the DFSDSCMx member.

**Recommendation:** To avoid losing descriptors when member DFSDSCMx is replaced, store descriptors that you create by using TSO or MVS utilities in IMS.PROCLIB member, DFSDSCTy. If you need to update descriptors that are created at system definition in DFSDSCMx, use TSO or an MVS utility (such as IEBUPDTE) to make the updates.

---

<sup>14</sup> The system definition creates a device characteristics table based on the stage-1 input that can be used as one of the inputs to the MFSDCT utility (DFSUTB00). The device characteristics table contains what the system-defined device characteristics table contains, plus any additional entries from device descriptors.

## Coding ETO Descriptors

### Creating Logon Descriptors

Logon descriptors provide IMS with information about the physical characteristics of the terminals that establish logon sessions. These characteristics must be consistent with the VTAM logon BIND characteristics.

This section describes how to create and use logon descriptors.

#### Logon Descriptor Format

The format for a logon descriptor is:

```
L descname KEYWORD=parm1 KEYWORD=parm2...KEYWORD=parmN
```

**L** Indicates that the descriptor type is 1ogon.

##### **descname**

The name of the descriptor, which can be an IMS default descriptor name, the field name of the TERMINAL macro, or a unique name you create.

Default names include: DFS3270, DFS327P, DFSSFIN, DFSSLU1, DFSSLU2, DFSSLU4, DFSSLUP, DFSLU61, DFSNTO.

##### **KEYWORD=**

One of the following keywords: <sup>15</sup>

ALOT=  
ASOT=  
BACKUP=  
COMPT1=  
COMPT2=  
COMPT3=  
COMPT4=  
EDIT=  
FEAT=  
MODELTLB=  
OPTIONS=  
OUTBUF=  
PTRSIZE=  
SEGSIZE=  
UNIT= <sup>16</sup>  
UNITYPE=

**parm** A parameter value for the keyword.

**Related Reading:** For more information on valid parameter values, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

**Example:** An example of a logon descriptor is:

```
Column      Column  
1           12  
L MKT01LU2 UNITYPE=SLUTYPE2 ALOT=30
```

---

15. Because you don't need to define the maximum number of ETO-ISC sessions, the SESSION= keyword is not supported on the ETO descriptor. You can continue to add sessions while storage and processor capacity exist.

16. Non-SNA 3270 terminals only.

You can use multiple records to create a single descriptor name (for example, if the number of parameters causes the descriptor to exceed one 80-byte record).

**Example:** The following is an example of a logon descriptor with multiple records:

```
L MKT01LU2 UNITYPE=SLUTYPE2  
L MKT01LU2 ALOT=30
```

### Creating Logon Descriptors at System Definition

When you specify IMS system definition options to create ETO logon descriptors, IMS dynamically creates a logon descriptor for every unique VTAM TERMINAL macro <sup>17</sup> that you have specified. IMS system definition can produce up to 37 common logon descriptors for each device type. The descriptor that defines the largest number of terminals of that type becomes the default logon descriptor. This logon descriptor assumes the IMS-defined default name for that type. The other terminals of that type create their own unique logon descriptor by using a suffix on the default name.

For terminal definitions that don't match one of the 37 common descriptors, IMS creates an individual logon descriptor. These descriptors are generated as comments (with an asterisk in column 1). To choose a descriptor that you need, remove the asterisk.

**Example:** \*L3270A

The naming convention for the 37 common logon descriptors that are created during the system definition process is:

- The last character of the name must be unique.
- Blank for the most common, then 0-9 and A-Z.

### Restrictions:

- During system definition, IMS does not create ETO logon descriptors for the primary or secondary master terminal, or for LU 6.1 terminals that are defined as XRF ISC links.
- During large system generation (LGEN), IMS does not generate logon descriptors.
- The following keywords are not supported on logon descriptors: PU, SIZE, MODEL, TYPE, MSGDEL.

### Criteria for Selecting Logon Descriptors

The logon descriptor contains data that is related to the node and the VTAM CINIT. This data allows IMS to create a control block structure that supports a session.

IMS uses the following criteria (in sequence) to select a logon descriptor:

1. IMS uses existing control blocks for terminals. IMS does not look for a descriptor if it finds existing control blocks.
2. IMS determines whether the Logon exit routine is used to define logon descriptor names. The exit routine extracts the name from the VTAM CINIT <sup>18</sup> user data, or another appropriate algorithm. IMS rejects invalid logon descriptor names.

---

17. For each TERMINAL macro definition, a node user descriptor is also created for use as a migration step to ETO.

18. The LUTYPE and TS= (transmission service level) fields in the VTAM CINIT data must agree with the selected descriptor; otherwise, the logon is rejected.

## Coding ETO Descriptors

- In response to a request to establish a session (logon), IMS examines the LOGOND parameter.<sup>19</sup> The LOGOND parameter can indicate the logon descriptor for IMS to use.

**Restriction:** LOGOND is not valid for ISC parallel sessions.

- If IMS does not find a logon descriptor name, it looks for a logon descriptor with the same name as the VTAM CINIT LUNAME.

**Recommendation:** Use the node name on the logon descriptor if you expect any of the following situations:

- You don't expect to add more of the same terminal type.
- You don't have many of the terminal type.
- You want to simplify the logic in the Logon exit routine.
- For ISC, you want to specify parameters (such as OUTBUF=) other than the default ISC logon descriptor, and you have not coded an exit routine.

If these criteria do not yield a valid logon descriptor name, IMS selects a descriptor by using the default criteria.

### Criteria for Selecting a Default Logon Descriptor

IMS provides a default logon descriptor for each VTAM terminal type. The logon descriptor name is based on the LU type and TS profile. If necessary, you can add logon descriptors. If you do not provide a logon descriptor, and IMS does not locate one in IMS.PROCLIB or in the Logon exit routine (DFSLGNX0), IMS uses the algorithm shown in Table 7 to assign a default logon descriptor name. The VTAM LUTYPE, IMS UNITYPE, and IMS default logon descriptor names are shown for each terminal type.

Table 7. Mapping for VTAM LUTYPE, IMS UNITYPE, and Default Logon Descriptor Names

VTAM LUTYPE	IMS UNITYPE	Default Descriptor Name
X'06'	LUTYPE6	DFSLU61
X'04'	SLUTYPE4	DFSSLU4
X'02'	SLUTYPE2	DFSSLU2
X'01'	SLUTYPE1 (Default) <sup>1</sup> NTO	DFSSLU1 DFSNT0
X'00' (TS = X'04')	SLUTYPEP (Default) <sup>2</sup> Finance 3601	DFSSLUP DFSFIN DFSFIN
X'00' (TS = X'02' or X'03')	3270 3270, UNIT=3284	DFS3270 DFS327P

<sup>19</sup> LOGOND is a parameter in the VTAM CINIT user data. LOGOND is also a keyword on the IMS /OPNDST command.

Table 7. Mapping for VTAM LUTYPE, IMS UNITYPE, and Default Logon Descriptor Names (continued)

VTAM LUTYPE	IMS UNITYPE	Default Descriptor Name
-------------	-------------	-------------------------

**Notes:**

1. IMS does not distinguish between SLUTYPE1 and NTO terminals; SLUTYPE1 is the default. To support both SLUTYPE1 and NTO terminals, do the following:
  - Override the DFSSLU1 default logon descriptor by using the Logon exit routine (DFSLGNX0) or by specifying the LOGOND parameter.
  - If your installation has no SLUTYPE1 terminals, rename the DFSNTO logon descriptor to DFSSLU1 in order to make it the default for LU type X'01' terminals.
2. IMS does not distinguish between SLU type-P, Finance, and 3601 terminals; SLUTYPEP is the default. ETO considers Finance and 3600 series terminals to be the same. To support both SLUTYPEP and 3600/Finance terminals, do the following:
  - Override the DFSSLUP default logon descriptor by using the Logon exit routine (DFSLGNX0) or by specifying the LOGOND parameter.
  - If your installation has no SLUTYPEP terminals, rename the DFSFIN logon descriptor to DFSSLUP in order to make it the default for LU type X'00' (TS = X'04') terminals.

The default logon descriptor is determined in one of three ways:

- IMS looks for the most common logon descriptor created during IMS system definition and uses it as the default.
- If no terminal definition exists for that terminal type, IMS creates a default logon descriptor.
- Your installation can create a default logon descriptor.

### Using NTO, 3600/Finance Terminals

Because of conflicting CINIT information, IMS cannot generate DFSNTO or DFSFIN as a default logon descriptor name. CINIT parameters for SLU 1 are identical to NTO, and SLU P is identical to 3600/Finance.

**Recommendations:** For 3600/Finance and NTO terminal types, take each of the following actions:

- Always supply the appropriate logon descriptor name in the Logon exit routine (DFSLGNX0). You can generate this name as a constant, based on LU name, LU type, or CINIT user data.
- Ensure that the logon descriptor name is provided as the LOGOND parameter from the BIND user data.
- Ensure existence of a logon descriptor that matches the node name of the terminal that is logging on. If IMS does not find a default logon descriptor, the logon attempt fails.

### Recovering ETO Terminals Using XRF

To recover ETO terminals in an XRF environment, use the BACKUP= parameter when specifying the logon descriptor. The BACKUP= parameter sets the priority that controls the order in which IMS informs NCP (using VTAM) to switch sessions.

When 3600/Finance and SLU-P terminals are defined as class-2 terminals in an XRF environment, automatic re-logon and re-signon occur during a takeover.

**Related Reading:** For general information on XRF, see *IMS/ESA Administration Guide: System*.

## Coding ETO Descriptors

### Creating User Descriptors

User descriptors provide information relating to user options and user structure names. IMS needs user descriptors in order to create control blocks that enable users to use ETO terminals.

The three types of user descriptors are:

- Installation-created
- Node user
- DFSUSER

IMS chooses the first valid descriptor, using the sequence above. IMS creates DFSUSER and node user descriptors using system definition options.

This section describes how to create user descriptors.

#### User Descriptor Format

The format for all three types of user descriptors is:

U username KEYWORD=parm1 KEYWORD=parm2...KEYWORD=parmN

**U** Indicates the descriptor type is user.

##### username

The name of the descriptor, which can be a unique user ID you create, a node user name assigned to a TERMINAL or SUBPOOL macro, or the IMS default descriptor name, DFSUSER.

##### KEYWORD

One of the following keywords:

- ASOT=
- AUTLDESC=
- AUTLGN=
- AUTLID=
- AUTLMOD=
- LTERM=
- OPTIONS=

**parm** A parameter value for the keyword

**Related Reading:** For more information on valid parameter values, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

**Example:** An example of a user descriptor specification is:

Column	Column
1	12
U SMITH	ASOT=20 LTERM=(SECLT1) AUTLGN=SEC01LU2

#### Creating User Descriptors at System Definition

User descriptors are generated from each VTAM TERMINAL or VTAMPOOL SUBPOOL macro. User descriptors that are created during system definition by using the VTAM TERMINAL macro have the same name as the terminal. User descriptors that are created by using the VTAMPOOL SUBPOOL macro have the same name as the subpool.

For user descriptors that are created by using a SUBPOOL macro, you cannot set a response option (TRANSRESP, NORESP, FORCRESP), because it is defined on the TERMINAL macro for static definitions. You need to add the response option (appropriate for your installation) to any user descriptor that is created with a SUBPOOL macro.

### Criteria for Selecting User Descriptors

When a user signs on, if the user structure does not exist, IMS selects a user descriptor to build the user structure. IMS selects the user descriptor according to the following criteria:

1. An installation-written exit routine can select the name of the user descriptor (user ID, node name, or DFSUSER).
2. IMS looks for USERD, provided at logon. IMS also looks for a descriptor name (user ID, node name, or DFSUSER) that is specified in the USER descriptor field of the /SIGN or /OPNDST command.
3. IMS looks for a descriptor that has the same name as the user ID (installation-created user descriptor only). If IMS finds one, and an LTERM keyword is not specified in the descriptor, IMS creates a user structure and a single LTERM, both of which have the same name as the user ID.
4. IMS looks for a descriptor that has the same name as the VTAM node (node user descriptor). If IMS finds one, and an LTERM keyword is not specified in the descriptor, IMS creates a user structure and a single LTERM, both of which have the same name as the VTAM node. However, no output security is associated with this user structure. Any user that signs on at a terminal can receive messages that are queued for that terminal.
5. IMS selects the default user descriptor, DFSUSER. IMS creates a user structure and a single LTERM, both of which have the same name as the user ID.

### Using Installation-Created User Descriptors

You can create your own installation-specific user descriptors that meet important criteria for your site. The name of the installation-created user descriptor is the same as the user ID. You can add values to these user descriptors that are not provided with node user descriptors or DFSUSER descriptors.

### Using Node User Descriptors

Node user descriptors help in migrating from static terminal definitions to ETO dynamic terminal definitions. During IMS system definition, node user descriptors are created as an option. Node user descriptors can be useful when exit routines that perform descriptor selection are not yet complete.

Node user descriptors must have the same name as their associated terminals. Therefore, IMS creates unique node user descriptors that retain user options and user structure names that exist in the system definition. When IMS system definition creates ETO descriptors, a node user descriptor is created for each terminal that has a VTAM TERMINAL macro or VTAMPOOL SUBPOOL macro. Node user descriptors are created even when they match the DFSUSER options.<sup>20</sup>

If the LTERM parameter of the node user descriptor contains the same name as a statically defined LTERM, the node user descriptor is ignored. IMS issues message

---

20. No descriptor is created for an ISC terminal that is defined for parallel session support.

## Coding ETO Descriptors

DFS3673W. Consequently, all node user descriptors that are created during system definition are generated as comment statements. To use the node user descriptors, remove the asterisks.

In most cases, node user descriptors are not needed. You only require node user descriptors when the default user descriptor (DFSUSER) or the Signon exit routine (DFSSGNX0) cannot supply the user options you desire. Using a TSO or an MVS utility, you can discard most of the node user descriptors that are created during system definition.

Be aware that when you use node user descriptors, you cannot use ETO's ability to eliminate the need for predefining terminals that connect to IMS. Using node user descriptors lose output security for each user as well.

### **Recommendations:**

- If continuous availability of IMS is critical, eliminate node user descriptors as soon as possible. Because adding new terminals by using node user descriptors requires an IMS restart, use exit routines instead of node user descriptors.
- To avoid using node user descriptors, use the Signon exit routine (DFSSGNX0) to have IMS build a user structure using the node name as the user name. Although this is similar to using node user descriptors, it avoids predefining large numbers of these descriptors at IMS initialization.
- You can use node user descriptors to start a session when output is available, using /OPNDST, LOGON, or autologon. To use autologon, specify the SHARE option on the TERMINAL macro.
- If you select a default user descriptor using the Signon exit routine (DFSSGNX0), you can set a bit to indicate that IMS should create the user structure using the node name for the user structure name. This is the same as selecting a node user descriptor. By creating node-name structures by using the Signon exit routine, rather than by defining a node user descriptor for each terminal, large network installations benefit from decreased complexity.

## Using DFSUSER User Descriptors

If IMS does not find a user descriptor that has the same name as the user ID or the terminal that is signing on, and no exit routine has provided one, IMS uses DFSUSER as the default descriptor. IMS uses DFSUSER for both signon and application program output processing. Using DFSUSER also enables IMS to dynamically create a user structure for a signon request when no other user descriptor is available.

IMS creates DFSUSER when you specify the system definition options to create ETO descriptors. DFSUSER contains the options specified most frequently in your IMS system definition, and it should satisfy most users. However, using the DFSUSER descriptor does not require you to use all of these options. You can code the Signon exit routine (DFSSGNX0) to make changes to the structures that are built by using DFSUSER.

DFSUSER builds a user structure that has the same name as the user ID that is specified on the /SIGN command. IMS allocates to this user structure a single LTERM, which also has the same name as the user ID. You can use the Signon exit routine (DFSSGNX0) in order to supply multiple queues, if necessary.



**Recommendation:** Fully implementing ETO involves creating most user resources using DFSUSER. After migrating to ETO, use the DFSUSER descriptor for as many users as possible. Minimizing the number of descriptors reduces the administrative workload of an IMS network.

The user structure name becomes the user ID. When you use the DFSUSER descriptor, you can modify user structure names and other options by using the Signon exit routine (DFSSGNX0).

**Related Reading:** For more information on using the Signon exit routine (DFSSGNX0), see *IMS/ESA Customization Guide*.

If an application program has a dependency on an LTERM name that exists in an I/O PCB, you can customize ETO in one of two ways:

- Provide a user descriptor for each user that is to use a particular application program.
- Use the Signon exit routine (DFSSGNX0) to tailor the default action for a subset of the user population.

The Signon exit routine (DFSSGNX0) and the Output Creation exit routine (DFSINSX0) enable you to dynamically create user structures, even when the specified user descriptors do not contain data to create LTERMs.

**Related Reading:** For more information on the Signon exit routine (DFSSGNX0) and the Output Creation exit routine (DFSINSX0), see *IMS/ESA Customization Guide*.

## Creating MFS Device Descriptors

With MFS device descriptors, you can define screen size and feature combinations that are not generated during IMS system definition. The MFSDCT utility (DFSUTB00) uses the MFS device descriptors in order to update device type, screen size, and features in the MFS device characteristics table. MFSDCT also uses MFS device descriptors to generate new MFS default formats, without requiring a system generation. The MFS device descriptors are not created as part of a system generation, but are instead an optional step after IMS system generation.

MFS device descriptors define terminal characteristics of dynamic terminals that differ from static terminals. Unless you use exit routines to override screen sizes, you need to ensure that all ETO terminal screen sizes that are different from those that are defined for static terminals are defined using MFS device descriptors.

**Recommendation:** If you have users who log on to terminals that have different device characteristics, you need to create an expanded set of MFS formats. Input and output messages that are delivered to non-originating terminals are formatted to the terminal according to the MFS formatting specifications that are defined for that device.

### MFS Device Descriptor Format

The format for an MFS device descriptor is:

```
D descname KEYWORD=parm1 KEYWORD=parm2 KEYWORD=parm3
```

**D** Indicates that the descriptor type is MFS (device).

**descname**

The name of the descriptor.

## Coding ETO Descriptors

### KEYWORD=

One of the following keywords: FEAT=, SIZE=, TYPE=.

**parm** A parameter value for the keyword.

**Related Reading:** For more information on valid parameter values, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

**Example:** An example of an MFS descriptor is:

```
Column      Column
1           12
D IMSTYP22  TYPE=3270-A04 SIZE=(43,80) FEAT=IGNORE
```

**Related Reading:** For more information on the values for the FEAT, SIZE, and TYPE keywords on the TERMINAL macro, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

### Building the Device Characteristics Table

When someone logs onto an ETO 3270 or SLU-2 terminal that is connected to IMS, usually the screen size is provided to IMS in the VTAM CINIT PSERVIC data. Sometimes the model number is also provided. IMS uses this screen size and the features that are specified on the logon descriptor in order to search the MFS device characteristics table, which contains one or more entries. Each entry identifies the following:

- The MFS device type
- The screen size
- The features

Searching the table, you can find the MFS device type. If the search is unsuccessful, the logon attempt is rejected, and messages DFS3646 and DFS3672 are issued.

The MFS device characteristics table is built in one of two ways:

- The IMS system definition process builds an MFS device characteristics table based on the TYPE=, SIZE=, and FEATURE= specifications of the TERMINAL macro. Each unique combination has an entry in the MFS device characteristics table.
- The MFSDCT utility (DFSUTB00) builds or modifies an MFS device characteristics table based on the input of the MFS device descriptor. These descriptors have the TYPE=, SIZE=, and FEATURE= specifications that the TERMINAL macro has. When the VTAM CINIT PSERVIC indicates a model number instead of a screen size, the MFS device characteristics table is not searched. A model number indicates a certain screen size and MFS device type.

The MFS device characteristics table is not searched if the VTAM CINIT PSERVIC field contains a model number instead of a screen size.

**Related Reading:** For more information on using the MFSDCT utility (DFSUTB00), see *IMS/ESA Utilities Reference: Transaction Manager*.

### Using the MFSDCT Utility (DFSUTB00)

During session initiation, IMS searches the device characteristics table in order to locate MFS device information, in either of the following situations:

- The screen size is overridden.
- The model value in PSERVIC is X'00', X'7E', or X'7F'.

## Coding ETO Descriptors

The MFSDCT utility (DFSUTB00) enables you to define screen sizes that are not defined during IMS system definition. The MFSDCT utility uses MFS device descriptors in PROCLIB member DFSDSCMx and DFSDSCTy, without performing an IMS system generation. The new screen size definitions are added to those that are already defined.

The MFSDCT utility procedure is found in IMS.PROCLIB. You must evaluate the screen size requirements and code MFS device descriptors to meet those requirements. The MFSDCT utility must generate all possible combinations of screen sizes and features that your installation might require.

To use the MFSDCT utility, follow these steps:

1. Execute DFSUTB00 (the MFSDCT utility). The MFS utility does the following:
  - Reads device descriptors from IMS.PROCLIB members DFSDSCMx and DFSDSCTy.
  - Builds one device descriptor table entry statement for each new device descriptor.
  - Terminates if no descriptors are specified.
  - Optionally loads the existing device characteristics table from IMS.RESLIB, and builds one entry statement for each existing table entry.
  - Passes the table entry statements and the DCTBLD and MFSINIT macros as input to assembler.
  - Prepares assembler output as a new or updated device characteristics table, as well as a new set of default MFS format definitions. Output is in separate files for subsequent processing.
2. Assemble the new device characteristics table.
3. Link edit the new device characteristics table into IMS.RESLIB.
4. Execute phase 1 of the MFS Language utility in order to generate new default MFS format control blocks.
5. Execute phase 2 of the MFS Language utility; the new default MFS formats are loaded into IMS.FORMAT.

You can specify any of the following parameters in the PARM field of DFSUTB00:

### **DCTSUF=x**

Specifies the suffix character that is to be appended to member DFSUDT0x. This suffix indicates which device characteristics table is to be used for input that is merged with the device descriptions.

If this suffix is not specified, IMS builds a new device characteristics table.

### **DSCTSUF=x**

Indicates which member DFSDSCTx is to be used for input. This parameter is required if DSCMSUF is not specified.

### **DSCMSUF=x**

Indicates which member DFSDSCMx is to be used for input. This parameter is required if DSCTSUF is not specified.

### **DEVCHAR=x**

Specifies the suffix character that is to be appended to the name DFSUDT0x. This name becomes the new name for the new or updated table. The default is 0 (zero). Any existing table with the same name is replaced.

## Coding ETO Descriptors

Problems with the device characteristics table are often indicated (by error messages DFS3646 and DFS3672) during logon processing.

## Creating MSC Descriptors

MSC descriptors relate remote LTERMs to statically defined MSC links. IMS creates MSC descriptors when you specify IMS system definition options to create ETO terminal descriptors. MSC descriptors relate remote NAME macros to defined MSC links.

**Recommendation:** Define all LTERMs in remote IMS systems by using MSC descriptors. The MSC Verification utility in the target IMS system can then associate the remote LTERMs with the corresponding local LTERMs. IMS issues message DFS2331W if the corresponding local LTERMs are not found.

IMS only processes those MSC descriptors that are associated with the MSC links that are defined within the system being initialized. IMS ignores all other MSC descriptors. If each logical link path name is unique in the network, you can maintain your network's MSC definition source in a single PROCLIB member, IMS.PROCLIB.

### MSC Descriptor Format

The format for an MSC descriptor is:

```
M linkname name1 name2...nameN
```

<b>M</b>	Indicates the descriptor type is MSC (device).
<b>linkname</b>	The name of the link path from an MSNAME statement.
<b>name</b>	Any valid remote LTERM name that is accessed through the linkname.

**Example:** An example of an MSC descriptor is:

```
Column      Column
1           12
M REMSYS01  REM01AAA  REM01BBB  REM01CCC  REM01DDD  REM01EEE  REM01FFF
M REMSYS02  REM01GGG  REM01HHH  REM01III
```

---

## Using Exit Routines

You can use the following exit routines to customize ETO:

- Initialization exit routine (DFSINTX0)
- Logon exit routine (DFSLGNX0)
- Logoff exit routine (DFSLGFX0)
- Signon exit routine (DFSSGNX0)
- Signoff exit routine (DFSSGFX0)
- Output Creation exit routine (DFSINSX0)
- Greetings Message exit routine (DFSGMSG0)

If ETO is enabled, these exit routines are loaded during IMS initialization. All non-MSC and non-LU 6.2 VTAM terminals (static or dynamic) can use these exit routines.

You can code these ETO exit routines with a wide range of processing logic for any of the following purposes:

- Enforcing naming conventions
- Selecting user queues
- Overriding terminal characteristics, such as screen size
- Enhancing security by limiting access to IMS terminals
- Overriding security by allowing signon without using a security product, such as RACF
- Selecting operational parameters, such as timeout values

**Related Reading:** For more information on coding these exit routines, see *IMS/ESA Customization Guide*.

---

## Starting ETO

To include ETO on your IMS system, code the ETOFEAT parameter on the IMSCTRL system definition macros during system definition. At initialization time, specify ETO=Y on the execution parameter to start ETO.

The IMS system execution default is ETO=N. If you specify ETO=N, IMS rejects requests to establish sessions for undefined terminals. Messages that are destined for nonexistent queues are refused, and IMS issues message DFS064 or an A1 status code.

When IMS initializes, it calls the Initialization exit routine (DFSINTX0). You can code this exit routine to disable ETO, or to create and load data that you want ETO to use. IMS maintains a pointer to this data and passes this pointer to the ETO exit routines as an input parameter. IMS passes this pointer to non-ETO exit routines through the SCDINTXP field in the system contents directory (SCD).

When you enable ETO, IMS validates each ETO descriptor:

- If a descriptor is coded incorrectly, IMS ignores it and issues message DFS3640W to the MTO.
- If IMS detects an invalid parameter within a valid descriptor, IMS substitutes the default parameter, and processing continues with message DFS3641W.
- If IMS is unable to read all the descriptors (for example, because of read errors), IMS abends with abend U0015. IMS does not abend if it finds one valid logon descriptor and one valid user descriptor.
- IMS does not start a system whose descriptor information is incomplete.

**Related Reading:** For information on messages and error codes that might be issued at initialization time, see *IMS/ESA Messages and Codes* and *IMS/ESA Failure Analysis Structure Tables (FAST) for Dump Analysis*.

Descriptors, exit routines, and the MFS device characteristics table can be added, deleted, or updated before IMS initialization time. However, if a control block for a session exists across restart, a change in the descriptor that built the control block does not take effect until after the control block is deleted. For example, 3600/Finance, SLU-P, and ISC sessions can warm start with control blocks created from an original descriptor, even when that descriptor is changed or deleted after the control blocks were created.

### Logging onto ETO Terminals

You can log on to your terminal in three ways. You can use:

- The IMS /OPNDST command, and optionally include signon data.
- The SNA commands INITSELF, INITOTHER, or USS LOGON, and optionally include user data for signon.
- ETO autologon, which includes user data based on user descriptors or exit routines. The user data is then passed to signon.

If you specify ETO=Y, you can establish sessions with ETO terminals, but only if each of the following is true:

- An available logon descriptor provides sufficient information to create the control blocks necessary to accept the session request.
- In addition to the required logon descriptor, a 3270 or SLU-2 terminal requires an entry in the MFS device characteristics table that matches its screen-size and features. However, IMS does not search the MFS device characteristics table if the 3270 or SLU-2 terminal is identified as model 1 or 2.
- If the screen-size control byte is X'7F' in the PSERVIC field of the VTAM MODEENT macro, IMS searches the MFS device characteristics table using the alternate screen-size. If no match is found, IMS searches the MFS device characteristics table using the default screen-size. If you specify both sizes, and you want to use the default screen-size, the Logon exit routine (DFSLGNX0) must specify the default screen-size as an override.
- If the user cannot specify the logon descriptor in the user data, you must use the Logon exit routine (DFSLGNX0) or let IMS choose the default logon descriptor based on the terminal type.

### Limiting Dynamic Logon To Specific Terminal Types

If you want to limit dynamic logon to specific terminal types, delete the default logon descriptor for those terminal types you do not want to logon dynamically. You can also reject the default logon for specific terminal types by using the Logon exit routine (DFSLGNX0). Dynamic logon for these terminal types fails.

**Related Reading:** For information on selecting default logon descriptors, see “Criteria for Selecting a Default Logon Descriptor” on page 140.

### Creating and Reusing LTERM Control Blocks

A user structure containing an LTERM is created:

- When the user signs on, and the user structure does not currently exist.
- When an asynchronous message is created for an LTERM, and the LTERM does not currently exist.

The user structure is allocated to a terminal after successful signon.

### Using Default CINIT or BIND User Data Formats

Each request for session initiation can include VTAM CINIT or BIND user data to provide logon descriptor or signon data. Your installation can provide a logon exit routine to process this data.

## Logging onto Your ETO Terminal

IMS can receive optional user data when you establish a session using one of the following methods: <sup>21</sup>

- Using an IMS /OPNDST command
- Using autologon
- The RTO provides a user logon

You can expand the user data formats to meet your own requirements. You can either supply the logon descriptor name in your logon exit routine by using user data, or you can create the logon descriptor name by using an IMS algorithm.

The user data appears in the CINIT user data field, and it is available to IMS when the VTAM Logon exit routine is scheduled. One optional parameter, the logon descriptor name, applies to the IMS logon process. The remaining parameters apply to the IMS signon process and, optionally, to RACF. During either process, IMS does minimum processing on the CINIT user data parameters before first calling the optional installation Logon exit routine (DFSLGNX0), and later calling the Signon exit routine (DFSSGNX0).

Although the Logon and Signon exit routines can translate any installation-defined user data format, IMS has defined a default user data format that:

- the installation can expand.
- IMS can process in the absence of exit routines. This format is the logon descriptor name followed by signon data in the same format as the IMS /SIGN command.

**Restriction:** For warm session initiation of an STSN device, the user data must be the same as in the original logon.

### **Related Reading:**

- See *IMS/ESA Customization Guide* for more information on:
  - The Logon and Signon exit routines
  - The format of the CINIT user data
  - Restrictions that apply to ISC
- See *IMS/ESA Operator's Reference* for more information on the /SIGN command.
- For information on the default user data format, see "Appendix D. Format for CINIT User Data Parameters" on page 485.

---

## Signing On and Queue LTERM Allocation

**Definition:** *Signing on* to an ETO terminal identifies a user to IMS, creates a user structure, and connects this user structure to the terminal structure.

Users at VTAM terminals can sign on by:

- Issuing the /SIGN command.
- Signing on at the DFS3649 message screen.
- Providing user data with the session initiation request.

---

21. ETO ISC terminals do not have optional data available. ISC supports an SNA format defined in the *IMS/ESA Customization Guide*.

## Signing On and LTERM Allocation

Users that establish a dynamic VTAM session with IMS must enter valid signon data before LTERMs are allocated to the session. You can require that the signon data be validated by a security product, such as RACF.

## Providing Signon Data

Users can enter signon data by using one of the following methods:

- Using the /SIGN command
- Including the signon data with the logon user data
- Coding the Logon exit routine (DFSLGNX0)

## Providing Signon Data for ISC, SLU-P, Finance, and Output-Only Devices

For dynamic ISC, SLU-P, Finance, and output-only devices, you must provide user signon data at session initiation. IMS validates the signon data and allocates LTERMs to the terminals, based on:

- User descriptors in IMS.PROCLIB
- Applicable LTERM and user option defaults
- Signon exit routine (DFSSGNX0) output

**Restriction:** A user cannot enter signon data from an output-only device during logon or when using the /SIGN command. For a dynamic output-only terminal, users must enter signon data at session initiation.

If a user forgets to include signon data during session initiation for an output-only device, IMS issues message DFS2085 (with return code 264) to the MTO. If signon data is omitted for dynamic ISC, SLU-P, or Finance terminals, IMS issues messages DFS3645 and DFS3672.

## Signing On Multiple Times

Users can concurrently sign on to one or more terminals. These terminals can be a combination of both static and dynamic terminals. For both static and dynamic terminals, you must specify the EXEC parameter, SGN=M (multiple signons enabled).

For dynamic terminals, the name of the user structure that represents the user to IMS must be unique. You can use one of four methods to make the user structure unique:

- Use the Signon exit routine (DFSSGNX0) to return a 1- to 3-byte suffix to the user ID. The total length of the user ID plus the suffix cannot exceed eight characters.

**Recommendation:** Use a naming convention that ensures unique user IDs. For example, you can create suffixed user IDs using the Signon exit routine (DFSSGNX0), and check the uniqueness of these user IDs by using IMS Callable Services.

**Recommendation:** If you are operating in a Sysplex environment, create a naming convention that ensures unique user IDs across the Sysplex by creating a suffix from the following:

- As the first part of the suffix, the Signon exit routine can attach the unique part of the IMS system identifier, which is specified on the IMSID parameter, to the user ID.



## Signing On and LTERM Allocation

- As the second part of the suffix, use IMS Callable Services to choose an available value for the user on that IMS system.

**Example:** A user with a user ID of USER signs on to IMSA. A unique suffix of A1 is created by:

1. The Signon exit routine appends the A from IMSA to the user ID.
2. IMS Callable Services appends a 1, representing the first user named USER to sign on to IMSA.

In this example, user IDs must be less than seven characters in length.

- Use the Signon exit routine to specify that the user structure name is the same as the node name.
- Use the Signon exit routine to specify a name that is unrelated to the user ID or the node name.
- Specify a user descriptor name that is the same as the node name, causing the name of the user structure to be the same as the node name. You can do this by using one of the following methods:
  - Enter signon data that contains the user descriptor name.
  - Use the Signon exit routine to specify the node user descriptor.
  - Specify no user descriptor and let IMS use the node user descriptor by default.

Unless you specify SGN=M to enable multiple signons, the user ID for a dynamic user must be unique. If the user ID is used as the user-structure name, it must be unique, regardless of whether you specify SGN=M. If the user ID is not used as the user-structure name (for example, by using the Signon exit routine), you must specify SGN=M for multiple signons.

**Recommendation:** Assigning a single name to an IMS user is useful in determining output status. If you use multiple names for individual users (for example, when the Signon exit routine assigns them), you must provide a means to determine the user names that are created by signons.

**Restriction:** You cannot use the same name for a dynamic LTERM and a static LTERM that is defined during system definition.

**Related Reading:** For information on the Signon exit routine (DFSSGNX0) and IMS Callable Services, see *IMS/ESA Customization Guide*.

## Receiving the Signon Required Message—DFS3649A

The ETO user must sign on before a session can enter transactions or commands. The user can enter the /RCLSDST command only before signing on. IMS issues message DFS3649A (indicating that a /SIGN command is required) in each of the following situations:

- After a signon failure
- After a signoff
- After a logon in which no user data is entered

Any terminal user can then issue a /SIGN ON command to begin the next session.

You do not receive the DFS3649A message in any of the following situations:

- Signon data is included in the VTAM CINIT or BIND.
- The session is an ISC, SLU-P, Finance, or 3270 printer session.

## Signing On and LTERM Allocation

- The terminal is a SLU–1 terminal running in unattended mode.
- The terminal is the subject of an autologon attempt.

### **Related Reading:**

- For more information on receiving message DFS3649A, see *IMS/ESA Messages and Codes*.
- For more information on the /SIGN ON command, see *IMS/ESA Operator's Reference*.

## Receiving the Session Status Message—DFS3650

After a user successfully signs on (or if signon is not required), IMS issues message DFS3650, indicating the status of the session with IMS.

**Exceptions:** In the following situations, IMS does not send message DFS3650:

- When SLU–1 terminals run in unattended mode.
- When you specify the NOTERM option on the user descriptor.

Message DFS3650 provides information on session status, such as:

- Whether the user is in conversation mode
- Whether user output security exists for the terminal

**Related Reading:** For more information about when message DFS3650 is issued, see *IMS/ESA Messages and Codes*.

## ETO Terminal-LTERM Relationship

The system programmer is responsible for the relationship between a terminal that is in session with IMS and a particular user's LTERMs:

- For a single-component terminal (such as a SLU 2), IMS creates a single default LTERM name that is the same as the user ID that was supplied during signon. Using an exit routine or a user descriptor, you can give the LTERM a different name.
- For a multicomponent terminal (such as a SLU P), you need to ensure that sufficient LTERMs are created in order to use that terminal. You can use exit routines, installation-created descriptors, or node user descriptors<sup>22</sup> to ensure that a sufficient number of LTERMs is created. Otherwise, IMS creates a single LTERM that is allocated to the first component of the terminal.

It is possible for an application program to associate a specific LTERM name with a specific terminal. The system programmer must ensure that names of alternate PCBs are consistent with LTERM names defined in:

- User descriptors
- Signon exit routine
- Insert exit routine
- Recovery requirements for ISC, SLU-P, and Finance sessions

---

22. The user needs to use the node user descriptor to establish the correct relationship between the LTERM and the node, either explicitly or in the Signon exit routine (DFSSGNX0).

## How IMS Determines Which Queues to Allocate

IMS uses the following method to determine which LTERMs to allocate:

1. If the user control blocks already exist within IMS, these control blocks are reallocated, and the Signon exit routine is called, if appropriate.
2. If the control blocks do not exist and a user descriptor is specified at signon, IMS looks for the specified descriptor.
  - If the descriptor specified is not the user ID, node name, or DFSUSER descriptor, IMS sends error message DFS3649A (with return code 148).
  - If the descriptor specified is a valid user descriptor for the user that is signing on, IMS builds a table of available descriptors and calls the Signon exit routine, if appropriate.
  - If the Signon exit routine exists and RC=0, IMS continues normal signon processing. If the return code does not equal 0, the signon is rejected.
3. If no user descriptor is specified at signon and no user descriptor is specified in the Signon exit routine, IMS chooses a valid descriptor from the following (in order):
  - a. User ID
  - b. Node name
  - c. DFSUSER

**Related Reading:** For more information on the Signon exit routine (DFSSGNX0), see *IMS/ESA Customization Guide*.

## Setting Special Processing Modes

After user signon, you can set the following processing modes by using IMS commands:

<b>Exclusive mode</b>	/EXCLUSIVE command
<b>Preset destination mode</b>	/SET command
<b>MFS test mode</b>	/TEST MFS command
<b>Test mode</b>	/TEST command

These special processing modes are retained for the user after signing off and are reestablished with the next terminal on which the user signs on.

When you issue the following commands, IMS creates the required control blocks for a terminal or user, or it maintains a user's status:

<b>/EXC USER</b>	Places a user structure in exclusive mode.
<b>/STOP NODE or /STOP USER</b>	Stops a node or user structure.
<b>/TEST MFS USER</b>	Places a user structure in MFSTEST mode. When a user signs on, IMS places the terminal in MFSTEST mode, if the terminal supports MFS.
<b>/TEST USER</b>	Places a user structure in test mode.
<b>/TRACE NODE</b>	Traces a logged-on node.

## Signing On and LTERM Allocation

To reset terminal status and make the control blocks eligible <sup>23</sup> for deletion at the next simple checkpoint, use the following commands:

<code>/END</code>	Clears exclusive and test modes.
<code>/RESET</code>	Removes the preset destination.
<code>/RSTART</code>	Starts stopped resources.
<code>/START</code> <sup>24</sup>	Starts stopped resources.
<code>/TRACE</code>	Sets trace off.

Use the preceding set of commands if the control blocks exist solely for status retention.

You can use the Output Creation exit routine (DFSINSX0) to reset these states.

**Related Reading:** For more information on deleting control blocks, see “Improving Performance by Deleting ETO Control Blocks” on page 166.

---

## Using ETO with Printers

Two methods of implementing printer support exist in an ETO environment:

- Direct printing
- Associated printing

How you implement these two methods depends on how your application programs identify printer LTERM names.

### Direct Printing

**Definition:** *Direct printing* is a printing technique by which application programs insert messages to the VTAM LU name. The dynamic LTERM and the user and terminal resources are all named after the VTAM LU name. Many application programs can queue data to the same printer LTERM, but this can create data interleaving problems.

### Associated Printing

**Definition:** *Associated printing* is a technique for directing application program printer output to a specific printer node name.

#### How Associated Printing Works

For associated printing, application programs insert messages to the queue that is associated with the screen-user queue name. User printing requests cause application programs to queue data to different printer LTERMs. This avoids data interleaving problems.

Printer signoff and signon are required to change the user queue. Overhead can be high if the printer has many users.

**Recommendation:** Carefully plan how your printers are to be shared among application programs.

---

23. Control blocks are not immediately deleted when special status is removed. They are deleted at the next checkpoint if they meet all deletion requirements. If an outstanding status exists, they are not eligible for deletion.

24. See the *IMS/ESA Operator's Reference* for other status reset by the `/START` command.

By implementing exit routines and application programs, the terminal operator can provide the destination during logon or signon.

### Identifying Printer Node Names

You can identify printer node names in one of two ways:

- As logon user data, you can include one or more printer node names when a user establishes a session.
- Your installation can modify the MFS format for the DFS3649A greeting message in order to allow the user to supply the printer node names at signon time. During the signon, however, the Signon exit routine must be able to detect the printer node names as user data.

### Coding the Signon Exit Routine for Associated Printing

To use associated printing, code the Signon exit routine (DFSSGNX0). Code the Signon exit routine to do each of the following steps:

- Identify the printer LU name and user name for each screen user.
- Determine printer node names from the input user data.
- Name user-related LTERM structures to service the selected printers.
- Enable application programs to determine (using the user ID) the queues associated with a particular user so that the programs can insert to the correct message queue (alternate PCB).
- Pass the printer node names to IMS as associated print parameters.
- Determine the user name that is allocated to each printer when users supply printer node names. You can use either an algorithm or a table to make this determination. The exit routine should pass these user names to IMS, which then creates the necessary user control blocks.

A unique name should exist for the user ID that is signed on and for each printer-related user that is required.

**Example:** To indicate the name of a user's printer identification, add a "P" to the end of the user ID. If the user ID is AAA, the name of this user's printer identification is AAAP.

- Specify one value for each of up to four printers (the number of simple checkpoints before the user structure is deleted).

**Related Reading:** For more information on the Signon exit routine (DFSSGNX0), see *IMS/ESA Customization Guide*.

The application program can use the same method as the Signon exit routine in order to determine the printer LTERM name from the input terminal user ID and queue output data as required through an alternate PCB. When output data is queued, IMS allocates the user structure to the correct printer and delivers the output.

After associated printer LTERMs are allocated and then emptied, the queues are deallocated from the terminal.

## Defining Your Printers

Printers are usually output-only devices; however, they can be implemented so that they send input.

You need to decide whether to have single-user or multiple-user structures share the same printer. If your application programs are sensitive to queue names, you might be limited to one or the other approach.

## Using ETO with Printers

### Single-User Structure

Using a single-user structure that represents a printer is the simplest method for defining a printer. In this case, messages sent to the printer are printed in the sequence in which they originate. Interleaving of output can occur. Multisegment messages always have all segments printed in contiguous sequence. Multiple messages might behave differently, depending on their method of origination.

### Multiple-User Structures

Multiple users are supported for printers. All messages for a single user are printed, and the next user is selected for printing only when the current user has no more output. Although interleaving of messages for the same user occurs as above, the switching of users can also be equivalent to message interleaving from an application standpoint. Application and operational awareness of the way printers are shared is important.

The problem of sharing printers is not unique to dynamic terminals, and in fact are the same as for static terminals.

IMS prints a separator page whenever the next message to be printed is from a different user. You can control the contents of this separator page by using an exit routine to change its content; however, this page is always printed, even if blank. Messages from the same user are not separated by a separator page. Using the NOTERM option can prevent the DFS3650 separator message from being used.

## Sharing Printers Using ETO

Several users can use the same output terminal:

- Users can define a single-user descriptor that contains all the LTERM names that are used to deliver data to an output device (node).
- Users can define autologon parameters for a node.

Before deciding which of these methods to use, ask these questions:

- Are the LU-to-LTERM relationships generally unchanging?
- Is interleaving at the message level acceptable?
- Is delaying one user's output acceptable while another user's output is being printed?
- Is continuous autosignoff and autologon processing overhead too excessive for a particular terminal because of the minimal message delivery rate of each LU user?

The answers to these questions can help to determine your method of implementation:

- If the answer to all of the questions is "yes", consider creating a single multi-LTERM user for the printer.
- If any of the answers is "no", consider dynamic allocation of multiple-user LTERMs, using autologon.
- You can implement a combination of these two options.

---

## Using Operator Commands

This section describes using the /OPNDST and /ASSIGN commands as they are used with ETO.

## Using the /OPNDST Command

The VTAM mode table that is used when the terminal first logs on determines the device characteristics. However, if the first reference to a terminal is through an /OPNDST command, the MODETABLE operand of that command determines the device type.

Omitting the MODETABLE operand on the command causes it to default, which might not be desirable. After the terminal is in session with IMS, the rules for block deletion apply. The device type remains set until the block is deleted; if the terminal is closed and then reopened, IMS uses the existing block, if available. If a block is deleted, that block is rebuilt during the next terminal logon. As a result, the /OPNDST command can have different results, depending on two things:

- Whether the block is still available (and has not been deleted).
- Whether the exit routine or descriptor has been changed since the previous initialization.

## Using the /ASSIGN Command

You can use the ASSIGN command to move queues from one terminal to another. You cannot use it to reassign a static terminal to a dynamic (ETO) terminal, or to assign a dynamic terminal to a static terminal.

**Related Reading:** For information on reassigning user IDs, see the /ASSIGN USER command in *IMS/ESA Operator's Reference*.

---

## Using System Definition Parameters

This section describes the system definition parameters you can use with ETO.

### Using the DLQT Parameter To Set DEADQ Status Time

User control block structures are normally created as a result of any of the following situations:

- When a terminal is logged on and a user signs on.
- When the AO exit routine (DFS AEOU0) inserts a message to an LTERM or transaction.
- When an asynchronous transaction output message is sent, or a terminal message switch or /BROADCAST LTERM command is issued.

Messages might be sent to destinations that are unknown, no longer valid, or nonexistent. IMS sets a status of dead-letter queue (DEADQ) when an ETO user control block structure or its associated message queues have not been accessed within the time limit that is set by the DLQT execution parameter (1-365 days).<sup>25</sup>

The DEADQ status can be set regardless of whether messages have been queued for the user or whether the user is still allocated to a terminal. The DEADQ status is set during IMS checkpoints, and the MTO is notified with message DFS3643.

If the user has messages queued, remove the DEADQ status by signing on the user or by issuing the /DEQUEUE or /ASSIGN command.

---

25. A value of 1 is not usually recommended. It can result in many premature and misleading DEADQ status settings during the first checkpoint.

## Using System Definition Parameters

User control block structures without queued messages can result in a DEADQ status. User control blocks are not deleted if a special status is pending. The status might have been set during the prior signon (such as, response or conversation mode) or as a result of a command (such as, /STOP, /TEST, or /EXCLUSIVE). If the control block remains unused for longer than the time specified for the DLQT execution parameter, IMS assigns the control block a DEADQ status, and the MTO is notified with message DFS3643.

If the user control block has DEADQ status, the status is removed when the user signs on or during the next checkpoint after all messages are dequeued and recoverable status conditions are removed using appropriate commands. In the latter case, the control blocks for the user and associated message queues are also deleted.

User control block structures that have DEADQ status might or might not be allocated to a terminal. In the case of LU type-6, SLU-P, and Finance terminals, the user structure can be allocated to the terminal with no active session. Logoff or other session termination can leave these terminals pending message recovery (SNA STSN). The user remains allocated to the terminal, and messages might or might not be queued.<sup>26</sup>

If the user control block structure is allocated to an LU type-6, SLU-P, or Finance terminal, remove the DEADQ status by logging on and signing on the user, or by using the /DEQUEUE command. For LU type-6 (ISC) terminals use the /DEQUEUE command, or force the LU type-6 session to cold start. A forced-session cold start is a /STO NODE, /ASSIGN (USER TO VTAMPOOL), /STA NODE sequence that is valid only for LU 6. Forced cold start is not possible for SLU-P or Finance terminals. Clearing the allocation of an ETO user to a SLU-P or Finance terminal requires an IMS cold start.

In a shared-queues environment, you can use the /DISPLAY QCNT MSGAGE command to find the age of a queue.

## Using Autosignoff (ASOT)

**Definition:** *Autosignoff* involves deallocating users from an idle session. Users are automatically signed off if no activity occurs within an allotted time. The system programmer sets this time using the ASOT (auto signoff time) parameter in the DFSPByyy member. If users are automatically signed off, they must sign on again in order to use the session.

If more than one allotted time value exists, IMS uses the following criteria to determine which allotted time value to use:<sup>27</sup>

- If the valid ASOT value is specified in the user descriptor, this allotted time value is used.
- If the user descriptor does not specify an allotted time value, the allotted time value from the logon descriptor is used.
- If the logon descriptor does not specify an allotted time value, the time value from the DFSPByyy member is used.

---

26. This combination is not possible for other VTAM terminal types, because a deallocation of the terminal and user ID is forced at logoff and signoff.

27. The time value on the ASOT EXEC parameter does not apply to 3600, SLU P, or ISC devices.



## Using System Definition Parameters

- If the DFSPByyy member does not specify an allotted time value, the default value of 1440 is used.
- If the value on the DFSPByyy member is not valid, the default value of 10 is used.

The Logon exit routine (DFSLGNX0) can override the ASOT and ALOT values during logon. The Signon exit routine (DFSSGNX0) can override the ASOT value during signon, even when the control block structure exists.

The values for the allotted time specified on the ASOT parameter in the DFSPByyy member are:

ASOT = 0

The user is signed-off immediately when no output is available to be sent. This specification is normally used with Autologon terminals for signoff immediately when:

- No IMS input or output message is available
- After the last available output message completes

This specification is not recommended for interactive terminals such as 3270's or SLU2. These terminals sessions normally return a PA key to continue following signon. Idle time results in immediate signoff, not waiting for terminal input.

The value on the DFSPByyy member is not used for these device types.

ASOT = (10 - 1439)

The user is signed off after the allotted number of minutes has elapsed without terminal activity.

ASOT = 1440

The user is never automatically signed off. This is equivalent to not having autosignoff. The system default value for SLU-P, 3600/Finance, and ISC terminals is 1440.

After autosignoff completes, IMS attempts to locate a user that has the same node name that is waiting for autologon. If IMS finds another user with output waiting, the user is allocated to the terminal, and the queues are drained.

## Using Autologoff (ALOT)

**Definition:** *Autologoff* involves terminating a session with IMS for a terminal that has been signed off for an allotted period. After an allotted time, the terminal is automatically logged off. The system programmer sets this time on any of the following:

- On the ALOT (auto logoff time) parameter in the DFSPByyy member
- On the logon descriptor that is used to create the session control blocks
- On the EXEC parameter at initialization

If more than one allotted time value exists, the following criteria are used to determine which allotted time value to use:

- If the ALOT value is specified on the logon descriptor, this allotted time value is used.
- If the logon descriptor does not specify a valid allotted time value, the time value specified in the DFSPByyy member is used.

## Using System Definition Parameters

- If the DFSPByyy member does not specify an allotted time value, the default value of 1440 is used.
- If the value on the DFSPByyy member is not valid, the default value of 10 is used.

You can specify ALOT=1440 on the logon descriptor for a 3600/Finance, SLU-P, or ISC terminal in order to specify that no autologoff is desired. The system default value for these devices is 1440. Logon descriptors created for ISC terminals should have ALOT=1440 specified to show that autologoff should not occur. The Logon exit routine (DFSLGNX0) can override the ALOT value during logon, even when the control block structure exists.

The values for the allotted time specified on the ALOT parameter in the DFSPByyy member are:

### **ALOT = 0**

The terminal is logged-off immediately when no signon is in effect. This specification is normally used in terminal sessions when the user is signed-on automatically during the logon process. During autologon, signon data can be provided in one of the following ways:

- Signon data supplied via the IMS /OPNDST command
- Signon data supplied by logon userdata (BIND)
- Signon data supplied by logon exit (DFSLGNX0)

There are two modes of operation for using ALOT=0, either of which can be set using the DFSINTX0 User Initialization Exit parameter list.

In default mode, when signon errors are encountered, the session is automatically signed off and then logged off; no message is sent. If you do not supply the DFSINTX0 exit, or you supply the exit and indicate default mode for ALOT=0, then signon data must be supplied during the logon process. A typical autologoff scenario might work as follows:

1. A non-signon, or errors detected during signon or input processing, result in immediate logoff.
2. /SIGNOFF results in immediate logoff .
3. /SIGNON signs off the current user and signs on a new user. However, errors encountered during the signon process, such as detection of an incorrect or expired password, result in immediate logoff.

**Restriction:** Default mode should not be used for interactive terminal sessions that require a response to the DFS3649 message; these sessions will not wait for input signon and will logoff immediately.

In alternate mode, when signon errors are encountered, the session is automatically signed off, a message is sent and the session is logged off. Signon data can be supplied but is not required. A typical autologoff scenario might work as follows:

1. A non-signon error detected during input processing results in immediate logoff.
2. No signon data has been provided by the logon userdata (BIND) or the Logon Exit (DFSLGNX0).
3. A /SIGNOFF, or errors resulting from a /SIGNON, cause message DFS3649(A) (Signon Required) to be sent, and a fixed ten-minute timer set to wait for a new signon. If no signon occurs during that interval, then the session will be logged off.

### **ALOT = (10 - 1439)**

The session is terminated after the allotted number of minutes has elapsed without a signed-on user.

### **ALOT = 1440**

The session is never automatically terminated. This is equivalent to not having autologoff.

## Using the Autosignoff and Autologoff Timer

The VTAM I/O Timeout Facility (if active) detects users or sessions that should be automatically terminated. A timer pops at intervals of one minute if the VTAM I/O Timeout Facility is active, or five minutes if the Timeout Facility is not active. The timer starts a routine that determines which resources are due for autosignoff or autologoff. The specified timeout value is the minimum value for which a user or session is automatically terminated. Termination actually occurs the next time the timer pops after the specified timeout value.

## Using Autologon

Instead of using the SHARE option on the TERMINAL macro to request that IMS automatically initiate a terminal session when output is available, ETO offers autologon support for ETO terminals and users. You specify autologon parameters when defining the user to IMS.

**Definition:** *Autologon* allows IMS to log on and sign on your terminal automatically. If you specify the autologon option for a user, the queueing of data to any of the user queues causes IMS to establish a session. You can specify autologon using:

- AUTLGN= parameter on the user descriptor
- Output Creation exit routine (DFSINSX0)
- Signon exit routine (DFSSGNX0) for associated printers

Autologon includes both automatic logon and automatic signon. At restart, IMS attempts to start sessions with those terminals that are defined with autologon and that have queued data waiting. After the session is established, IMS automatically signs on the terminal. If a queue of waiting users exists and the allocated queues are drained, the autologon user is signed off, regardless of an existing ASOT value.

IMS manages a serial queue of waiting users if more than one user is contending for the same autologon terminal. After the autologon user is signed off, the next autologon user for the same terminal is automatically signed on. When all autologon users have signed off, the terminal is free to begin its ALOT cycle in order to terminate the session.

Autologon replaces the TERMINAL macro OPTIONS=SHARE for static terminals. OPTIONS=NOASR (no automatic session restart) on the logon descriptor is ignored for autologon printers. IMS always assumes OPTIONS=ASR for autologon printers.

Autologon is normally specified for output-only terminals. Autologon and occasional users generally do not share the same terminal session, but sharing terminal sessions is possible for interactive terminals. Interactive users on terminals that are subject to autologon must supply user signon data with the session initiation request (logon) in order to avoid contention with autologon output. Occasional terminal users can use autologon in order to have output delivered to default terminals when the output becomes available after signoff. If a terminal is stopped,

## Using System Definition Parameters

the /START NODE command does not start a session. The /OPNDST NODE USER command can be used to restart the session.

---

## Assigning Output

### Asynchronous Output

ETO provides IMS the flexibility to create user structures for any authorized user that is signed on. IMS automatically creates user structures for message switches and for the application insert call (ISRT) process when the LTERM cannot be found.

With ETO, all destinations are valid unless they are rejected by exit routines. Therefore, you can mistakenly create queues for users if you make typographical errors when entering any of the following:

- Alternate PCBs
- Message switches
- /BROADCAST commands
- MSC output

**Definition:** The LTERMs used in the previous situations are known as *dead-letter queues*. IMS provides commands for the MTO to monitor these queues and dispose of them.

**Related Reading:** For more information on dead-letter queues, see “Asynchronous Output to an Invalid Destination” on page 164.

The two types of asynchronous output destinations are valid and invalid.

**Definitions:** A *valid destination* is one intended to receive output. An *invalid destination* is one that is not intended to receive output (for example, a misspelled destination).

#### Asynchronous Output to a Valid Destination

You can send asynchronous output (such as inserts, broadcasts, and message switches) after you have specified the following:

- ETO=Y
- A descriptor that is valid for creating a user structure
- A valid destination LTERM name

If control blocks exist for a previously created user structure and LTERM, the control blocks are reused.

#### Asynchronous Output to an Invalid Destination

IMS refers to data that cannot be delivered as “dead letter”. Data cannot be delivered in each of the following situations:

- No autologon destination is available for queued output.
- The user ID to which the data is associated is not a valid user ID.
- The user signon is always rejected by an installation Signon exit routine.
- An invalid destination is specified on the input or output message (for example, resulting from a typing error).

You can specify a DLQT value on the EXEC parameter at initialization in order to automatically notify the MTO when LTERM queues exceed this value and have not dequeued data or removed the status. You can use each of the following commands against dead-letter queues:

- Use the /DISPLAY USER DEADQ or /DISPLAY STATUS USER command to identify users whose LTERM queues are older than the dead-letter-queue time (DLQT), and who have not dequeued data or removed the status.
- Use the /ASSIGN command to reassign dead-letter queues to other dynamic users so they can review queued data.
- Use the /DEQUEUE command to purge data on the dead-letter queues.

In a shared-queues environment, you can use the /DISPLAY QCNT MSGAGE command to determine the messages that are considered to be dead-letter queues.

**Related Reading:** For more information on these commands, see *IMS/ESA Operator's Reference*.

## Delivering Output Messages to Non-Originating Terminals

IMS sends your output to the terminal on which you are signed on. Using ETO, you can receive your output messages at a different terminal than the one from which you entered the input. However, the input and output messages are formatted subject to the MFS specifications defined for both the terminal and messages, as follows:

- MFS formatting is mandatory for 3270R (non-SNA) and SLU-2 terminals, except when you use MFS bypass. MFS formatting is optional for all other VTAM terminals. Use MFS on a message-by-message basis, based on MID and MOD control block availability. MFS paging support is not available for SLU-1 or NTO terminals.
- You must define MID and MOD control blocks by using device statements that generate appropriate DIF and DOF control blocks. This allows you to map the message to and from a specific terminal type at the time that the message is sent to or received from the terminal. Default mapping occurs when the appropriate MFS blocks are not available. IMS issues error message DFS057 when the format blocks cannot be found.

If you plan to deliver output messages to non-originating terminals, you must develop or expand MFS formats, procedures, and restrictions. This allows users to move freely between terminals.

## Inadvertent Output Data Streams

Using ETO, dynamic terminal users can move freely between terminals; limited only by installation constraints. It is possible to erroneously send terminal-specific data to the wrong terminal type by mistyping the IMS /ASSIGN command. When this happens, the data, when delivered, has errors or is not recognizable. Be sure to create MFS definitions for all terminal types for which users can log on.

---

## Signing Off

**Definition:** *Signing off* of an ETO terminal ends the identification of a user to IMS, and (in most cases) disconnects the user structure from the terminal structure and deletes the user structure.

When a user signs off from an ETO VTAM terminal, IMS calls the Signoff exit routine (DFSSGFX0).

## Signing Off

**Recommendation:** If you have provided a Signon exit routine (DFSSGNX0) that maintains system information, provide a Signoff exit routine (DFSSGFX0) also, to complement that processing.

---

## Logging Off

When a user logs off from a VTAM terminal when ETO is used, IMS calls the Logoff exit routine (DFSLGFX0).

**Recommendation:** If you have provided a Logon exit routine (DFSLGNX0) that maintains system information, provide a Logoff exit routine (DFSLGFX0) also, to complement that processing. Ensure that the Logoff exit routine handles all non-MSA and non-LU 6.2 terminals with which IMS communicates.

Using the Logoff exit routine, you might want to maintain a count of the terminals that are logged on.

---

## Improving Performance by Deleting ETO Control Blocks

ETO support allows you to dynamically delete control blocks that were previously created. Dynamically deleting these control blocks reduces storage usage and can improve performance.

If special terminal processing options (TRACE and STOPPED) are reset, IMS deletes session control blocks if one of the following occurs:

- No user is signed on, and a checkpoint occurs.
- The session is terminated normally or abnormally by either an MTO command or by an autologoff timeout.

If the node is a 3600/Finance or SLU P terminal, message resynchronization is necessary, and the control blocks are not deleted. For ISC terminals, the control blocks are deleted after a cold-session termination. The control blocks remain, however, after an ISC warm-session termination.

If the user resets special processing options, such as those that exist after issuing a /SET, /TEST MFS, /EXCLUSIVE, or /TEST command, and issues the /SIGN OFF command (or is automatically signed off), IMS deletes the user control blocks, provided:

- No messages are queued to any LTERMs related to this user
- The user is not in conversation, Fast Path, or response mode

If the preceding conditions exist, dynamically created user control blocks are deleted. If the preceding conditions do not exist, the user control blocks can continue to exist until the conditions exist or until an IMS cold start occurs. After special processing options are reset and if all other criteria for deletion exist, the control blocks are deleted at the next checkpoint.

---

## Using the IDC0 Trace Facility

You can use the IDC0 Trace facility to diagnose logon and logoff errors. This facility provides information that the IMS message DFS3672 cannot provide. To use the facility, enter: /TRACE SET ON TABLE IDC0. The facility traces the following events:

## Using the IDC0 Trace Facility

- Errors that occur in the IMS VTAM exit routines (within module DFSCNXA0). These errors are also identified in a DFS3672 message, regardless of whether the IDC0 Trace is in effect.
- Errors that occur when attempting to log onto a nonexistent VTAM node (such as entering a /OPNDST command for a nonexistent terminal). IMS issues associated messages DFS2061 or DFS2062.
- Synchronization anomalies that occur between the time that an IMS VTAM exit routine completes processing and the time that the request is accepted by normal IMS processing. The result is a X'6701' log record identified with a VTPO string.

**Related Reading:** For more information on the IDC0 Trace facility and the format of the X'6701' log record identified with a VTPO string, see *IMS/ESA Diagnosis Guide and Reference*.

---

## ETO and LU 6.1 (ISC) Terminals

For LU 6.1 (ISC) terminals, IMS supports parallel sessions to the same node name. In this case, a separate structure is built for each session. However, each session and its associated structure operate independently, as a separate terminal.

ISC supports an SNA-defined user-data area within the BIND. When establishing a session for ISC, each half-session partner is identified through an appropriate session qualifier that is included as user data with the logon. These two qualifiers cannot be specified using the DFSLGNX0 exit routine. One belongs to each half session. IMS uses the session qualifier of the current half session as a user structure. This user structure is used to allocate an associated set of LTERM queues and to automatically provide a RACF signon, if required. The other half-session qualifier is saved with the IMS user structure. Both qualifiers are used for session-restart requests and SNA STSN message resynchronization after session failures.

**Restriction:** The SNA-predefined format for user data does not support some of the parameters and options of the non-ISC end-user format:

- The RACF password and group name are not supported. IMS supports RACF signon for ISC with PASSCHK=NO during user structure allocation as part of session initiation.
- The LOGOND and USERD are not supported. IMS uses defaults, unless specified on the Logon and Signon exit routines.
- When autologon is generated for ISC terminals, the AUTLDESC keyword in the user descriptor is ignored, and the LOGOND keyword in the user data is omitted.

**Related Reading:** For information on BIND user data formats, see *IMS/ESA Customization Guide* and “Using Default CINIT or BIND User Data Formats” on page 150.

---

## ETO and STSN Terminals

This section provides information on administering ETO for STSN terminals.

### SNA STSN Terminal Considerations

ETO terminals that use the SNA STSN function (Finance, SLU-P, and ISC) must provide a user queue name during logon, because this queue is used to resolve the

## ETO and STSN Terminals

sequence number exchange that is part of the IMS connection process for this type of terminal. The ways to provide the user name include:

- CINIT data sent by the terminal
- CINIT data provided using a separate terminal host product, such as VTAM unsolicited system services (USS)
- A user Logon exit routine (DFSLGNX0), except for ISC
- An IMS command (/OPNDST can specify user name)
- Autologon parameters supplied by user descriptors or by the Output Creation exit routine

The Logon exit routine is the last opportunity to provide the user name. If no user name is provided for ETO STSN terminals, the logon is rejected with an error message. This differs slightly from other terminal types, which allow user signon after the logon has occurred. A signon is required before being able to use the terminal for IMS activity (transactions or commands), so the difference is only in requiring the signon data earlier for STSN terminals. The /SIGN command is supported for STSN terminals.

## ETO and 3600/Finance and SLU P

You can sign on to static system-defined 3600/Finance and SLU P terminals in one of two ways: using the /SIGN command or using logon user data. You can change the signon identification by using another IMS /SIGN command at any time. LTERMs are assigned to the terminal during system definition and are not affected by the signon process or by the SNA STSN message recovery process. The signon simply provides user access and input authorization.

IMS supports 3600/Finance and SLU-P terminals as dynamic terminals. They can use autologon or signon data with the logon request in order to dynamically allocate user structures. Signon data must be provided at session initiation for ETO 3600/Finance and SLU-P terminals. Signon data can be supplied in the Logon exit routine (DFSLGNX0) at the cold start of a session. The LTERMs and user IDs allocated at the cold-start session are retained across sessions and IMS outages because of the VTAM STSN message resynchronization requirements. This requires that the same user signon data be used for subsequent warm start sessions to both reverify the user and to allow message resynchronization.

When dynamic XRF Finance and SLU-P terminals are defined as XRF class-2, automatic re-signon and logon occur at takeover time.

## /SIGN Support for ETO STSN Devices: ISC, Finance, and SLU P

For ETO STSN devices, user data is required at the time that the session is allocated to create the user structure. After the user structures are created and allocated to the terminal, /SIGN commands are accepted from ETO STSN terminals.

When you issue the /SIGN command from an ETO STSN terminal, IMS initiates a complete signon process to create the security profile associated with the session for the new user.

When the user signs off, the user's security profile is deleted, leaving the session without any security. RACF rejects all access to RACF-protected resources. The DFS3662 message is displayed if the failing resource is a command. The DFS2469 message is displayed if the failing resource is a transaction. When a user signs on



to an STSN device, the same user structure allocated during session allocation is used for the new user. IMS updates the security profile of the session and stores the signon information.

The user ID of a user that is signing on to an ETO Finance, SLU-P, or ISC device with a /SIGN command is a different name from that of the user structure name. Such users do not require suffixing by the Signon exit routine (DFSSGNX0) in order to support multiple signons, because the user structure for these devices was already created during the session initiation.

**Restriction:** Because the user structure allocated to a Finance, SLU-P, or ISC device cannot be changed, most of the options available to the Signon exit routine are not supported, and the work areas are not passed to the Signon exit routine.

**Related Reading:**

- For more information on using execution parameters, see *IMS/ESA Administration Guide: System*.
- For information on using exit routines, see *IMS/ESA Customization Guide*.

The DFS3650 message is displayed after a signon, and the DFS058 message is displayed after signoff. The user field in the DFS3650 message reflects the user structure's name rather than the RACF user ID. This is the same as for any other ETO terminal.

**Restriction:** Issuing the /SIGN command from STSN output-only devices is not allowed.

---

## ETO with Conversation Mode and Response Mode

ETO user structures are distinct from static terminal structures, because terminal status is maintained for each user rather than for each terminal. An IMS conversation at a static terminal is distinct from an IMS conversation at a dynamic terminal, even for the same user name. The conversation at a static terminal can be held, and must be released (resumed) at the static terminal. It cannot be moved to a dynamic terminal.

With ETO, an IMS conversation can be resumed at the same terminal or another dynamic terminal. Because the conversation is an attribute of the user structure, it normally follows the user to a different terminal when the user signs on to IMS. This can be a cause of confusion, especially if the user is not aware of which terminals are static and which are dynamic. Also, if an exit routine selects different user structures for signons to different physical terminals, confusion can occur.

Resume full-function and Fast-Path response mode at the same static terminal. For ETO, resume full-function and Fast-Path response mode from the same or another dynamic terminal. This is similar to the situation with conversations described in the previous paragraph, and the same considerations apply.

## Conversation and ETO

For ETO, conversations are associated with the user—not the terminal that initiates the conversation. Conversations are also associated with the terminal, but only while the user is signed on. By signing off, the user can continue a conversation on a different terminal. This flexibility requires the installation to address output formatting problems.

## ETO with Conversation Mode and Response Mode

**Related Reading:** For information on how to resolve these problems, see “Delivering Output Messages to Non-Originating Terminals” on page 165.

Users in conversations that are not in response mode can sign off. Regardless of response mode, users in conversation can be automatically signed off through autosignoff or by using an MTO command. Any form of signoff leaves the terminal available for the next user. The conversation mode status follows the user to the next terminal.

## Response Mode

Response mode is defined on the TERMINAL macro for static terminals, on the ETO user descriptor for dynamic (ETO) users, and on the TRANSACT macro for transactions. Also, response mode is either full function or Fast Path. You can also use response mode with conversation mode, if you are not running Fast Path. Response mode is primarily associated with the user and the transaction, rather than with the dynamic terminal.

When a user is in response mode, the keyboard or input response is locked until the output reply is available. During this time, it is not possible to enter input at the terminal. Normal signoff and logoff commands are not allowed. However, these functions can occur automatically during abnormal session termination. This can happen in one of three ways:

- VTAM can detect an error and end abnormally (abend).
- The MTO can issue the IMS /CLSDST or /STOP command.
- IMS can autosignoff after the specified autosignoff interval.

Regardless of how signoff occurs, response mode is retained for the user that has been automatically signed off. The user's response mode operation is reestablished with the next terminal on which the user signs onto and remains until the response-mode output reply is available. If autosignoff occurs, the terminal is available for a subsequent signon by any remote terminal operator (RTO).

**Related Reading:** For more information on delivering messages to a different terminal than the originating terminal, see “Delivering Output Messages to Non-Originating Terminals” on page 165.

---

## Part 3. Administering Multiple Systems Coupling

<b>Chapter 8. Overview of Multiple Systems Coupling</b> . . . . .	173
Multiple Systems Coupling Concepts and Terminology . . . . .	173
Physical Links . . . . .	173
Logical Links . . . . .	175
Data Partitioning with Multiple Systems Coupling . . . . .	176
Remote and Local Systems . . . . .	177
Flow of Data within Multiple Systems . . . . .	179
Message Routing . . . . .	180
Routing Path . . . . .	181
Logical Destinations . . . . .	181
Input, Destination, and Intermediate Systems . . . . .	182
System Identifiers (SYSIDs). . . . .	184
Using the Destination Name and SYSIDs To Route Messages . . . . .	185
Remote LTERMs. . . . .	187
Routing Exit Routines . . . . .	188
Terminal Routing Exit Routine (DFSCMTR0) . . . . .	188
The Link-Receive Routing Exit Routine (DFSCMLR0) . . . . .	189
MSC Program Routing Exit Routine (DFSCMPR0) . . . . .	189
Input Message Routing Exit Routine (DFSNPRT0) . . . . .	190
MSC Directed Routing. . . . .	190
Remote Destination Verification . . . . .	191
Logical Link Path . . . . .	191
<b>Chapter 9. Administering Multiple Systems Coupling</b> . . . . .	193
Design Considerations for Multiple Systems . . . . .	193
Minimizing Resource Consumption . . . . .	193
Balancing Resource Demand . . . . .	194
Planning for Conversational Processing . . . . .	194
Routing Exit Routines with Conversations . . . . .	196
Remote Destination Verification for Conversations . . . . .	196
Saving Truncated Data in the SPA . . . . .	196
Conversation Termination . . . . .	197
Abnormal Conversation Termination. . . . .	197
System Definition for Multiple Systems Coupling . . . . .	197
Local System Definitions . . . . .	198
Defining Partner Systems . . . . .	198
Defining the Physical Link . . . . .	199
Defining the Logical Link . . . . .	199
Defining a Logical Path . . . . .	200
Setting Link Priorities for Remote Transactions. . . . .	200
Specifying Routing Exit Routines . . . . .	201
Types of Routing Exit Routines . . . . .	201
Managing Error Messages on an MSC Link . . . . .	202
How Network Definition Is Affected by Multiple Systems . . . . .	203
Verifying Transaction Definitions Across Systems . . . . .	203
Using the Multiple Systems Verification Utility . . . . .	203
Verifying the System Definition Status Online . . . . .	204
Security Considerations for Multiple Systems . . . . .	205
Establishing Operating Procedures for Multiple Systems . . . . .	206
Initializing Multisystem Communication. . . . .	206
Terminating Multisystem Communication . . . . .	206
IMS Commands That Affect MSC Operation. . . . .	207
Logical Link Assignments . . . . .	207

Restarting a Logical Link . . . . .	208
Commands That Help Control Resources. . . . .	208
Using the /DISPLAY Command . . . . .	209
Logical Link Path Control. . . . .	210
Recovery Considerations. . . . .	210
Message Recovery . . . . .	211
Stopped Transactions . . . . .	211
Application Program Abnormal Termination . . . . .	211
Dequeuing Messages, Responses, and Transactions . . . . .	211
Monitoring and Tuning Multiple Systems . . . . .	212
Coordinating Performance Information . . . . .	212
Interpreting IMS Monitor MSC Reports. . . . .	213
Determining Cross-System Queuing . . . . .	213
Assessing the Effect of Link Loading . . . . .	214
Assessing Link Queuing Times . . . . .	215
Extracting Multiple-System Transaction Statistics . . . . .	215
Controlling the Log Merge . . . . .	215
Interpreting the Transaction Analysis Report. . . . .	216

---

## Chapter 8. Overview of Multiple Systems Coupling

**Definition:** *Multiple Systems Coupling (MSC)* makes it possible for transactions to be entered in one IMS system and processed in another IMS system. The responses can be returned to the terminals that entered the transactions or to other terminals. IMS uses MSC to route and control message traffic between connected systems.

### **In this Chapter:**

- “Multiple Systems Coupling Concepts and Terminology”
- “Data Partitioning with Multiple Systems Coupling” on page 176
- “Remote and Local Systems” on page 177

---

### Multiple Systems Coupling Concepts and Terminology

**Definitions:** A *link* is a connection between two systems. All links must be defined during the IMS system definitions for each IMS system. Two types of links exist: physical and logical.

- A *physical link* is the actual hardware connection.
- A *logical link* is the mechanism through which a physical link is related to the transactions and terminals that make use of that physical link.

You can assign a logical link to a physical link during system definition, or the master terminal operator can assign it dynamically during system execution.

### Physical Links

Multiple Systems Coupling supports three types of physical links:

- VTAM
- Channel-to-channel (CTC) adapter
- Memory-to-memory (MTM) under one MVS

Only the VTAM and CTC adapter represent actual hardware links. The MTM link is a software link between IMS subsystems that are running in the same MVS system, and it is used primarily for backup and testing purposes. Physical link buffer sizes must be equal between the two systems. Figure 23 illustrates the three types of physical links.

## Multiple Systems Coupling Concepts and Terminology

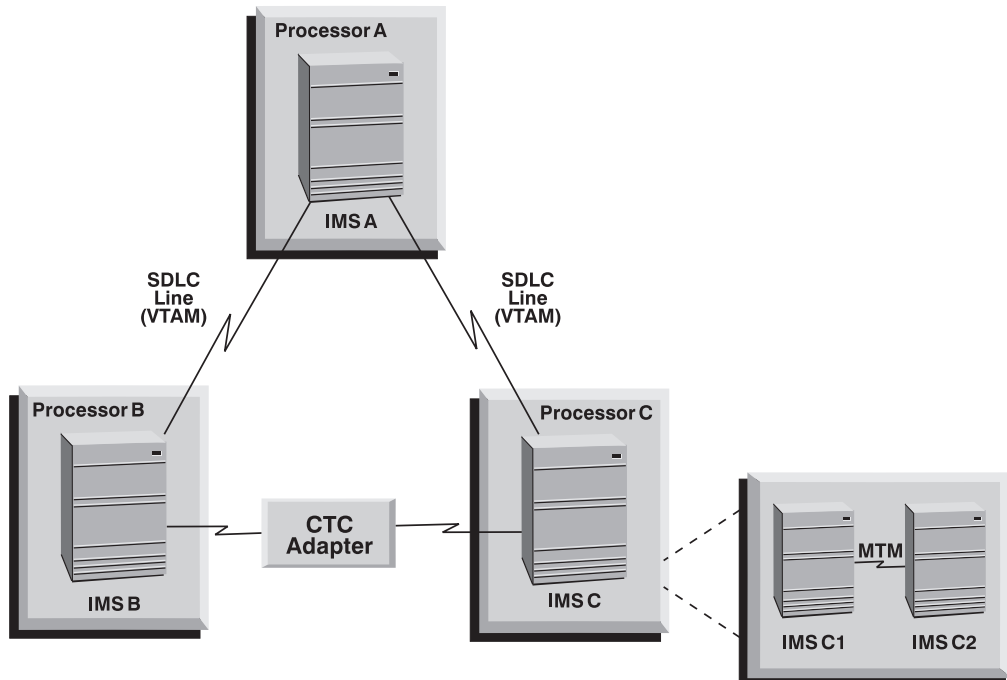


Figure 23. MSC Physical Link Types

Several IMS systems that coexist in a single processor can be linked to each other by one or more of each physical link type, as shown in Figure 24.

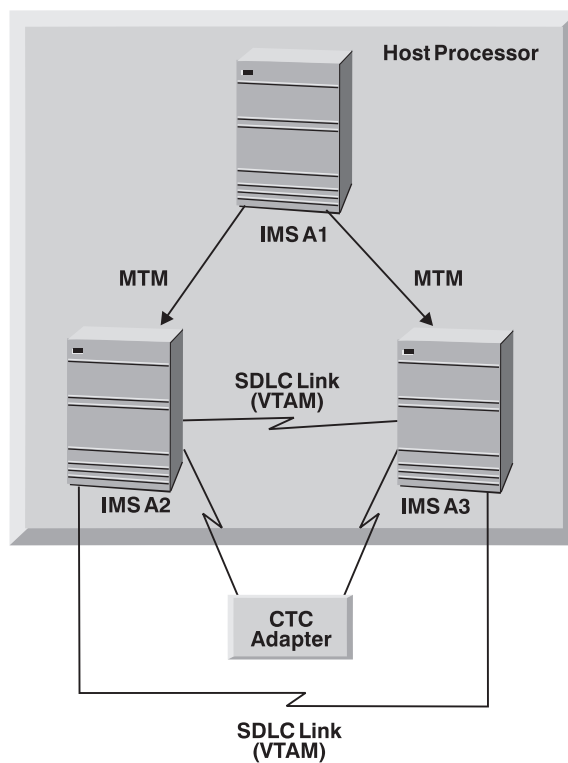


Figure 24. Multiple Physical Links in One Processor

## Multiple Systems Coupling Concepts and Terminology

You can also use multiple links between processors as shown in Figure 25 .

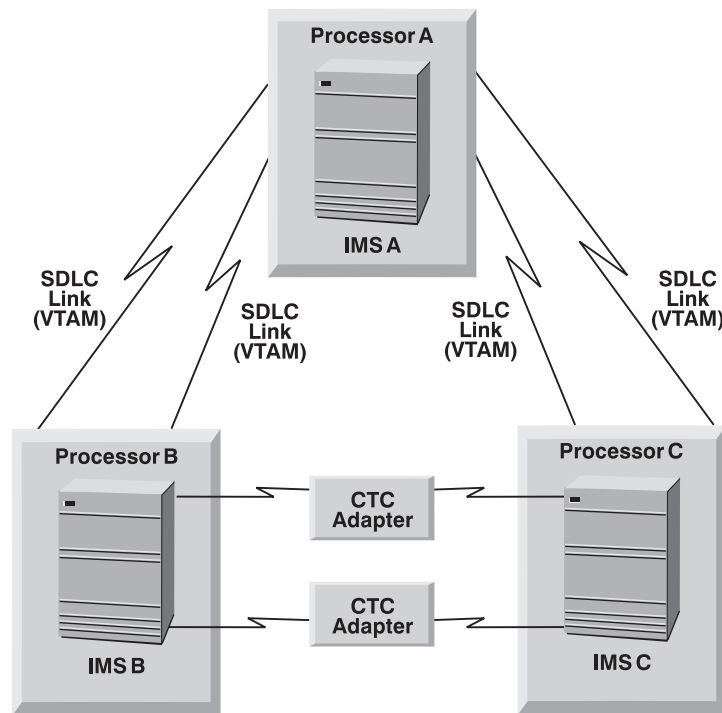


Figure 25. Multiple Physical Links with Several Processors

## Logical Links

A logical link relates a physical link to the transactions and terminals that can use that physical link. Each system in a multisystem configuration has one or more defined logical links.

**Definition:** Two IMS systems that are defined to communicate with each other, each through a specific logical-link, are called *partner systems*.

To establish connection between two IMS systems, each partner must have a logical-link definition. The two logical link definitions must specify the same partner identifications and be assigned to the same physical link. The IMS system definition process assigns a number to each defined logical link. Logical link numbers are assigned sequentially, beginning with 1, in the order in which the links are defined. A logical-link can be reassigned to a different physical link, but the two systems must always communicate through a logical link partnership.

With the SDLC link that uses VTAM, multiple sessions can use the link. During IMS system definition, you can specify how many sessions are to share the SDLC link. Each session that is activated becomes a logical link between partner systems.

The IMS system definition process does not require that a physical link be specified for each logical link. You can assign a physical link to the logical link online by using the IMS /MSASSIGN command. No communication between partners can occur until the assignment is made.

## Data Partitioning with Multiple Systems Coupling

MSC provides the ability to connect geographically dispersed IMS systems in such a way as to allow programs and operators of one system access to programs and operators of the connected systems. Communication can occur between two or more (up to 2036) IMS systems running on any supported combination of MVS systems.

MSC also provides a way to 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).
- Applications execute in one system with the complete database that they reference attached to that system (vertical partitioning); the transactions can originate in any system.

Figure 26 illustrates horizontal partitioning.

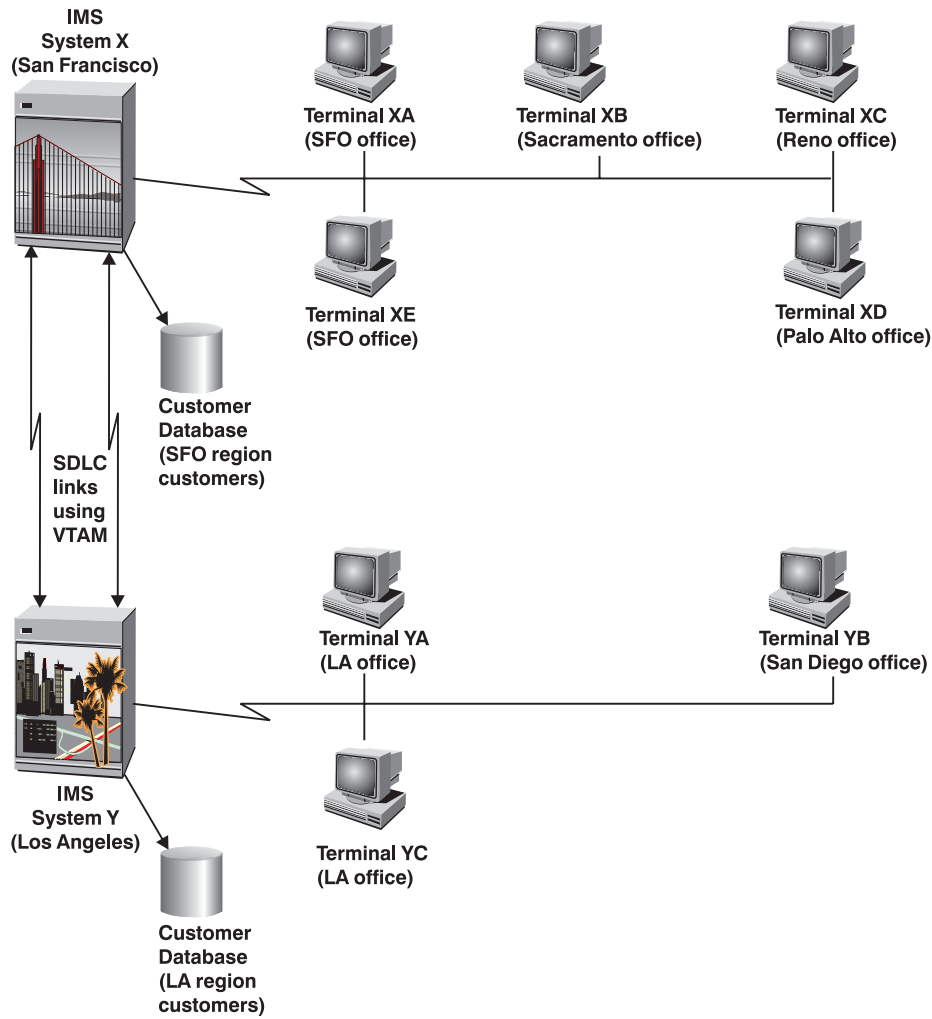


Figure 26. Horizontal Partitioning - Common Database Function with Physically Split Data



## Data Partitioning with Multiple Systems Coupling

Figure 27 illustrates vertical partitioning.

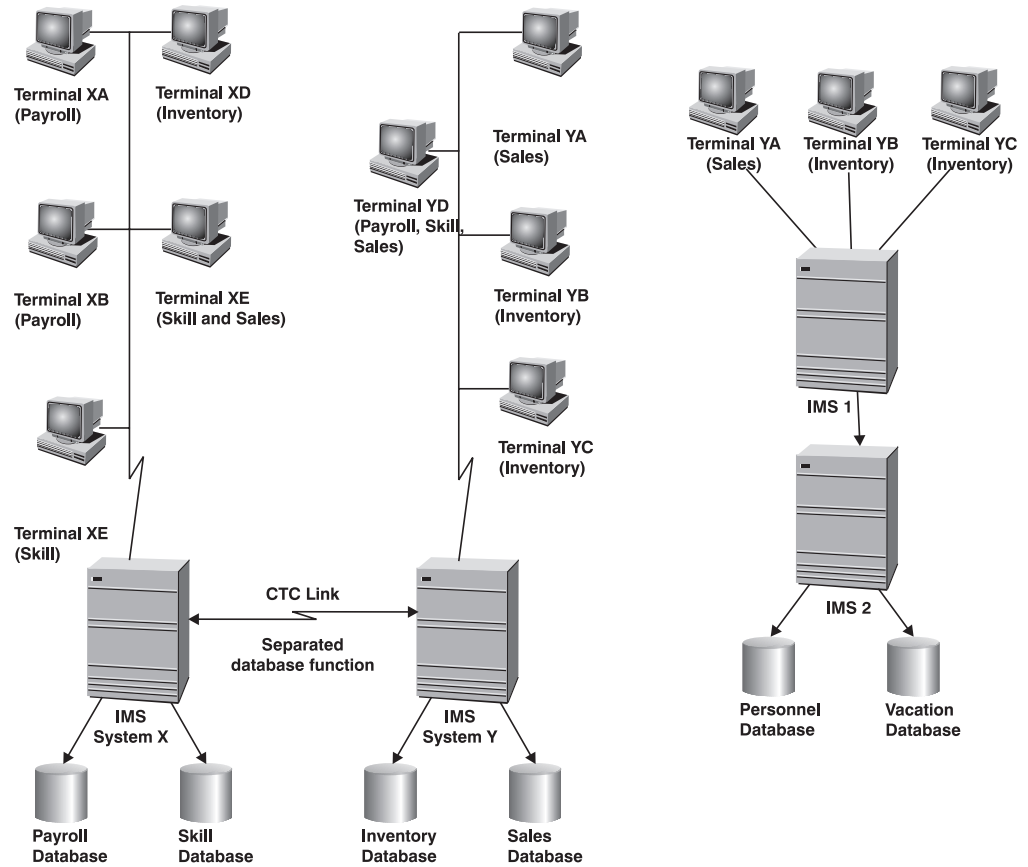


Figure 27. Vertical Partitioning - Different Applications in Each IMS System

## Remote and Local Systems

In this general description of MSC, an example is used of a multisystem environment consisting of four systems: IMS A, IMS B, IMS C, and IMS D. Figure 28 on page 178 shows the multisystem environment.

## Remote and Local Systems

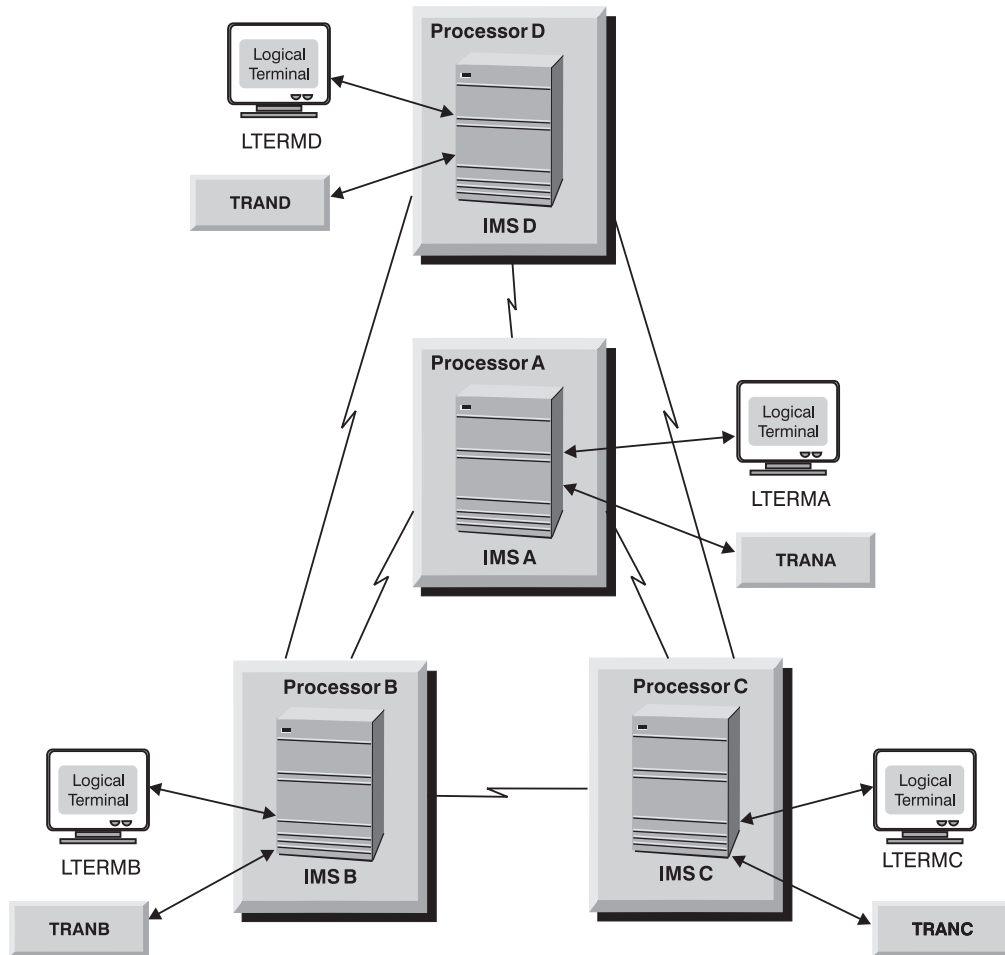


Figure 28. A Multisystem Environment — Four Systems

### Definitions:

- In a multisystem environment, a *local system* refers to a specific system within the multiple configuration. All other systems are considered *remote systems* in regard to the specific local system.
- A *local transaction* is a transaction that is processed in the same system in which it is entered.
- A *remote transaction* is a transaction that is entered into a system from a terminal or a link that is not processed in that system.

In Figure 28, IMS B is receiving a message. IMS B is the local system, and IMS A, IMS C, and IMS D are remote systems. Transaction TRANB, when entered from LTERMB in IMS B and processed in IMS B, is considered a local transaction. When TRANB is entered from IMS A, IMS C, or IMS D and sent across an MSC link to be processed in IMS B, it is considered a remote transaction. Similarly, transactions TRANA, TRANC, and TRAND are remote transactions when they are entered from LTERMs in one system and processed in another system; they are local transactions when they are processed on the same system from which they are entered.

## 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 29 on page 180, 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.<sup>28</sup>
- MSC support is responsible for removing the message off of the 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 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 off of 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)**.

---

28. The message is queued to the MSNAME associated with the specified remote destination.

## Remote and Local Systems

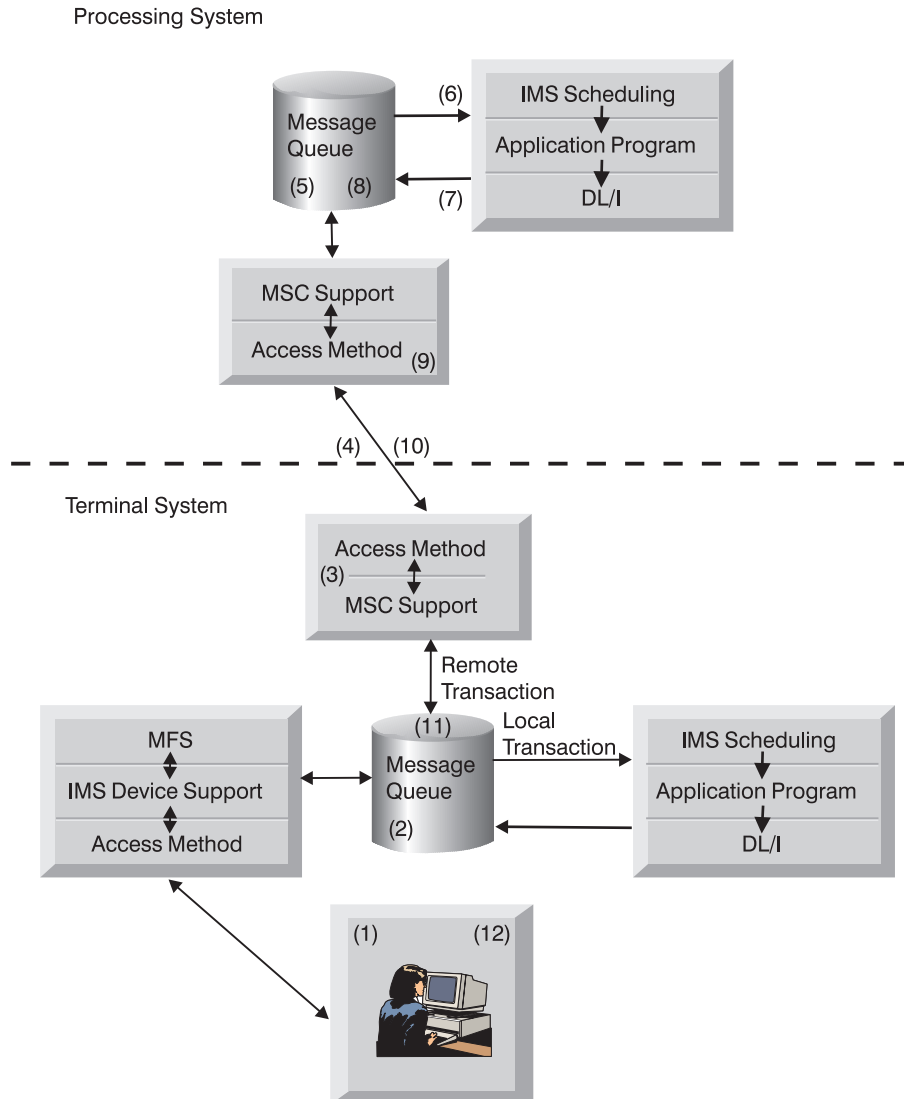


Figure 29. Multiple IMS Systems Transaction Flow

## Message Routing

The message routing function of MSC supports several types of message routing:

- Routing transaction messages from a terminal in one system to an application program in another system. The transaction can be defined as recoverable, irrecoverable, response mode, or conversational.
 

**Restriction:** Fast Path transactions across an MSC link are not supported. However, Fast Path transactions within the same local MSC system are supported.
- Routing message switches from an LTERM in one system to an LTERM in another system is supported, as are message switches between LTERMs in the same system. This includes messages sent with a /BROADCAST command.
- Routing response messages from an application program to the input terminal that sends the transaction, or messages from an application program to an alternate terminal is supported. Routing messages to alternate remote terminals (the transaction and the LTERM are in different systems) requires that the

alternate LTERM be defined as a remote LTERM. If directed routing or an MSC Program Routing exit routine is used, the alternate LTERM does not need to be defined as remote.

- Program-to-program switches between transactions in different IMS systems or within the same IMS system are supported.

## Routing Path

IMS passes messages from the originating system to the processing system on a routing path. One or more systems can be included in a routing path. In Figure 30, IMS B has a path to IMS A and back (path BA). Similarly, back-and-forth paths exist between IMS B and IMS D (path BD), IMS B and IMS C (Path BC), and IMS A and IMS D (path AD). A path can go through an IMS system in order to get to another IMS system, such as path CAD between IMS C and IMS D. More than one path can exist between the same two systems. IMS C and IMS D have two direct paths, CD1 and CD2, in addition to the indirect path CAD. A total of 255 paths are allowed in an MSC configuration.

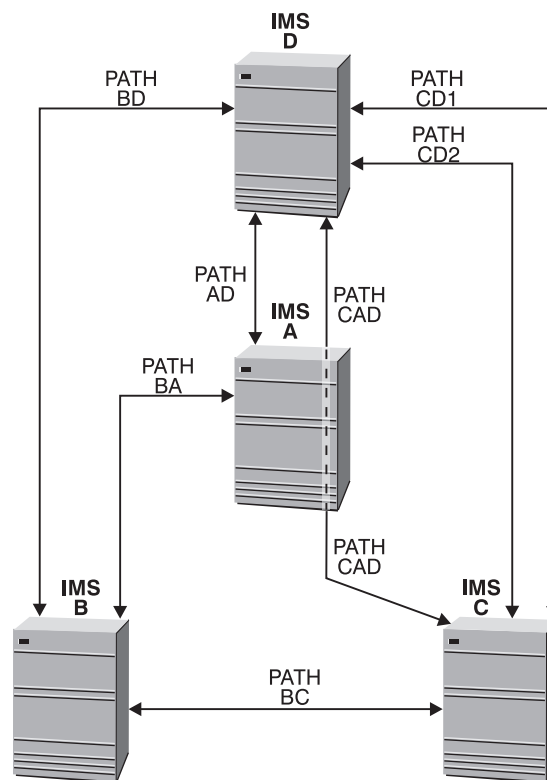


Figure 30. Routing Path

## Logical Destinations

Message routing for a multiple-system configuration uses logical destinations, as does a single-system environment. A destination is either a logical terminal or a transaction code. A local destination resides in the local system, and a remote destination resides in a remote system. Within each local system, all local and remote destinations must be defined with unique names. In Figure 31 on page 182, all the local and remote destinations are uniquely defined within each local system. Destinations that are defined remotely are not also defined locally within the same system. Similarly, destinations that are defined locally are not also defined remotely.

## Remote and Local Systems

The same destination names can be used for local destinations in different IMS systems that are connected by MSC. The destination names cannot conflict with the global intent of the destination within the MSC configuration. For example, in Figure 31, TRANAB is a local transaction in IMS B and IMS A. It is a remote transaction in IMS C. IMS C is referencing the local TRANAB in IMS A only and not the one in IMS B. IMS C cannot remotely reference TRANAB in IMS B. The destination system that is referenced in a remote destination is determined by the system identification (SYSID) value.

**Related Reading:** For more information on SYSIDs, see “System Identifiers (SYSIDs)” on page 184.

Directed routing and the MSC Program Routing exit routines enable you to send messages from an application program to a remote destination that is not explicitly defined in the transaction processing system. The validation of the destination name in the transaction processing system is delayed until the message arrives in the local system. Delaying this validation provides more flexibility in the resource naming requirements.

**Related Reading:** For more information on directed routing and MSC exit routines, see *IMS/ESA Customization Guide*.

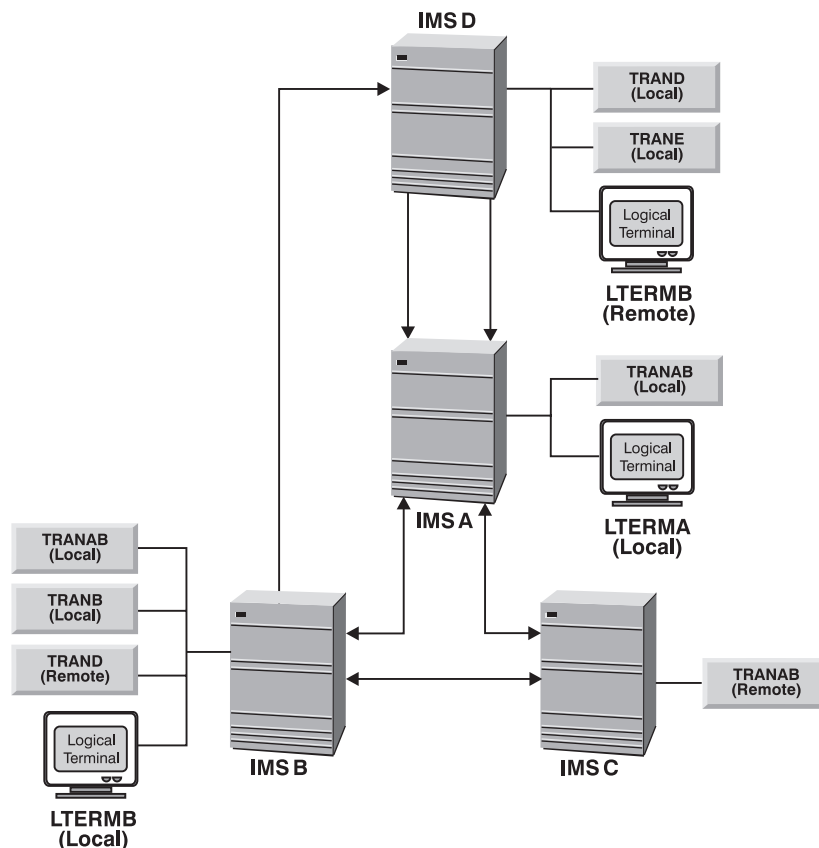


Figure 31. Logical Destinations

## Input, Destination, and Intermediate Systems

Depending on the message, an MSC system can be either an input system, a destination system, or an intermediate system. For example, in Figure 32 on page 183

page 183 for the transaction message originating from LTERMB in IMS B and routed over path BD to TRAND in IMS D, IMS B is the input system and IMS D is the destination system. This message is called a primary message. When the message is processed by the application program in IMS D and a reply is returned to the input terminal in IMS B, this reply message is called a response. For this response message, the destination and input system is IMS B. The destination of the message is LTERMB in IMS B. The input that caused this response message is also from LTERMB in IMS B, which is the primary transaction.

If a message is sent through one IMS system and routed directly to another IMS system for processing, the routing system is called the intermediate system. For example, if TRAND is sent from IMS B through IMS A to IMS D (path BAD), IMS A is the intermediate system and IMS D is the destination system. Similarly, IMS A and IMS D are intermediate systems for TRANC when TRANC is sent from IMS B through IMS D through IMS A to IMS C (path BDAC).

Remote destination names must be defined as remote in the input system and as local in the destination system. Intermediate systems, however, do not need to define the input, destination name, or the path back (a local SYSID) for the routed message. Only the path to the destination system needs to be defined. The path back might not even be through the intermediate system. Input and destination name checking is not performed on a message when it is routed through an intermediate system.

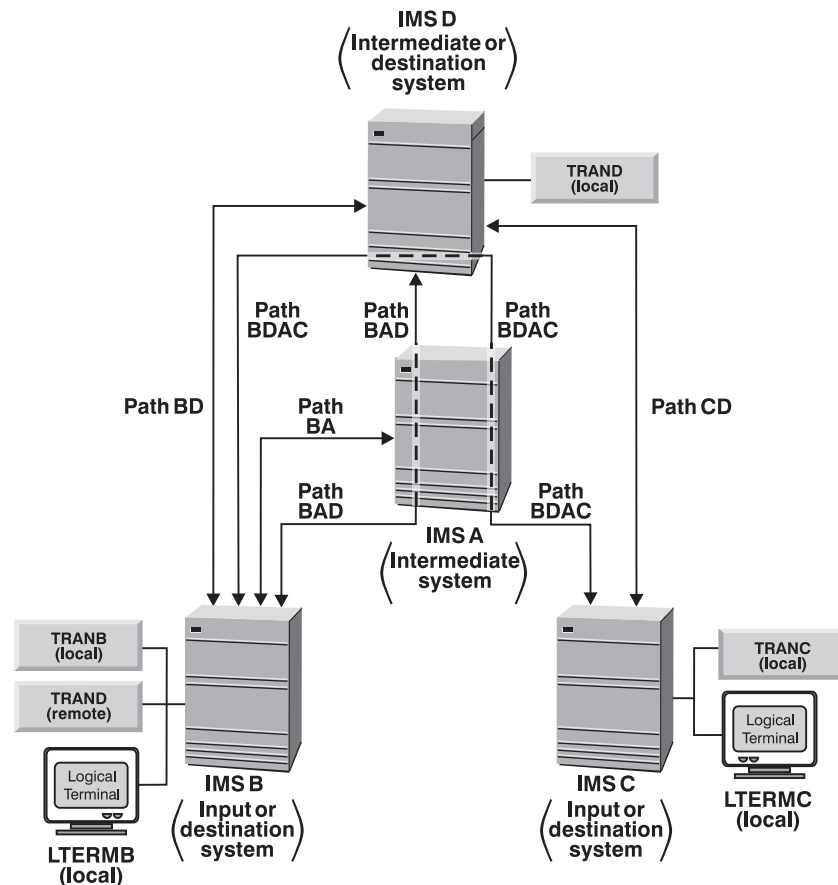


Figure 32. Input, Destination, and Intermediate Systems

## Remote and Local Systems

### System Identifiers (SYSIDs)

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 to the owning system and a remote SYSID to any other IMS system that has a path to this system. In Figure 33 on page 185, IMS B has local SYSIDs 1, 2 and 3. IMS A has local SYSID 4. IMS C has local SYSID 5. IMS D has local SYSIDs 6 and 7. Local SYSIDs are implicitly defined when MSNAME macros specify the paths to other IMS systems. That is, the local SYSID value of the SYSID parameter on the MSNAME macro determines which local SYSIDs are defined to an MSC system.

**Related Reading:** For more information on the MSNAME macro, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

In Figure 33 on page 185, IMS B has local SYSIDs 1, 2, and 3 because of the three MSNAME macros with local SYSIDs of 1, 2, and 3. Paths to other systems are determined by the remote SYSID value on the MSNAME macro SYSID parameter. IMS B has three paths to remote SYSIDs 6 (IMS D), 7 (IMS D), and 5 (IMS C). IMS D and IMS B cannot route messages to IMS A, because they do not have paths to SYSID 4 (IMS A). IMS A does, however, have a path to SYSID 7 (IMS D) and SYSID 2 (IMS B). IMS A cannot send messages that originate in IMS A to SYSID 7 or SYSID 2, because the source SYSID (SYSID 4) is not recognized by IMS B or IMS D. In this configuration, IMS A can only function as an intermediate system for IMS B and IMS D. Another path (MSNAME) must be defined for IMS B to communicate with IMS A and IMS D.

**Related Reading:** For more information on paths and how a transaction uses a path, see “Logical Link Path” on page 191.



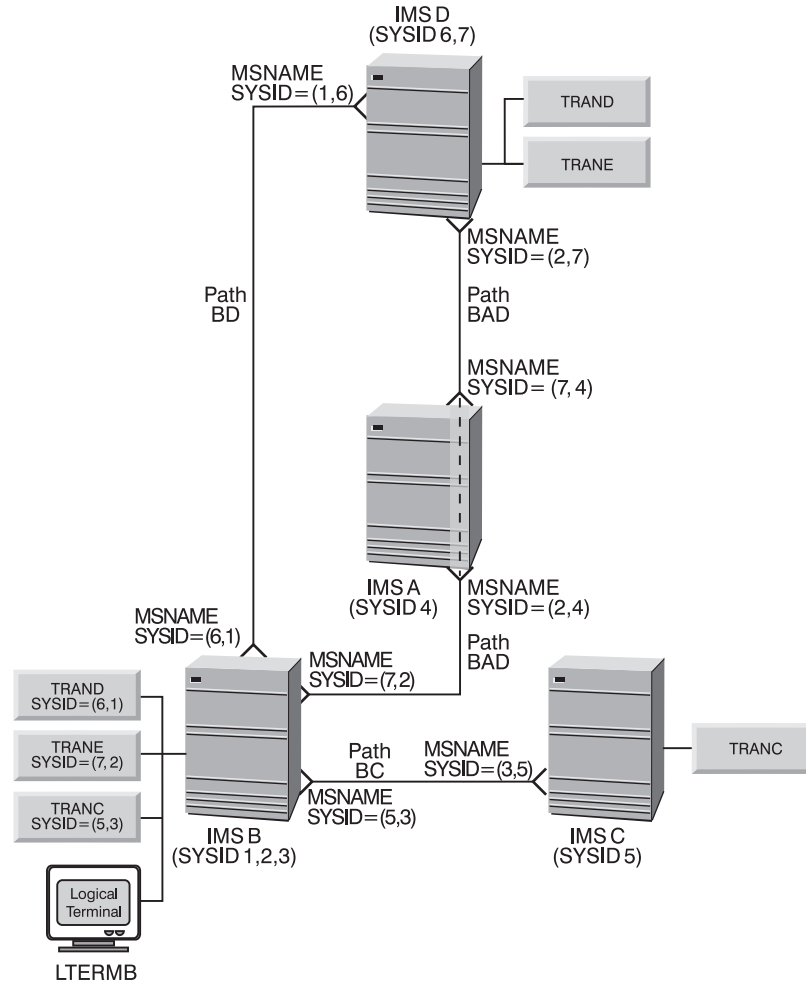


Figure 33. System Identifiers (SYSIDs)

## Using the Destination Name and SYSIDs To Route Messages

Messages in an MSC configuration contain information that makes it possible to route the message between IMS systems. When a remote transaction is entered from an LTERM, a transaction message is built and queued on the message queue. This message contains information that is needed to route the message to its remote destination:

- Remote destination name (transaction code)
- Local or source LTERM name of the LTERM that entered the transaction
- Remote SYSID value
- Local or source SYSID value

If the transaction is processed by the application program and secondary messages are sent to other transactions (program-to-program switches), these messages have the destination name and SYSID of the switched-to transaction. The source (origin) name and SYSID remain the same. That is, the source SYSID and name never change. This facilitates sending the response message back to the input LTERM, regardless of where the transaction is processed.

Any system that locally processes a message that is received from a remote IMS system must have a local SYSID defined to it that is the same as the remote

## Remote and Local Systems

SYSID. It must have a path back to the source (origin) of the message. In Figure 34, if TRANA is entered from LTERMB in IMS B, it is sent across the path MSNAME=(4,2) to IMS A. The destination name and SYSID of the message are TRANA and 4. The source name and SYSID are LTERMB and 2. IMS A accepts the message and processes it. It has SYSID=4 defined as local and has a path back to IMS B with the destination SYSID=4. If TRANA in IMS A issues a program-to-program switch to TRAND in IMS D, the destination name and SYSID are TRAND and 5. The source name and SYSID remain LTERMB and 2. IMS D accepts and processes the transaction. The transaction has SYSID=5 defined locally and has a path back to IMS B with destination SYSID=2.

If TRAND in IMS D sends a response back to input LTERMB, the response message has a destination name and SYSID of LTERMB and 2, and the source name and SYSID are also LTERMB and 2.

The input LTERM, LTERMB, is never defined as remote in IMS A or IMS D, yet the response message is returned to LTERMB in IMS B. IMS carries the originating LTERM name and SYSID in the primary message and all secondary messages. IMS knows where to route the response message when the application program responds to the input LTERM.

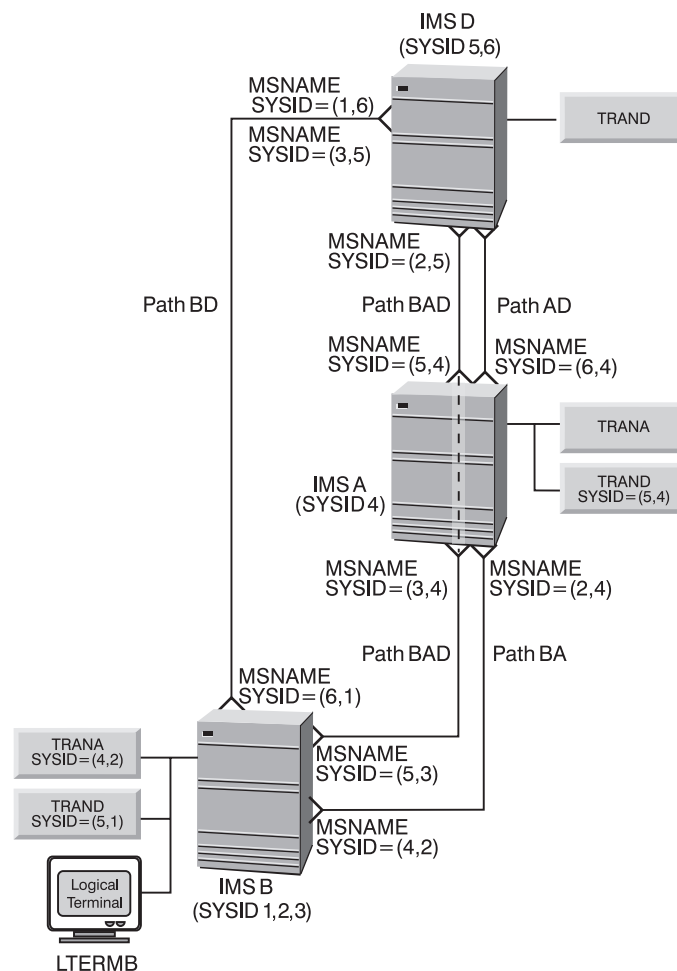


Figure 34. Message Routing

## Remote LTERMs

**Definition:** A *remote LTERM* is a logical terminal that does not reside on the local IMS system.

**Recommendation:** Define the input LTERM as a remote LTERM only to send message switches, remote broadcasts, or messages from an application program to an alternate remote terminal (alternate PCB).

You can define remote LTERMs by using ETO MSC descriptors. The ETO MSC descriptor relates remote LTERMs to statically defined MSC links.

**Related Reading:** For more information on ETO MSC descriptors, see “MSC Descriptor Format” on page 148.

To help return the message to the input (source or origin) LTERM, IMS carries the source LTERM name and SYSID in the remote message. In the remote IMS system, define NAME macros that have the name of the local terminal in the local system. Associate the NAME macro with the MSNAME that defines the destination SYSID of the local IMS system.

For example, in Figure 35 on page 188, IMS A has local LTERMA, IMS B has local LTERMB, and IMS D has local LTERMD. IMS B can send message switches and remote broadcasts from LTERMB to LTERMD. LTERMD is defined remotely in IMS B. LTERMD is not actually defined with SYSID=(5,2), but it assumes those SYSIDs when the message is issued. LTERMD is associated with MSNAME BAD, which is defined with SYSID=(5,2) in IMS B. This association is established by placing the NAME macro for LTERMD after the following MSNAME macro:

```
BAD MSNAME SYSID=(5,2)
```

This naming differs from remote transaction definitions that are explicitly defined with remote and local SYSIDs.

Also, LTERMA in IMS A can send message switches or remote broadcasts to LTERMD in IMS D. Similarly, LTERMD in IMS D can send message switches or remote broadcasts to LTERMA in IMS A. Both IMS A and IMS D have remote LTERM specifications for the other system’s local LTERM.

TRAND in IMS D can send alternate messages to LTERMB in IMS B or to LTERMA in IMS A. MSNAMEs are defined with names for the purpose of:

- Displaying the queue counts for the named logical link path
- Stopping the sending of all messages from a terminal except those continuing a conversation
- Starting the logical link path
- Purging the logical link path in an MSC configuration for which input is stopped
- Allowing application programs to use directed routing
- Reassigning using the /MSASSIGN command

In Figure 35 on page 188, the name of each MSNAME is the two-character name preceding MSNAME.

## Remote and Local Systems

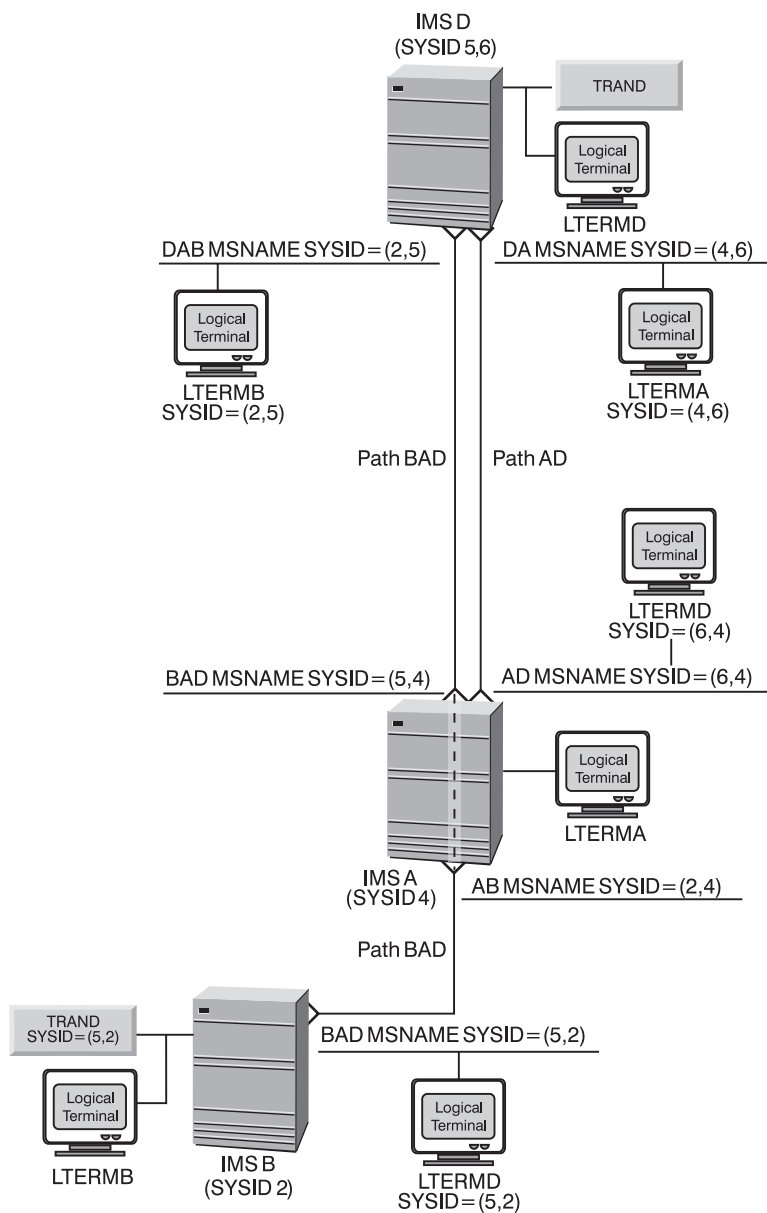


Figure 35. Remote LTERMs

## Routing Exit Routines

Message routing is automatic, according to the defined scheme, unless a routing exit routine is provided by the user. The routing exit routine is called before the message destination is verified. The types of routing exit routines are:

- Terminal routing
- Link routing
- Program routing
- Input message routing

### Terminal Routing Exit Routine (DFSCMTR0)

The Terminal Routing exit routine executes in the input system and is called when a message is received from a terminal. This routine can inspect the specified

destination (that is, the transaction code) and, if desired, change it to any local or remote destination. If this routine does not change the destination, the originally specified destination is used for routing.

This routine can also examine the first segment of the message to determine the transaction code entered at the terminal. The routine can alter the code independently of the destination. If the transaction code is changed in the message segment, this code remains in the segment when it is passed to the application program.

When used with the Link-Receive Routing exit routine, terminal routing can reduce the number of remote transaction definitions for each system. In the remote system, one transaction of each type, for example, conversational or inquiry, can be defined. This transaction is used for the destination, while the original transaction code that is entered at the terminal remains in the first message segment.

IMS does not call a Terminal Routing exit routine for commands, for any message received from a terminal in preset destination mode, or from a terminal that is continuing a conversation.

In a configuration using horizontal partitioning, the Terminal Routing exit routine can be used to evaluate all input messages and route them to the appropriate processing system based on information in the first segment of the input message. If transactions and links are appropriately defined, this routine can also be used to set a common destination for a set of input messages. On arriving at the destination system, the messages are processed according to their individual transaction codes.

**Related Reading:** For more information on how to code the Terminal Routing exit routine (DFSCMTR0), see *IMS/ESA Customization Guide*.

### **The Link-Receive Routing Exit Routine (DFSCMLR0)**

The Link-Receive Routing exit routine can determine the destination (transaction code) or LTERM name, if directed routing is used, when the transaction arrives at the processing IMS system. This exit routine is called when the transaction reaches the remote processing system. It can inspect the transaction code or the LTERM name that is used as the destination and, if desired, change it to another transaction code with the same attributes. This routine can examine the first segment of the message to determine what the new transaction code should be. If the routine does not change the transaction code, the destination remains the location that is specified by the original transaction code.

**Related Reading:** For more information on how to code the Link-Receive Routing exit routine (DFSCMLR0), see *IMS/ESA Customization Guide*.

### **MSC Program Routing Exit Routine (DFSCMPR0)**

The MSC Program Routing exit routine allows you to control the routing of a message that is entered from a local or remote terminal. This routine can route a message in the following three ways:

- You can route a message to a specific application program in the originating system.
- You can route the message to the LTERM in the system associated with the SYSID or the MSNAME.
- You can route the message to the application program in the system associated with the SYSID or the MSNAME.

## Remote and Local Systems

You can use this exit routine to avoid defining unique names for remote LTERMs and transactions. With this exit routine, you can have the same name for master terminals throughout your MSC network. You can use this exit routine to execute the DL/I CHNG call for the application program, and to change the destination from local to remote.

This exit routine is given control when a DL/I CHNG call is issued by an application program that meets all of the following requirements:

- The application executes in an IMS system in which MSC is defined.
- It executes as an MPP or a message-driven BMP.
- The transaction is not conversational.
- The PCB is not an alternate response PCB.
- The application has executed a GU call to an I/O PCB.

**Related Reading:** For more information on how to code the MSC Program Routing exit routine (DFSCMPRO), see *IMS/ESA Customization Guide*.

### Input Message Routing Exit Routine (DFSNPRT0)

You can code an Input Message Routing exit routine (DFSNPRT0) to change the destination of an LU 6.2 input message immediately after it is received in the originating system, but before it is processed. This exit routine receives control when the first segment of an input message is received from an LU 6.2 application program.

This exit routine provides the same function as the Terminal Routing exit routine, DFSCMTR0. You can change the destination of an input message to any local or remote destination. The exit routine receives control when the first segment of an input message is received from an LU 6.2 application program. If both DFSCMTR0 and DFSNPRT0 exist in the system, only DFSNPRT0 is called.

One of the ways you can use this exit routine is to help you balance your workload. You can use this exit routine, in combination with the existing Link-Receive Routing exit routine, in order to reduce the number of remote transactions that must be defined. For example, if 40 different transactions execute in a remote IMS system and all of them have the same attributes (such as nonconversational or non-inquiry), define one IMS remote transaction to represent all 40. In the Input Message Routing exit routine, when a message for any one of the 40 remote transactions is detected, change the destination name to a name that represents all 40 transactions in the remote system.

Both exit routines must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library that is concatenated in front of the IMS.RESLIB.

**Related Reading:** For more information on the Input Message Routing exit routine (DFSNPRT0), see *IMS/ESA Customization Guide*.

## MSC Directed Routing

**Definition:** *MSC directed routing* is a function of MSC that allows an application program to specify the system name (MSNAME) and destination within that system for a message to an LTERM or an application program. The receiving application program can determine the MSNAME of the system that originally scheduled it. With directed routing, the specified remote destination (a transaction or an LTERM) in another system does not need to be declared explicitly in the IMS system

definition for the sending system. These logical (local) names for terminals enable different IMS systems in the MSC network to use the same logical names for terminals and transaction codes. Names must still be unique within a given system. The Multiple Systems Verification utility (DFSUMSV0) cannot detect errors associated with MSC directed routing.

### **Restrictions:**

- MSC directed routing does not support a program-to-program switch between conversational transactions.
- MSC directed routing does not support a program-to-program switch from a nonconversational transaction to a conversational transaction. For example, a conversational transaction in System A cannot use directed routing to perform a program-to-program switch to invoke a conversational transaction in System B.
- Response mode cannot be propagated on a DL/I ISRT call in a directed routing transaction.

## Remote Destination Verification

To maintain system integrity and prevent errors, an IMS system in a multisystem configuration verifies all specified destinations, unless MSC directed routing is used. When MSC directed routing is used, IMS only ensures that a program-to-program switch is not being performed from a nonconversational transaction to a conversational transaction. Remote destination verification occurs when a message is received from a terminal or on receipt of an application program reply if a remote destination is specified for the message. Destination verification occurs as follows:

### **Destination      Verified For:**

**LTERM**            Destination type: The original destination must have been a logical terminal.

**Transaction**      Destination type: The original destination must have been a transaction.

Transaction attributes: The following attributes must be consistent in the transaction definitions in the input and destination systems:

    Single segment or multiple segment

    Recoverable or irrecoverable

    Conversational or nonconversational

The SPA size for a conversation from a remote system cannot be changed except by the remote system on an insert.

When an invalid destination is recognized, IMS cancels the message, sends an error message to the input terminal and to the master terminal of the local system, and logs an invalid request. If the message is conversational, the Conversation Abnormal Termination (name) exit routine is called in the input system, and the conversation is terminated.

## Logical Link Path

**Definition:** A *logical link path* is a path between any two IMS systems. One or more logical link paths must be defined for each logical link. A logical link path is defined in the MSNAME macro and specifies the following:

- A system identification for the system in which messages using this path are to be processed.
- A system identification for the system being defined.

## Remote and Local Systems

Each system in a multisystem configuration has one or more unique system identifiers (SYSIDs) ranging from 1 to 2036. SYSID assignments are implicit, based on the logical link paths defined by MSNAME macros.

**Example:** Consider the two MSNAME definitions:

```
MSNAME  SYSID=(2,1)
MSNAME  SYSID=(3,1)
```

The first definition says that messages using this logical link path are processed in the remote system whose local SYSID is 2. The second definition says that messages using this logical link path are processed in the remote system whose local SYSID is 3. By using these definitions, the IMS system definition process assigns SYSID 1 to the system being defined and recognizes two remote systems with SYSIDs of 2 and 3. If a third path were defined with SYSID=(5,4), IMS would also assign SYSID 4 to the local system.

Transactions are assigned to logical link paths in the APPLCTN macro definition.

**Example:** Consider the following application definitions, each with one transaction code defined:

```
APPLCTN  PSB=A
          TRANSACT  CODE=A
APPLCTN  PSB=B,SYSID=(2,1)
          TRANSACT  CODE=B
APPLCTN  PSB=C,SYSID=(3,1)
          TRANSACT  CODE=C
```

The SYSID keyword identifies the logical link path to be used for the transactions associated with the application. Transaction A is considered to be a local transaction, because the absence of the SYSID keyword indicates transaction A is only processed by the system being defined. Transactions B and C are remote transactions. Relating the application definitions to MSNAME definitions, IMS would return responses from transactions B and C to the system defined as SYSID 1, unless the application program specified an alternative destination for the response.

If any logical terminals in a remote system are referred to by messages originating in the local system, the logical link definition must also include NAME macros to identify those remote logical terminals, unless directed routing is used.

Figure 36 summarizes the relationships of one physical link to one logical link to multiple logical link paths.

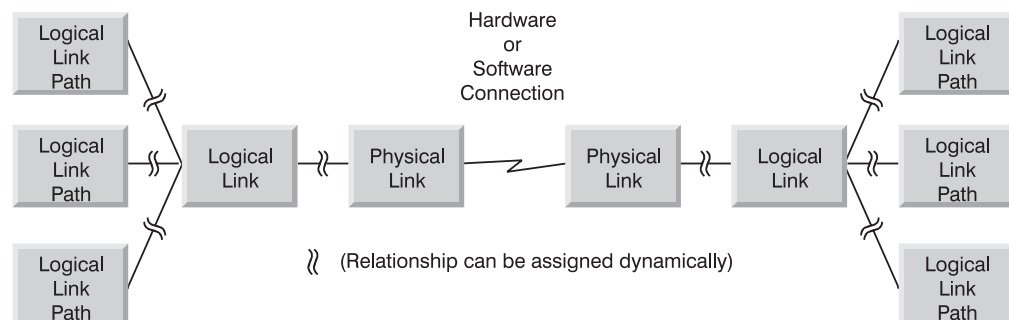


Figure 36. Relationship of Physical Link to Logical Link to Logical Link Path



---

## Chapter 9. Administering Multiple Systems Coupling

This chapter describes the system administration activities required when you connect two or more IMS online systems in a network using MSC.

LU 6.2 application programs can process transactions on remote IMS systems. ETO also supports remote LTERMs.

**Related Reading:** For more information on remote LTERMs, see “Remote LTERMs” on page 187.

### **In this Chapter:**

- “Design Considerations for Multiple Systems”
- “Planning for Conversational Processing” on page 194
- “System Definition for Multiple Systems Coupling” on page 197
- “Verifying Transaction Definitions Across Systems” on page 203
- “Security Considerations for Multiple Systems” on page 205
- “Establishing Operating Procedures for Multiple Systems” on page 206
- “IMS Commands That Affect MSC Operation” on page 207
- “Recovery Considerations” on page 210
- “Monitoring and Tuning Multiple Systems” on page 212

---

## Design Considerations for Multiple Systems

The major design goals for MSC are:

- Minimizing resource consumption by defining suitable connections between the systems
- Balancing resource demand by distributing functions among systems to obtain acceptable performance
- Programming design considerations for multisystem conversational transactions

The design and tuning recommendations that apply to a single IMS system are applicable to each IMS system in an MSC environment. Resource demand and consumption are taken into account when defining systems that are part of an MSC configuration.

An IMS transaction that is processed in a local system uses the same hardware and software resources that it would in a non-MSC environment. Transactions that are processed in a remote system require additional resources. In addition to resources used to transmit the transaction over physical links to the remote processor and to receive the response from the remote processor, resources are needed for message queuing and logging. Performance considerations are directly related to minimizing the resources consumed by remote processing and balancing the resource demand between several processors in an MSC configuration.

## Minimizing Resource Consumption

To minimize resource consumption, do each of the following:

- Design the environment so that as many transactions as possible are processed locally.

## Design Considerations for Multiple Systems

- Provide physical links that go directly from local to remote systems; no intermediate systems should be involved in the transaction routing process. Transactions that must be routed through intermediate systems require additional processor activity, message queue activity, and logging activity.
- Design the message queue buffer pool in each processor to eliminate unnecessary message queue I/O activity.
- Use SDLC links (MSC VTAM), rather than CTC links; SDLL links provide a common communication technique and are independent of physical links.
- Define the physical link buffer sizes large enough to hold the message prefix plus all the segments of most messages. Also, the queue buffer size should be large enough to hold the largest segment that is inserted to the message queue by a transaction as response to the input terminal.

## Balancing Resource Demand

In an MSC environment with two or more processors, try to distribute the workload in a way that avoids excessive use of any one processor. You can distribute the workload by distributing IMS applications and their associated transactions and terminals between the available processors. Depending on the complexity of the application and the capability of the processor, you can avoid overloading any single processor.

If the current design of the databases is such that the databases and their associated applications cannot be distributed across the available processors, you can:

- Duplicate inquiry-only databases; this allows more than one system to reference the whole of the database. (This is called vertical partitioning.)
- Split the databases into several component databases. (This is called horizontal partitioning.) The component databases must be completely independent for distribution among the available processors. For example, it might be possible to divide a database by key range intervals. The new databases and their associated applications can then be distributed among the existing IMS systems, and you can use the Terminal Routing exit routine to route incoming transactions to the correct IMS system. Another possibility is to divide the database by geographic area. Each IMS system could process the transactions that refer to the databases for its own geographic area and route transactions that refer to a remote geographic area.

In addition to balancing the workload across processors, you might also need to balance the workload on physical links. This occurs when a physical link between two systems is of the SDLC type and multiple physical links have been installed. You can balance the workload on physical links by:

- Specifying, during IMS system definition, proper logical link paths and logical links for each remote application.
- Using a user-written Terminal Routing exit routine to distribute the transaction load on each of the alternative physical links.

---

## Planning for Conversational Processing

Conversational processing is available to terminals that are attached to any system in a multisystem configuration to the same extent as if they were in a single-system configuration.

## Planning for Conversational Processing

The following differences apply to conversational processing in a multisystem configuration:

- All transactions used in a conversation must be defined as conversational in each system of the multisystem environment.
- The input system controls the conversational resources for the duration of the conversation. When the input system receives a conversational transaction, it inserts the scratch pad area (SPA) as the first message segment, and routes the message to the destination application program.
- Any system in the multisystem configuration can process any step in the conversation.
- Program-to-program switches can be routed from system to system.
- SPAs that are used in multisystem conversations must follow these rules:
  - For the SPA ISRT, conversational program-to-program switches can occur to a transaction with a SPA that is equal, larger, or smaller in size.
  - For transactions sent to IMS Version 5.1 systems or older (using MSC links), the following requirements apply:
    - If the SPA ISRT occurs on a remote system, the SPAs must be the same size.
    - If the SPA ISRT is being returned to the input terminal, the SPA must be less than or equal to the SPA size of the transaction issuing the ISRT.
  - The minimum size of a SPA is 16 bytes (X'10'), and the maximum size is 32767 bytes (X'7FFF').

Generally, terminal operators and application programs are unaware of whether a conversation is multisystem.

### **Exceptions:**

- Suppose a conversational program inserts a message to a response alternate PCB in a remote system. By implication, this destination is in the input system and is verified by the input system. Destination verification in this case involves confirming that the specified logical terminal is still assigned to the input terminal. If the logical terminal has been reassigned, the input system activates the Conversation Abnormal Termination exit routine, and terminates the conversation. The status code that is returned to the application program is blank, indicating success. This status code is blank even when the actual result is unsuccessful due to the condition described above.
- Suppose an application program executing in a system other than the input system uses the SPA to specify a transaction code, thereby passing conversation control to another program. If the specified transaction code is invalid, the input system activates the Conversation Abnormal Termination exit routine, and terminates the conversation. No status code is returned to the application program.

MSC support for APPC/IMS includes all IMS transaction types:

- Conversational
- Nonconversational
- Response mode
- Nonresponse mode

**Restriction:** MSC does not support Fast Path.

## Planning for Conversational Processing

### Routing Exit Routines with Conversations

A Terminal Routing exit routine can be used for the input message that starts a conversation. It is not applicable at any other conversational step, because the application program, not the input terminal, provides the destination for continuation of the conversation.

A Program Routing exit routine cannot be used during conversations.

### Remote Destination Verification for Conversations

Destinations for program-to-program switches are verified in the system in which the program requesting the switch executes, except where MSC directed routing is used. When MSC directed routing is used, IMS ensures only that a program-to-program switch is not being performed from a nonconversational transaction to a conversational transaction. If valid, the system sends the SPA and the message directly to the destination transaction. If invalid, the system returns a status code to the application program.

Destination verification for a message to the input terminal is performed by the input system. The specified logical terminal must still be assigned to the input terminal. The input system also verifies, except when MSC directed routing is used, the next transaction that is specified in the SPA. If the destination is invalid, the input system invokes the Conversation Abnormal Termination exit routine and terminates the conversation. No status code is returned to the application program.

### Saving Truncated Data in the SPA

The SPA=STRUNC option on the TRANSACT macro applies to conversations that use program switches to other transactions that have different sized SPAs:

- If you use the SPA=STRUNC option, IMS preserves all of the data in the SPA, even when a program switch is made to a transaction that is defined with a smaller SPA. The transaction with the smaller SPA does not see the truncated data. However, when this transaction switches to a transaction with a larger SPA, the truncated data is used. IMS tracks the longest data that is inserted to the SPA to determine the truncated data length.

**Example:** If you have the three transactions:

```
TRANA SPA=100  
TRANB SPA=50  
TRANC SPA=150
```

If the application program for TRANA switches to TRANB, the last 50 bytes of the SPA for TRANA are not sent to TRANB. If the application program for TRANB subsequently switches to TRANC, the SPA that is received by TRANC contains the following 150 bytes:

- The first 50 bytes from the SPA that was inserted by TRANB
  - The second 50 bytes from the second 50 bytes that was inserted by TRANA
  - The third 50 bytes are binary zeros.
- If you do not use the SPA=STRUNC option, the truncated data is lost. In the previous example, the second 50 bytes that is received by TRANC would be binary zeros.

**Restriction:** Truncated SPA data from a previous transaction is lost if it is sent using MSC to an IMS 5.1 (or earlier) system.

## Conversation Termination

A conversation can be terminated by either the application program or the terminal operator. An application program terminates a conversation by inserting a message to the input terminal with a SPA that contains either blanks as the transaction code or the transaction code of a nonconversational transaction.

A conversational transaction can also be terminated by the `/EXIT` command.

An IMS shutdown does not terminate conversations. The conversation is continued after IMS restarts, unless an IMS cold start occurs or the conversation is terminated by an operator command such as `/EXIT`. If the input system is shut down and subsequently cold starts, all the conversations that it controls are lost. The input system cancels any conversational messages it receives for input terminals that were previously involved in active or held conversations. IMS retains conversations over a restart. An IMS cold start is an unusual procedure, and can cause many problems beyond losing conversation state.

If a remote system is shut down when a conversational step is processing or in its queue, and it is subsequently cold started, all references to the conversation are lost. A conversation that is lost in this way must be specifically canceled in the input system by the `/EXIT` command.

## Abnormal Conversation Termination

A conversation is abnormally terminated if any one of the following events occur:

- The conversational application program abnormally terminates.
- An invalid destination is recognized in the input system or in the remote system (for a conversation response, a program-to-program switch, or in the SPA).
- A conversational message is inserted into a terminal whose conversation was terminated.
- Destination verification fails for a conversational message.
- No output was generated in the application program.

The conversation's SPA (along with an indication of the cause of termination) is passed to the Conversation Abnormal Termination exit routine in the input system.

---

## System Definition for Multiple Systems Coupling

You need to define, using system definition macros, transaction codes for each IMS system that has any part in transaction entry or processing. An individual system can play several roles. It can be the input system, it can be an intermediate system responsible for routing transactions, or it can be a destination system in which the transaction is processed.

To generate an IMS online system that includes MSC facilities, your IMS system definition must include three macros: `MSPLINK`, `MSLINK`, and `MSNAME`.

**Related Reading:** For more information on the `MSPLINK`, `MSLINK`, and `MSNAME` macros, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Using MFS is the same in a multisystem environment as in a single-system environment. If a message is created in one system for a terminal that is attached to another system, the required message and format descriptions must be available

## System Definition for Multiple Systems Coupling

to the system to which the terminal is attached, and definitions with the same name must be defined identically in each system.

## Local System Definitions

When the multisystem environment is defined, the items that are defined for each local system include:

- All transactions entered or processed by that system
- All logical terminals that are attached to that system and all logical terminals in remote systems that are referenced by transactions processed in that system, or by terminal operators, unless a Program Routing exit routine is used
- The physical and logical connections between that system and the remote systems that share in the processing of the specified transactions

The system definition macros that you need to prepare in each system are summarized in Table 8. The first macro you use is IMSCTRL. Using the MSVID keyword, you assign a number in the range from 1 to 255 as a unique identifier of that system. This causes a control block, DFSMSxxx, to be built for use with the Multiple Systems Verification utility. (The 3-digit suffix, xxx, matches the MSVID parameter; one unique control block exists for each system in the multisystem network.) This offline utility helps verify that system definitions for all partner systems are consistent. The IMSMSV procedure is generated in IMS.PROCLIB in order to execute this utility.

Table 8. Summary of Macros for Multiple Systems

Resource	Identification	Macro	Number Coded
System	System ID	IMSCTRL	1
Programs	PSB name	APPLCTN	1 per PSB
Local transactions	Transaction code	TRANSACT	1 or more per APPLCTN
Remote transactions	Transaction code	TRANSACT	1 or more per APPLCTN in a remote system
Exit routines	Exit routine type	COMM	1
Physical connections	Physical link name	MSPLINK	1 per connection
Logical connections	Logical link name	MSLINK	1 per MSPLINK <sup>1</sup>
Routing names	Logical path name	MSNAME	1 or more per MSLINK
Local terminals	LTERM name	NAME	1 per local terminal
Remote terminals	LTERM name	NAME <sup>2</sup>	1 per remote terminal

### Notes:

1. Multiple-session SDLC links that are VTAM have additional MSLINK macros.
2. Define with an ETO MSC descriptor if the corresponding local terminal is an ETO terminal.

## Defining Partner Systems

A relationship exists between the three system definition macros (MSPLINK, MSLINK, and MSNAME) that define the names of connections between systems and the logical names that are used in commands.

### Defining the Physical Link

You can define three types of physical links to connect IMS systems. Your choice depends on the hardware that is required for each of the links. The choices are:

- Channel-to-channel (CTC)
- Memory-to-memory (MTM)
- VTAM

You use the MSPLINK macro to declare which type of physical connection you are going to use. You must declare all physical connections that can potentially be used, even if several might be for backup purposes or might not be intended for continual use. You must declare the physical link for partner systems in both system definitions. Using the TYPE keyword, declare the kind of physical link to use.

Assign a name to the MSPLINK macro; for example, LINK1. This name is used for logical link definition to match a connection between systems with the physical device or transmission technique that is to be used.

If the type is CTC, add a DD name and an address parameter value. Also specify a buffer size with the BUFSIZE keyword. This size must be equal for each end of a physical link. Use at least the size of the largest message segment that is to be transmitted across this physical link. The minimum size is 160 bytes. If the type is SDLC using VTAM, a minimum of 208 bytes is required for BUFSIZE. These buffer sizes are allocated from the communications input/output pool. Use the COMM keyword on the BUFPOOLS macro in order to allocate additional space for buffers for SDLC links. If part of a conversation is to be passed from one system to another, use the maximum scratch pad area (SPA) size, if that SPA size exceeds the message segment size.

LU 6.2 transactions that are transmitted over MSC links need larger link buffers in order to accommodate the LU 6.2 message prefix and the IMS extended prefix. The BUFSIZE parameter of the MSPLINK macro must specify a buffer that accommodates a prefix of 500 bytes. MSC non-VTAM links require a minimum buffer size of 160 bytes. MSC VTAM links require a minimum of 208 bytes and allow a maximum buffer size of 30720 bytes.

For SDLC that uses VTAM, you must supply a NAME keyword that matches the label on the VTAM APPL statement for the remote system. The NAME keyword and the VTAM APPL statement both specify the VTAM node name.

The label on the VTAM APPL statement also serves as a default value for the ACBNAME parameter on the same statement. Regardless of whether you use the default value for the ACBNAME parameter, this value must match the APPLID parameter on the IMS COMM macro.

The SESSION parameter indicates the number of parallel sessions that can be active for the physical link.

### Defining the Logical Link

The logical link definition is defined with the MSLINK macro, and has two parameters. The first parameter uses the PARTNER keyword and requires you to give a two-character identifier to represent a logical connection with the link of the same identifier in the partner system. For example, if BC is the partner identification in one logical link defined by the MSLINK macro, this same identifier is in the MSLINK macro of the partner system.

## System Definition for Multiple Systems Coupling

The second keyword, MSPLINK, enables you to match the type of physical connection to be potentially used with this logical link:

- When the physical link type is CTC or MTM, one MSLINK macro exists for each kind of physical connection.
- When the physical link type is SDLC that was VTAM parallel sessions, multiple logical links can be assigned to one physical link. For example, System\_B and System\_C can have two logical links, one using MTM and the other CTC; System\_C and System\_D can have two logical links, both of which use one VTAM physical link.

## Defining a Logical Path

When the operator connects two systems, a choice of physical connection type might be possible, therefore the logical link name is used. The MSNAME macro lets you define that logical path name. The SYSID keyword is used to declare the two systems that are joined in a pathway. You select the 1-digit identifiers from the range 1 to 2036, one for the remote system and the other for the local system. For example, (1,3) specifies that any message using this path is being sent to the remote system number 1, and that the local system number is 3.

The MSNAME macro can be followed by a set of NAME macros on which you can specify the LTERM names for terminals that are in remote systems. You do not need to declare every terminal in the remote system that is entering transactions, but only 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.

## Setting Link Priorities for Remote Transactions

A remote input transaction might have a different scheduling priority in the destination system from that defined for it during system definition of the originating system. This means there could be a long wait for a response, even though the remote transaction had a high priority when it was defined with the SYSID= parameter of the TRANSACT macro. You can assign priorities to remote transactions using the PRTY= keyword on the TRANSACT macro during system definition in the destination system.

**Example:** In Figure 37 on page 201, Terminal 1 is sending a remote transaction (ASMB1) to System B.



## System Definition for Multiple Systems Coupling

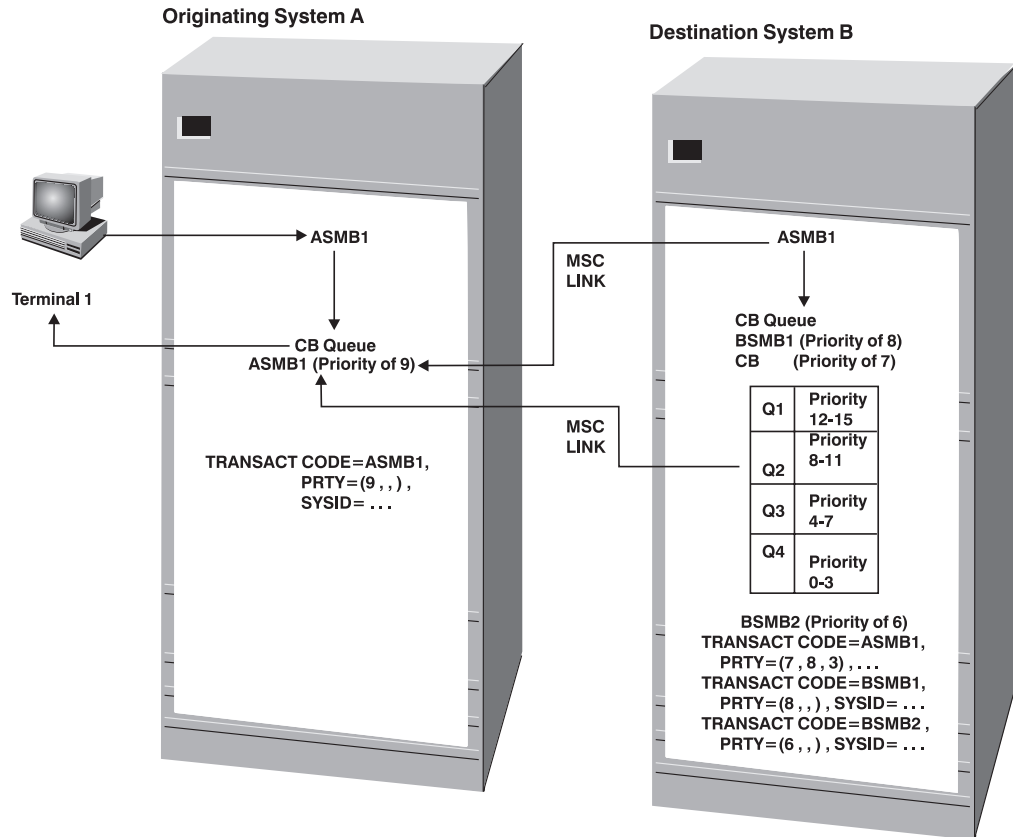


Figure 37. Link Priorities for a Remote Transaction

In IMS system A, this transaction is enqueued to a control block (CB) and has a priority of 9 (defined with the SYSID= parameter). When ASMB1 is sent to system B, it becomes a local transaction and is processed. When the message router is called to send a response, it enqueues the response on the control block that represents the MSC link. If ASMB1 is defined in system B with a priority of 7, all transactions with priorities of 12-15 (in queue 1) are processed first, and 8-11 (in queue 2) are processed next. The ASMB1 (with a priority of 7 in queue 3) is processed next. However, as shown in the figure, the limit priority can be set at 8 and the limit count can be set at 3. As a result, when the number of message queues to be processed reaches 3, the priority of ASMB1 is changed to 8, and subsequent responses are placed on queue 2. On system B, all messages for BSMB1 are sent before responses are sent. Messages for BSMB2 are sent last in this example.

## Specifying Routing Exit Routines

This section describes how to specify routing exit routines.

### Types of Routing Exit Routines

The four types of routing exit routines you can use are:

- Terminal routing using the DFSCMTR0 routine
- Link-receive routing using the DFSCMLR0 routine
- Directed routing for LTERMs using the DFSCMLR1 routine
- Program routing using the DFSCMPR0 routine

## System Definition for Multiple Systems Coupling

You need to incorporate these exit routines into IMS.USERLIB before executing stage 2 of the IMS system definition. Because an exit routine can change the original transaction code or the destination that is indicated in the system definition, be sure to clearly document the actions of these routines. One technique is to prepare a chart showing the input transaction code, the new codes, and the conditions that cause the modification.

The Terminal Routing exit routine is invoked in the input system. It can change both the transaction code and the destination. Intermediate systems do not invoke these exit routines. When a transaction arrives at its destination system, the Link-Receive exit routine can be invoked. These exit routines can change the transaction codes, possibly for the second time. Clearly document how the codes are changed, and explain the reasons for the new codes, which become the actual codes used for processing. You might need to correlate the transaction name changes as information for master terminal operators so that they can interpret a display of queue status.

You also need to be aware of how transactions can be routed for different stages in the total processing. If a primary input from System A is being processed in System C, the processing program can invoke the Program Routing exit routine to decide whether to send its output to the original input terminal or to another location. Your documentation of these exit routines should show the patterns of alternative message destination. An end user could be expecting output and be unaware that it is being sent to another component.

**Related Reading:** For more information on routing exit routines, see *IMS/ESA Customization Guide*.

### Managing Error Messages on an MSC Link

The Message Control/Error exit routine (DFSCMUX0) allows you to manage and control messages that are in error on an MSC link or on a message queue. It is activated by an MSC link start, link termination, send error, or receive errors. The Message Control/Error exit routine is called for the following situations for APPC/IMS:

- If an LU 6.2 session fails while sending an output message to an LU 6.2 program
- If a send to an LU 6.2 application program is rejected with a Deallocate with Send\_Error message
- When /DEQUEUE *luname tpname* is entered

You can customize the Message Control/Error exit routine to ask IMS to take any of the following actions:

- Take the default action: discard the message and proceed with the /DEQ command.
- Discard the message in error.
- Discard the message in error and notify the MTO or originating terminal of this error.
- Re-route the message in error to a different local or remote transaction, a local or remote LTERM, or an LU 6.2 application program.

The sample exit routine that is provided in IMS.TMSOURCE uses the default action.

**Related Reading:** For more information on the Message Control/Error exit routine, see *IMS/ESA Customization Guide*.

### How Network Definition Is Affected by Multiple Systems

Each system definition must have all terminals that are connected to it defined using the `TERMINAL` and `NAME` macros. However, if terminals in other systems are capable of sending messages that are destined for the defined system, those terminals are logically part of the network for that system.

Although MSC physical and logical links continue to be predefined through the system definition process, you can dynamically create MSC remote LTERMs during IMS initialization with Extended Terminal Facility (ETO). Although you cannot define your MSC links using ETO, ETO does allow you to associate one or more message queues with an MSC logical link.

**Related Reading:** For more information on creating MSC remote LTERMs with ETO, see “Chapter 7. Administering the Extended Terminal Option” on page 123.

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 the alternate PCB to direct a message to another terminal besides the input terminal, those destinations need to be declared as remote, unless directed routing is used. Define the LTERM names for all inputs with `NAME` macros. Position the `NAME` macros in a group after the `MSNAME` macro. You now have a set of LTERMs that collectively can occur in several system definition decks. For example, `TERMA` can be present in the input system, in the intermediate system, and in the processing system.

When planning for the network, keep in mind that message queues for an input system or an intermediate system must allow for the remote transactions being queued. When allocating space for the message queues take into account the message lengths and their expected loads. In a similar way, you must allow for the presence of these messages in I/O buffers, even when they are not going to be processed in that system.

---

### Verifying Transaction Definitions Across Systems

After you have completed system definitions for several systems that are to be part of a multiple-system network, the transaction codes, LTERM names, and system identification numbers are all identifiable in control blocks.

### Using the Multiple Systems Verification Utility

You can use the IMS Multiple Systems Verification utility offline in order to verify that the names you have used are consistent across system definitions.

**Example:** If you use `TRANX` as a code in two definitions but `TRANXX` in a third system, the Multiple Systems Verification utility sends you a message highlighting the single reference to `TRANXX`.

You can execute the Multiple Systems Verification utility by using the `IMSMSV` procedure in `IMS.PROCLIB`.

The Multiple Systems Verification utility also checks to ensure that transaction codes are not defined as local in more than one system.

The Multiple Systems Verification utility uses input control statements that specify the system identification numbers that you want to include in the checking. Its output is in the form of a multisystem path map.

## Verifying Transaction Definitions Across Systems

**Restriction:** The Multiple Systems Verification utility cannot detect errors caused by improper use of MSC directed routing.

**Related Reading:** For a description of the output and execution JCL for the Multiple Systems Verification utility, see *IMS/ESA Utilities Reference: Transaction Manager*.

## Verifying the System Definition Status Online

You can check the consistency of your system definitions or ETO MSC descriptors using the /MSVERIFY command when each of the component systems of your network is operational.

**Recommendation:** Use the /MSVERIFY command during the system test phase, rather than during production. The /MSVERIFY command generates considerable traffic.

The /MSVERIFY command validates that:

- Logical terminals exist for the remote LTERMs defined.
- Transactions are defined and have the same attributes in their remote system as in the local system.
- Logical path definitions are consistent and usable.

**Recommendation:** Define your remote LTERMs using ETO MSC descriptors. Otherwise, the Multiple Systems Verification utility recognizes remote LTERMs, but not the corresponding local LTERM in the target system.

The /MSVERIFY command verifies consistent definition between the local system and one remote system for each command. Verification begins with local SYSIDs and then proceeds to remote SYSIDs. As each segment of the processing begins, the MTO is notified of the local and remote SYSIDs to be verified. This message, DFS2234I or DFS2236I, also is time-stamped. As the specific local and remote SYSIDs are verified, the MTO receives verification-completed notifications with corresponding time stamps. If a definition or assignment error is discovered, error messages are returned.

During command processing, the local system sends all its own locally defined MSC elements (SYSIDs) to the remote IMS system specified in the /MSVERIFY command. The remote system responds by returning all destinations and their attributes that belong to the local system (as the remote system definition or assignments have them described). The command input system checks local and remote definitions for consistency.

The response list should be checked for completeness using the listing of SYSIDs that are to be verified given in the DFS2234I or DFS2236I IN PROGRESS messages and the correspondence of time stamps in responses.

The local system then sends all MSC elements to the remote system expecting them to be defined in that remote system with corresponding attributes (as the local system has them defined). The remote system performs consistency checks and routes error messages to the command input terminal.

MSC path consistency is only checked for current operational assignments of logical link paths to logical links, and logical links to physical links.

**Recommendation:** Ensure that the MSC definition and assignments are accurate, even when receiving favorable responses (that return no errors). Definition and

## Verifying Transaction Definitions Across Systems

assignment errors can prevent the return of a command response for some SYSIDs. Regard the absence of a response for a particular SYSID as significant.

Reviewing the following error responses from the /MSVERIFY command might help you to avoid these definition and assignment errors:

```
DFS2235I SYSID __ is defined as local in both systems
DFS2241I __ is defined as remote transaction in both systems
DFS2242I __ is not defined as LTERM in both systems
DFS2243I __ is not defined as transaction in both systems
DFS2245I Multisegment transaction flag for __ not consistent
DFS2246I Non-inquiry only flag for __ not consistent
DFS2247I Conversational flag for __ not consistent
DFS2248I Irrecoverable flag for __ not consistent
DFS2249I Fixed length SPA flag for __ is not consistent
DFS2250I The SPA length for __ is not the same
```

The /MSVERIFY command cannot detect errors caused by improper use of MSC directed routing.

---

## Security Considerations for Multiple Systems

For those transactions that are processed in another system, perform as much security checking as is required for the primary message.

RACF is an IBM security-orientated licensed program that is used with the MVS operating system. To be used for IMS security on static terminals, RACF requires the IMS Security Maintenance utility (SMU). SMU and RACF provide the same functions as do SMU and exit routines: signon verification and resource access security. Two RACF security variations, not supplied by SMU and exit routines, are terminal-user security and resource class.

**Related Reading:** For more information on security, see *IMS/ESA Administration Guide: System* and *IMS/ESA Customization Guide*.

Signon verification, combined with transaction authorization and password checking, allows you to control the processing at input time. The resource definition in the Security Maintenance utility must declare the transaction name, even when the transaction is not processed in the system where the security tables are built.

Security maintenance in a multisystem environment is performed independently for each system. Password security is verified on terminal input after execution of the Terminal Routing exit routine. Terminal security is verified at the time of terminal input and after an application's use of a DL/I CHNG call if the call is issued in the input system.

When a destination system receives a message to process, it verifies security, based on the input terminal's association with the logical link path. Using link security, you declare transactions in the Security Maintenance utility input of the destination system that are authorized for each link path name. This prevents transactions that are sent by unauthorized systems from being processed.

User verification and transaction authorization are checked at the time of terminal input. Transactions received by directed routing on a link are passed to the transaction authorization module for authorization checking, but because the password is not passed across the link, transaction authorization checking fails if a password is required. Transactions that don't require a password can be accepted.

## Security Considerations for Multiple Systems

To allow a transaction to be scheduled in a destination system, you can authorize its processing with an installation-written security exit routine, or you can use RACF. The transaction must be declared in the Security Maintenance utility input for the processing system as part of the application group name (AGN) table for the dependent region.

If the RACF security environment is not available in the destination system (as when a /SIGN ON command is entered with RACF), the security environment is dynamically created to allow the transaction authorization to proceed.

Local terminals that need access to a local secured transaction must be defined to the Security Maintenance utility in order to have that authority.

---

## Establishing Operating Procedures for Multiple Systems

Each system in a multisystem configuration is operationally an independent unit. Each system exclusively owns its own communication resources, which are controlled by its own master terminal.

## Initializing Multisystem Communication

Communication between two IMS systems using non-VTAM connections does not begin until the /RSTART LINK command is issued in each system. When the physical link type is controlled by VTAM, only one side needs to issue the /RSTART LINK command. The normal procedure is for the master terminal operator to issue this command when a system is started up. Communication is allowed only if the characteristics of the specified links are compatible. If a required link is not successfully started, messages wait until the links are reassigned.

The SYSID of each IMS system running in a shared queues group (SQG) can either be cloned across all systems, or not. If the SYSIDs are not cloned, then all IMS systems in the SQG must be initialized so that they are all running at the same time. Once all IMS systems have been initialized, transaction processing can begin, and individual IMS systems can be brought down and later restarted as needed. Initialization allows all IMS systems to exchange MSC SYSIDs and MSNAMES, and creates a SYSID table that contains all of the SYSIDs for that SQG. If the SYSIDs for all IMS systems in a SQG *are* cloned, then you do not need to perform this initialization.

If a system that has messages queued in it is cold started, these messages are lost. Because the messages that were lost can be from or to terminals and programs in other systems, the impact of a cold start is not limited to the cold-started system.

## Terminating Multisystem Communication

A /PSTOP LINK command from either of two linked systems terminates transmission on the specified link. When transmission is terminated on one side, its partner in the other system terminates its own transmission and notifies the master terminal operator.

The command forms shown in Table 9 on page 207 are used for physical link termination between partner MSC systems.

## Establishing Operating Procedures for Multiple Systems

Table 9. Commands to Terminate Physical Links

Command	Keyword	MSC Use/Effect
/PSTOP	LINK PURGE	<p>Transmissions associated with the logical link between two partner systems are halted, but queuing for the remote resources continues. Broadcast messages that would use the link are queued but not sent.</p> <p>The logical link is stopped in the partner system and its MTO is notified by message DFS2161.</p> <p>This command applies to CTC link type only, and forces a /PSTOP condition even if the other system fails while CTC I/O is in progress.</p>
/IDLE	LINK NOSHUT	<p>Forces termination of all transmissions on a physical link associated with the named logical link. Use only after shutdown checkpoint.</p> <p>This command applies to SDLC link type only, and forces the /IDLE condition without a checkpoint shutdown.</p>

### IMS Commands That Affect MSC Operation

This section contains information about the use of IMS commands that have a particular effect relative to the use of MSC facilities.

**Related Reading:** For the definition and format of these commands, see *IMS/ESA Operator's Reference*.

### Logical Link Assignments

Initial logical link assignments (logical link to physical link) are made during IMS system definition. Use the /MSASSIGN command to dynamically make or change a logical link assignment. Use this approach primarily for unscheduled reassignments resulting from failing physical connections or systems.

Because a logical link must **always** communicate with its partner, the master terminal operators of the two systems must coordinate their assignments to a corresponding physical link. Any type of physical link can be replaced by any other type of physical link.

The /MSASSIGN command is used to alter the logical assignments of MSC system resources. All such changes remain in effect until they are changed by another /MSASSIGN command or until IMS is cold started. The relationship of MSC resources is altered only in the local system.

The relationships that can be altered include:

Logical Link            to    Physical Link  
(Rel Link No.)            (MSPLINK Name)

Remote Resource    to    Logical Link  
(SYSID)                    (Rel Link No.)

Logical Link Path    to    Logical Link  
(MSNAME Name)        (Rel Link No.)

The /MSASSIGN command can also change a remote program to local, or it can change a local program to remote and assign a logical path.

## IMS Commands That Affect MSC Operation

When you use the /MSASSIGN command:

- Operating logical or physical links must be assigned one-to-one, except for a VTAM-type link with parallel sessions in effect. In this case, multiple logical links can be assigned to one physical link.
- Logical links must be halted by /PSTOP and must be idle before being reassigned.
- A destination SYSID cannot be reassigned to a logical link unless its currently assigned logical link is idle after a /PSTOP.
- MSC communication can occur only when logical links in two IMS systems share the same partner identification and have assignment to an operating communications facility between the systems.
- The /MSASSIGN command does not determine whether the requested assignment is reasonable or results in a valid configuration for MSC communication. This is only done by communication with appropriate remote systems, you can accomplish this using the /MSVERIFY command after making assignment changes.

## Restarting a Logical Link

If a restart is pending on a logical link because of a physical link failure, both systems should use the following procedure to reestablish communication through an alternative physical link:

1. Reassign the logical link to the alternate physical link.
2. Use the /RSTART LINK command to start the logical link.

If the link uses VTAM, the /PSTOP MSPLINK command must be executed before assignment either to or from the VTAM MSPLINK name.

## Commands That Help Control Resources

You can review the commands shown in Table 10 to assess their effects in your MSC environment.

If an MSC error occurs when the MSC trace is not operational, check the log for X'67' records, because some error information is logged even when the link trace is not set on.

The following use of the /TRACE command can be used when link problems are suspected:

```
/TRA SET ON LINK X LEVEL 3 MODULE DDM
```

*Table 10. MSC Environment Commands*

Command	MSC Use and Effect
/ASSIGN	Change transaction priorities.
/CHANGE	Change link mode table and session restart.
/DELETE	Remove password security.
/PSTOP	Queue for remote processing, but do not send.
/PURGE	Reject subsequent primary requests for remote processing. Accept secondary or continued conversational requests. Continue sending those requests that are allowed to queue.



## IMS Commands That Affect MSC Operation

Table 10. MSC Environment Commands (continued)

Command	MSC Use and Effect
/STOP	(Remote program or terminal stopped in input system) Return DFS065 to input terminal if a primary request. Queue but do not send secondary requests. (Local program or terminal accessed by remote MSC systems) Queue secondary requests. Return message DFS065 to input terminal in input system for primary requests. Message DFS065 identifies, by SYSID, the MSC processing system that rejected the message. Message is logged, but canceled.
/START	Reset previous /START and /PSTOP effects.
/BROADCAST	Use of the LTERM keyword value ALL sends the broadcast message to all terminals of the local system only. No general broadcast capability exists for remote MSC terminals. Their LTERMs must be defined in the broadcasting system and specified in the command. Broadcasts to MTOs of remote MSC systems are possible even when the MTO terminals are not defined in the broadcasting system; use the keyword MASTER. The system must have defined the SYSIDs of such remote systems by MSNAME macros.
/TRACE	Using the LINK keyword, you can invoke a trace of MSC operation. Trace data is recorded within type X'67' records on the IMS system log.

## Using the /DISPLAY Command

The IMS /DISPLAY commands show available information that helps you manage a Multiple Systems Coupling environment. These facilities operate only within the domain of the local system, however, and do not provide information that is available only from remote systems. Table 11 summarizes the information that is available to the MTO with the /DISPLAY command.

Table 11. MSC Information from the /DISPLAY Command

Command	Keyword	MSC Use/Effect
/DISPLAY	ASSIGNMENT	Display current assignment of logical link path (MSNAME) to logical link, and logical link to physical link
	LINK	Logical link (#)
		Physical link (MSPLINK name)
		Local and remote SYSID's by MSNAME
		Resynchronization for SDLC link type (forced or not)
	MSNAME	Same as link display
	MSPLINK	Logical link (#)
		Physical link (MSPLINK name)
		Physical link type (MTM, VTAM, CTC)
		Physical link address
Maximum parallel session number for SDLC link type		
SYSID	Same as link display	
/DISPLAY	LINK	Logical link (#)
		Partner ID
		Number of messages received, sent, enqueued, dequeued, currently queued for link, link status mode table names, and ASR status

## IMS Commands That Affect MSC Operation

Table 11. MSC Information from the /DISPLAY Command (continued)

Command	Keyword	MSC Use/Effect
/DISPLAY	MSNAME	Logical link path
		Total messages dequeued for MSNAME
		Messages currently queued for MSNAME

## Logical Link Path Control

Several commands are used to control logical link paths. Logical link paths are defined by SYSID pairs that identify sending and destination systems. Logical link paths are named by the MSNAME macro. The logical link path is the lowest level of control across an MSC environment, because it is the lowest level that is necessarily defined in intermediate MSC systems. Table 12 summarizes the commands used for link control.

Table 12. Commands Used to Control MSC Link Paths

Command	Keyword	MSC Use/Effect
/START	MSNAME	Start a previously stopped MSNAME.
/STOP	MSNAME	Stop the sending and receiving of primary request messages associated with the logical link path.  When stopped by an input system, primary requests for remote programs or terminals associated with the stopped logical link path are canceled, and message DFS065 is returned to the input terminal. Conversations in progress are allowed to continue.  When stopped by a destination system, messages received from other systems over a stopped logical link path cause a logical link path to be stopped in the sending system (input or intermediate), and MTOs of the sending and receiving systems to be notified by messages DFS2140 and DFS2142, respectively. The message remains queued in the sending system until the logical link is subsequently started in both systems.
/PURGE	MSNAME	Queuing of primary requests for all remote terminals and programs represented by the MSNAME is halted. Continued conversations and secondary requests are handled. Primary requests entered through an input terminal receive DFS065. Requests from other systems that require use of the logical link path for a response are not accepted but remain queued in the sending system. See /STOP MSNAME.

## Recovery Considerations

Each system in the multisystem configuration uses the full recovery capabilities of IMS. These capabilities assure that messages are not lost or duplicated within the single system, as long as no cold start is performed and no log records are lost. VTAM provides message integrity for SDLC in each IMS system link, in addition to the MSC control functions.

## Message Recovery

MSC assures that messages are not lost or duplicated across a multisystem link, as long as no system in the configuration is cold started and no log records are lost. This is accomplished by logging information about a transmission in both the sending and receiving systems. This information is restored during restart and exchanged between the systems after the link is started. The sending system can then dequeue a message that was received by the receiving system but for which the acknowledgment was lost because of a link or system failure. The sending system can also resend a message that was sent but not enqueued by the receiving system because of a failure in the receiving system. If a system in the multisystem configuration fails to recover, the messages for which it has recovery responsibility are lost.

Because IMS provides commands to dynamically change link assignments, an inoperable processor can be backed up by an alternate processor. The IMS system that resides in the inoperable processor can be executed in the alternate processor after all involved links are properly reassigned by the master terminal operators.

## Stopped Transactions

If a destination transaction code is stopped for queuing, the action taken by the destination system varies based on the type of request:

- For a primary request that is not conversational or that starts a conversation, IMS sends an error message to the input terminal and cancels the message.
- For a primary request that continues a conversation or a secondary request, IMS queues the message. If the request is the first one received for that stopped transaction, IMS also sends a message to the master terminal at that transaction's local system.

## Application Program Abnormal Termination

When an application program abnormally terminates, and the abnormal termination is not the result of a deadlock situation, a DFS554A message is sent to the master terminal of the system in which the abnormal termination occurred. If the input message is still available, an error message (DFS555I) that includes the first portion of the input message is sent to the input terminal. When the error message to the input terminal is sent, the DFS554A message includes the logical terminal name of the input terminal.

## Dequeuing Messages, Responses, and Transactions

You can use the /DEQUEUE command with the PURGE or PURGE1 keyword to dequeue messages, responses, and transactions that are in error for a specific MSC link. When you enter the /DEQUEUE MSNAME PURGE command, the Message Control/Error exit routine (DFSCMUX0) is called before the command executes. The exit routine protects you from inadvertently entering the command in error and potentially destroying a message, response, or transaction by dequeuing it before it reaches its destination. The exit routine can suppress the dequeue of a message.

The system programmer must be aware of each of:

- The existence of the /DEQUEUE command, and the consequences of using it
- The interaction of the /DEQUEUE command with the Message Control/Error exit routine

## Recovery Considerations

The MTO should be aware of each of the following:

- Your installation policy for using the /DEQUEUE command
- The fact that any messages that are dequeued are destroyed
- The /DEQUEUE command should be reserved for emergency use only, or as dictated by your site

### **Recommendations:**

- Only authorize the MTO to issue the /DEQUEUE command.
- Establish installation-specific standards for using the /DEQUEUE command. Be sure to identify in advance and validate the scenarios where the MTO should use this command. Consider the /DEQUEUE command when coding the Message Control/Error exit routine.

### **Related Reading:**

- For more information on the /DEQUEUE command with the PURGE keywords, see *IMS/ESA Operator's Reference*.
- For more information on coding the Message Control/Error exit routine, see *IMS/ESA Customization Guide*.

---

## Monitoring and Tuning Multiple Systems

Plan to obtain both statistical and performance data for IMS online systems that are part of a multisystem network. You can use the same monitoring tools that are used for generating performance data for single IMS systems:

- You can execute the IMS Monitor in several systems concurrently. You obtain IMS Monitor reports for each individual IMS system and coordinate your processing analysis.
- The Statistical Analysis utility produces summaries of transaction traffic for each individual system. Again, you combine the statistics for a composite picture.
- The IMS Transaction Analysis utility enables you to trace transactions across multiple systems and examine the traffic using the various active physical links.

## Coordinating Performance Information

If possible, expand your MSC network by increasing the number of SYSIDs (up to 2036) and the number of physical and logical links (up to 676). By expanding the MSC network, you can:

- Access an IMS subsystem from many other IMS subsystems
- Route transactions
- Distribute transaction processing
- Increase network throughput
- Grow beyond the capacity of one IMS system
- Respond to capacity constraints or response-time constraints

The IMS system log for each system participating in MSC contains only the record of events that take place in that system. However, logging is performed to record traffic on the links. Add the SYSIDs of all coupled systems to the system log documentation that records the checkpoint intervals. This helps to interpret reports, because you are aware of transactions that might be present in message queues but are not processed, and you can expect additional transaction loads from remote sources.

## Monitoring and Tuning Multiple Systems

Your analysis procedures should include ways of isolating the processing that is triggered by transactions originating from another system. Considerations for tuning buffers for the asynchronous communication processing should include a criterion that no exceptional conditions resulting from intersystem traffic exist.

To satisfy the need for monitoring with typical activity that includes cross-system processing, coordinate your scheduling of the DC Monitor and other traces between master terminal operators. The span of the monitoring does not need to be exactly the same, but if it is widely different, the averaging of report summaries might make it more difficult to interpret the effect of the processing that is triggered by cross-system messages.

## Interpreting IMS Monitor MSC Reports

The IMS Monitor Report Print Program includes three reports that highlight message events caused by system coupling:

- MSC Traffic Report, which shows, for the monitor interval, the counts of messages using the various link paths.
- MSC Summaries, which shows summaries of the traffic queues for each input transaction name, each destination name, each link number, and each destination system.
- MSC Queuing Summary Report, which is generated when intersystem messages are queued on the local system before the messages are sent to the destination system. The local system must be an intermediate system. This report shows:
  - Maximum time messages spend in queues
  - Average time messages spend in queues
  - Maximum queue lengths
  - Total number of messages queued for all links the local system participates in

All three reports can have entries in the Distribution Appendix, so that you can examine the frequency distributions of the traffic if you suspect unusual transmission patterns.

**Related Reading:** For more information on the IMS Monitor Report Print Program, see *IMS/ESA Customization Guide*.

## Determining Cross-System Queuing

The MSC Traffic report indicates the individual queue loads for all traffic between partner systems for which the monitored system is the local system. The report lists all the unique system identification numbers (SYSIDs) that are defined for communications for that local system. It then summarizes the total number of messages queued and dequeued for each combination of the following variables:

- Input name (terminal or program that was a message source)
- Destination name (terminal or program)
- Input system (SYSID)
- Destination system (SYSID)
- Link number
- Link type (MTM, CTC, or VTAM)

## Monitoring and Tuning Multiple Systems

Figure 38 is a sample MSC Traffic report. If a message originates in the local system, its presence is accounted for in the dequeue counts only. Messages with local destinations appear only in the enqueue count.

```

IMS Monitor  **MSC TRAFFIC REPORT**  TRACE START 1993 209 13:30:32  TRACE STOP 1993 209 14:03:49  PAGE 0016
LOCAL SID VALUES = 18, 19, 21, 22, 23, 24, 25, 26, 42

INPUT  DESTIN.  INPUT  DEST.  LINK  LINK  ENQUEUE  DEQUEUE
NAME   NAME      SID    SID    NO.   TYPE  COUNT    COUNT
-----
T3270V T3270V    11     11     1     VTAM  0        22
          13     13     2     VTAM  0        1
          12     12     1     VTAM  0        2
          21     21     1     VTAM  4        0
          TRAN21P0 11     21     1     VTAM  2        0
          CONV21V0 11     21     1     VTAM  22       0
          ITRL     11     11     1     VTAM  0        5
          CONV12V0 11     11     1     VTAM  0        2
          21     11     1     VTAM  0        3
          CONV12V1 12     12     1     VTAM  0        1
          CONV21V1 12     22     1     VTAM  3        0
          LTERM10V 11     11     1     VTAM  0        2
          21     11     1     VTAM  0        4
          CONV21C0 13     23     2     VTAM  1        0
          UTR10U6 14     24     3     VTAM  4        0
          FTRL     11     21     1     VTAM  4        0

TOTAL TRAFFIC                                40        42

```

Figure 38. MSC Traffic Report

## Assessing the Effect of Link Loading

The MSC Summaries report provides a picture of the enqueue and dequeue activity for messages that are handled by the local system but that are also a part of the MSC traffic.

The report format is illustrated in Figure 39.

The first set of queuing counts shows how many transactions of each type were entered in the monitor interval and how many were subsequently dequeued.

The second set of counts summarizes the total traffic for each destination name. You can distinguish the primary transactions and responses by the resource names and examine the relative servicing of the link transmissions using the difference between the enqueue and dequeue counts.

The third set of counts lists the active links by link number, and you can tell if buildup on the link exists by the difference in the enqueue and dequeue counts.

The fourth set of counts records the traffic that is going to other systems by any link path. By looking at the difference in enqueue and dequeue counts, you can determine whether the overall pattern of link priorities to one or more systems is causing buildup of cross-system traffic.

```

IMS Monitor  ****MSC SUMMARIES****  TRACE START 1993 13:30:32  TRACE STOP 1993 14:03:49  PAGE 0015

<---SUMMARY OF INPUT NAME---><---SUMMARY BY DEST. NAME---><---SUMMARY BY RELATIVE LINK---><---SUMMARY BY DEST. SYS. ID--->
INPUT  ENQUEUE  DEQUEUE  DESTIN.  ENQUEUE  DEQUEUE  LINK  ENQUEUE  DEQUEUE  DEST.  ENQUEUE  DEQUEUE
NAME   COUNT    COUNT    NAME     COUNT    COUNT   NO.   COUNT    COUNT  SID   COUNT    COUNT
-----
*NONE*  0        1        LINK21B1  0        1        1     12     14     11
          13     14
T3270L1 25     25        T327011  13     13        4     13     12     31     12     12
          CONV31B0 7        7
          TRAN31B0 5        5

```

Figure 39. MSC Summaries Report

## Assessing Link Queuing Times

The MSC Queuing Summary report provides information about intersystem message traffic only. You can use the sample of traffic recorded in the IMS Monitor interval in order to examine the maximum and average time messages spend in queues waiting to be sent on active links. You can detect whether the link priorities are causing undue delay of primary messages through the intermediate system, or whether a buildup of responses exists. The report shows the logical link paths for this system, which is an intermediate system. Each incoming link number shows the number of messages that are queued before transmission on their specified outward-bound link number. The maximum queue count is given, as well as the maximum and average time on the intermediate system queues.

The report is illustrated in Figure 40.

```
IMS Monitor *****MSC QUEUING SUMMARY***** TRACE START 1993 13:30:32 TRACE STOP
1993 14:03:49 PAGE 0016
```

ENQUEUE.....		DEQUEUE.....		MESSAGES	MAX. 0 LENGTH	MAX. 0 0 TIME	MEAN 0 TIME	DIST. NUMBER
LINK NO.	TYPE	LINK NO.	TYPE					
4	VTAM	1	VTAM	13	1	287011	101902	5
1	VTAM	4	VTAM	12	1	282596	73531	8
3	VTAM	3	VTAM	26	1	287012	73641	7
TOTALS.....				51			83021	

Figure 40. MSC Queuing Summary Report

## Extracting Multiple-System Transaction Statistics

You can use the Log Transaction Analysis utility to obtain counts of the message traffic both in local systems and between systems. The transmissions over the different types of physical links can also be examined. (The activity is summarized for each step of the logical link paths.) You must provide IMS system log input that reflects all partner system activity, that is, sets of system logs for each MSC system. To coordinate the sets of individual system logs, use the Log Merge utility. As many as nine separate system logs can be merged, each log being the output of a uniquely identified IMS system with MSC installed.

## Controlling the Log Merge

To control the log output, you need to:

- Choose the required systems that participate in the logical link paths you want to examine.
- Coordinate the series of input logs for each system so that they cover a similar time span.
- Specify a start and stop time for the Log Merge utility control statements if you want to sample the cross-system processing for a particular interval.

You can give both start date (Julian) and time of day, or just time of day. It is the first system log (specified by the LOG01 DD statement) to which these times apply. Other log activity is collected if it falls between the initial and final events that are present on the first log.

- Specify MSG to select log records that are suitable for the transaction analysis step. (ALL records is the default, but this means the DL/I activity for several systems is included in the utility input, and this can cause extended processing time.)

## Monitoring and Tuning Multiple Systems

### Interpreting the Transaction Analysis Report

Using the Log Analysis report produced by the IMS Transaction Analysis utility, you can obtain the following statistics for individual transactions that are processed in any system:

- The total response time
- The time on input and output queues
- The processing time

**Related Reading:** For an illustration of this report, see *IMS/ESA Utilities Reference: Transaction Manager*. It also includes definitions for the format of the detailed report records produced by the IMS Transaction Analysis utility and a list of processing type codes. The absence of times for a message GU call or MPP termination in the report lines indicates an input source or intermediate system report line.

The processing type field is an important one for the interpretation of the detailed report lines. The S code indicates that this line shows a send or receive event for the transaction. You can trace the progress of a cross-system conversation using the codes C, D, P, X, and Y.

The report headings include a column headed ID after the column for the GU to the message queue time. The number shown in a report line under the ID heading matches the sequence in which log input is fed to the Log Merge utility. The field corresponds to starting position 102, the 3-digit field named SYSTEM ID, in the detailed report records.

You can use the sort step to reorder the report records in any order you want, for example, by system ID within transaction code. The default order is the input sequence.



---

## Part 4. Administering Intersystem Communication

<b>Chapter 10. Overview of Intersystem Communication</b>	221
What Is ISC?	221
Comparing ISC to MSC	222
IMS Facilities Available to ISC	223
Distributed Transaction Processing	223
Distributed Services	224
IMS Transaction Types	224
IMS Execution Modes	224
IMS Editing Facilities	225
ISC Message Integrity	225
ISC Security	225
ISC in an XRF Complex	225
Sample System Configurations	226
ISC between IMS and CICS	228
Terminal Device-Dependent Data	229
Passing CICS Data to IMS	229
Pass-Through Input to IMS	229
Output from IMS	229
<b>Chapter 11. VTAM Facilities Used for ISC</b>	231
VTAM Commands and Indicators	232
Using the VTAM Application Program Interface	233
Specifying Logon Modes When Establishing Connection	233
Design Considerations for Secondary Logical Units	234
<b>Chapter 12. IMS Facilities Affected by ISC</b>	235
Editing Messages	235
Issuing IMS Commands from an ISC Session	236
Effects on Parallel Sessions	236
Using IMS Test Mode for ISC Sessions	237
IMS Control Block Storage on ISC Parallel Sessions	237
Relationship of ISC and IMS Execution Modes	238
External Specification of Execution Modes	238
Internal Definition of Execution Mode	238
Resultant Processing Mode	239
LTERM Users (Subpools) and Components	240
<b>Chapter 13. Designing Communications Using the ISC Protocol</b>	243
Determining Output Protocols	243
Accessing Existing Application Programs with ISC	244
Accessing Programs That Use MFS	245
Accessing Programs That Do Not Use MFS	245
Routing Messages	246
FM Headers for Message Routing	246
IMS Use of Routing Parameters	247
Routing Examples	247
Routing Messages through MSC to an ISC LTERM	255
Considerations for IMS-to-IMS ISC Sessions	255
Coexistence of ISC and MSC VTAM within One IMS Subsystem	255
Defining Single and Parallel Sessions	255
Ensuring Compatible Buffer Sizes	256
Remote Control of IMS via ISC	256
Restriction on IMS-to-IMS Conversation Mode	256

Routing Transactions to the Back-End IMS . . . . .	256
Sending Messages between the Subsystems . . . . .	257
Protocol Restrictions on IMS-to-IMS Sessions . . . . .	257
Statically Defining an ISC Node to IMS . . . . .	257
Choosing Parameters: System Design Considerations . . . . .	260
COMM Macro Statement . . . . .	260
NAME Macro . . . . .	261
SUBPOOL Macro . . . . .	261
TERMINAL Macro . . . . .	262
Specifying the OUTBUF Parameter . . . . .	262
Specifying the MSGDEL Parameter . . . . .	262
Specifying the COMPTn Parameter . . . . .	262
Specifying the SINGLE or MULT Parameter . . . . .	263
Specifying the VLVB or DPM-B1...DPM-B15 Parameter . . . . .	263
Specifying the IGNORE or 1, 2,...10 Parameter . . . . .	263
System Definition Summary . . . . .	263
<b>Chapter 14. ISC Protocol Guide and Reference . . . . .</b>	<b>265</b>
Operating the Network . . . . .	266
Making IMS Ready . . . . .	266
Bringing Up an IMS Network . . . . .	266
Shutting Down an IMS Network . . . . .	266
Controlling the Session (Session Control Protocols) . . . . .	267
Initiating a Session . . . . .	267
Establishing Connection with the XRF Complex . . . . .	268
Binding the Session . . . . .	268
Negotiable versus Nonnegotiable BIND . . . . .	269
Binding Single or Parallel Sessions . . . . .	269
Resolving a Bind Race . . . . .	269
After a Successful Session Bind . . . . .	270
Resynchronizing Sessions . . . . .	270
Designing Restart Resynchronization Procedures . . . . .	271
Maintaining Sequence Numbers . . . . .	271
Recovering Sessions with Cold Start . . . . .	272
Controlling Unilateral Decisions about Pending Work Units . . . . .	272
Recovering from In-Bracket Failures . . . . .	273
Response or Conversational Output Available at Restart . . . . .	273
Session Failures without IMS Failure . . . . .	274
Session Failures because of IMS Failure . . . . .	274
Recoverability of Commands and Execution Modes . . . . .	275
Coordinating the Restart Process . . . . .	275
Determining Session Synchronism Using STSN . . . . .	276
Performing the Resynchronization . . . . .	277
Completing Session Initiation . . . . .	278
Running the Session . . . . .	279
Terminating the Session . . . . .	279
Normal Termination . . . . .	279
Abnormal Termination . . . . .	280
Using STSN to Resynchronize Sessions . . . . .	280
Primary-to-Secondary Flow Matrix . . . . .	280
Secondary-to-Primary Flow Matrix . . . . .	282
STSN Command Format . . . . .	283
Controlling Data Flow (DFC Protocols) . . . . .	285
Handling IMS Response Mode or Conversational Output Errors . . . . .	285
Response Mode Errors . . . . .	286
Conversational Mode Errors . . . . .	286

Normal Conversation Termination Extension with ISC . . . . .	287
Keeping Half Sessions Synchronized . . . . .	287
Sync Points Requested on Input to IMS . . . . .	288
Sync Points Requested on Output by IMS . . . . .	289
Sync Point and Response Requirements . . . . .	289
Recoverable Messages . . . . .	290
Irrecoverable Messages . . . . .	291
Sync-Point Indicators on Messages . . . . .	292
Requests on IMS Input Messages . . . . .	292
Requests on IMS Output Messages . . . . .	293
Data Flow Control Protocol Reference . . . . .	295
BID Protocol . . . . .	295
Bracket and Half-Duplex Protocol . . . . .	296
Bracket Protocol for IMS Input . . . . .	296
Bracket Protocol for IMS Output . . . . .	299
CANCEL Protocol . . . . .	304
Chaining Protocol . . . . .	304
CHASE Protocol . . . . .	305
ERP Purging . . . . .	305
Resulting DFC State after Sender ERP Purge . . . . .	306
Resulting DFC State after Selective Receiver ERP Purge . . . . .	307
LUSTATUS Protocol . . . . .	308
Paged Messages ERP . . . . .	311
Ready-to-Receive Protocol . . . . .	312
RSHUT Protocol . . . . .	312
Selective Receiver ERP . . . . .	312
Selective Receiver ERP Sense Codes Supported . . . . .	313
X'0864xxxx'—Function Abort . . . . .	313
X'0865xxxx'—Function Abort . . . . .	314
X'0866xxxx'—Function Abort . . . . .	314
Receiver-Detected Errors during Data Flow Reset State . . . . .	316
Sender ERP . . . . .	316
Sender ERP Sense Codes . . . . .	317
Sender-Detected Errors on Nonpaged Messages . . . . .	318
Sense Codes that IMS Receives . . . . .	319
Sense Codes that IMS Sends . . . . .	319
SIGNAL Protocol . . . . .	320
Symmetrical Session Shutdown for LU 6.1 (SBI and BIS) . . . . .	320
Function Management Headers . . . . .	321
Using FM Headers to Invoke ISC Edit . . . . .	323
Initiating a Process: ATTACH FM Header . . . . .	323
Error Recovery Procedure FM Header . . . . .	324
Resetting the Active Process: RAP FM Header . . . . .	324
Requesting Asynchronous Transaction Processing: SCHEDULER FM Header . . . . .	325
System Message Process (SYSMSG) and Related FM Headers . . . . .	326
<b>Chapter 15. Using MFS with ISC . . . . .</b>	<b>329</b>
Activating MFS Input Formatting . . . . .	329
Activating MFS Output Formatting . . . . .	330
MFS Distributed Presentation Management (DPM) Messages . . . . .	330
MFS Page Delete Function . . . . .	331
MFS Online Error Detection . . . . .	331
Output Errors . . . . .	331
Input Errors . . . . .	331
Paging Errors . . . . .	332

The ATTACH and SCHEDULER FM Headers under MFS . . . . .	333
Data Descriptor FM Headers . . . . .	334
Input Data Descriptor FM Header . . . . .	334
Output Data Descriptor FM Header . . . . .	334
Controlling Demand-Paged Messages: QMODEL FM Headers . . . . .	335
Request (Input) QMODEL FM Headers . . . . .	336
QGETN FM Header . . . . .	336
QGET FM Header . . . . .	336
QPURGE FM Header . . . . .	337
Reply (Output) QMODEL FM Headers . . . . .	337
QXFR FM Header . . . . .	337
QSTATUS FM Header . . . . .	337
The RAP FM Header under MFS. . . . .	338
<b>Chapter 16. FM Header Format Reference . . . . .</b>	<b>339</b>
ATTACH FM Header Format . . . . .	339
ATTIU . . . . .	341
ATTDSP . . . . .	341
ATTDBA . . . . .	342
ATTDPN . . . . .	343
ATTPRN . . . . .	345
ATTRDPN and ATTRPRN . . . . .	346
ATTDQN and ATTDQ . . . . .	347
ATTACC . . . . .	347
Data Descriptor FM Header . . . . .	347
Input Data Descriptor FM Header Format . . . . .	347
Output Data Descriptor FM Header Format . . . . .	348
Error Recovery Procedure FM Header . . . . .	349
QMODEL FM Headers . . . . .	349
QGET FM Header Format . . . . .	349
QGETN FM Header Format . . . . .	350
QPURGE FM Header Format . . . . .	351
QSTATUS FM Header Format . . . . .	352
QXFR FM Header Format . . . . .	353
Reset Attached Process (RAP) FM Header Format . . . . .	355
SCHEDULER FM Header Format . . . . .	355
SYMSMSG Process Headers . . . . .	356
SYSERROR FM Header Format . . . . .	356
SYSSTAT FM Header . . . . .	356

---

## Chapter 10. Overview of Intersystem Communication

This chapter introduces the System Network Architecture (SNA) for Intersystem Communication (ISC) and the IMS implementation of this architecture.

### **In this Chapter:**

- “What Is ISC?”
- “Comparing ISC to MSC” on page 222
- “IMS Facilities Available to ISC” on page 223
- “Sample System Configurations” on page 226
- “ISC between IMS and CICS” on page 228

---

### What Is ISC?

ISC is a part of the IMS Transaction Manager. It is one of the ways to connect multiple subsystems. The other means of connection is Multiple Systems Coupling (MSC).

As defined under SNA, ISC is an LU 6.1 session that:

- Connects different subsystems to communicate at the application level.
- Provides distributed transaction processing permitting a terminal user or application in one subsystem to communicate with a terminal or application in a different subsystem and, optionally, to receive a reply. In some cases, the application is user written; in other cases, the subsystem itself acts as an application.
- Provides distributed services. Thus, an application in one subsystem can use a service (such as a message queue or database) in a different subsystem.

SNA makes communication possible between unlike subsystems and includes SNA-defined session control protocols, data flow control protocols, and routing parameters. The functions provided by each of these protocols and parameters are summarized below, and the IMS support for them is described in this chapter.

**Related Reading:** For more information on the SNA protocols and parameters, see “Chapter 14. ISC Protocol Guide and Reference” on page 265.

Session control (SC) comprises:

- Initiating sessions between subsystems
- Recovering and resynchronizing sessions, maintaining the integrity of session states and recoverable resources across both session and subsystem failures
- Terminating any or all session paths between IMS and another subsystem

Data flow control (DFC) includes:

- Controlling send and receive protocols within a session.
- Resolving contention for transaction initiation within a session.
- Monitoring error recovery processing.
- Monitoring symmetrical shutdown accomplished using the SNA commands stop bracket initiation (SBI) and bracket initiation stopped (BIS).
- Controlling synchronization of resources. Sync point control ensures that all resources are committed or backed out synchronously.

## What Is ISC?

ISC routing comprises:

- Using parameters in the SNA-defined function management headers to connect the process required for incoming messages and to route reply messages.

APPC/IMS only supports message switching to LU 6.2 destinations through the DFSAPPC service.

**Related Reading:** For information on the DFSAPPC service, see:

- “DFSAPPC System Service” on page 390
- “ETO and LU 6.1 (ISC) Terminals” on page 167

Before looking in detail at ISC, let’s compare it to MSC.

---

## Comparing ISC to MSC

IMS provides two ways to couple multiple subsystems: MSC and ISC.

Multiple Systems Coupling (MSC) is an IMS protocol that enables coupling of IMS systems to other IMS systems only. ISC, however, allows you to connect IMS subsystems with any other subsystem that supports the ISC protocol. This other system can be another IMS, CICS, or a user-written system.

MSC supports three types of links between IMS systems—channel-to-channel (CTC), memory-to-memory (MTM), and VTAM. In the balance of this chapter, MSC is used to identify the functions provided by any of these links. ISC uses only one kind of link—LU 6.1. Figure 41 compares MSC to ISC.

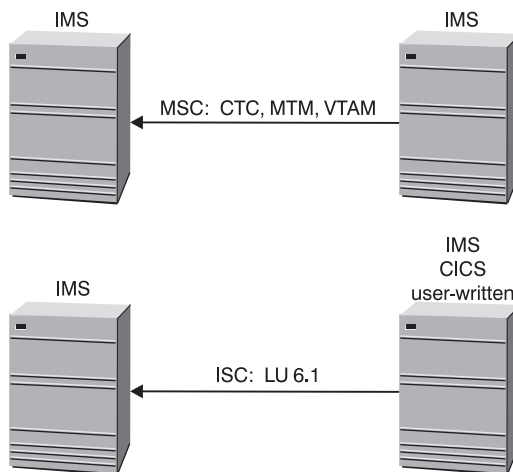


Figure 41. IMS Multiple Systems Coupling and Intersystem Communication

Both MSC and ISC enable a user to:

- Route transactions
- Distribute transaction processing
- Grow beyond the capacity of one IMS system

Also, both ISC and MSC-VTAM take advantage of the parallel session support VTAM provides. Some key differences exist, however. The following table highlights the major functions of MSC-VTAM and ISC, and shows the differences in support.

Table 13. Comparing MSC and ISC Functions

MSC functions	ISC functions
MSC connects multiple IMS systems only to each other. These IMS systems can all reside in one processor, or they can reside in different processors.	ISC can connect either like or unlike subsystems, as long as the connected subsystems both implement ISC. Thus, a user can couple an IMS subsystem to: <ul style="list-style-type: none"> <li>• Another IMS subsystem</li> <li>• A CICS subsystem</li> <li>• A user-written subsystem</li> </ul>
Communication in the MSC environment is subsystem-to-subsystem.	Communication is between application programs in the two subsystems. The subsystems themselves are session partners, supporting logical flows between the applications.
Processing is transparent to the user. That is, to the user, MSC processing appears as if it is occurring in a single system.	Because ISC supports coupling of unlike subsystems, message routing requires involvement by the terminal user or the application to determine the message destination. Specified routing parameters can be overridden, modified, or deleted by MFS.
When not using the MSC-directed routing capability, the terminal operator or application program does not need to know routing information. Routing is automatic based on system definition parameters.	ISC provides a unique message-switching capability that permits message routing to occur without involvement of a user application.
MSC supports the steps of a conversation to be distributed over multiple IMS subsystems, transparent to both the source terminal operator and to each conversational step (application).	ISC supports the use of Message Format Service (MFS) in an IMS subsystem to assist in the routing and formatting of messages between subsystems.
MSC does not support the use of the Fast Path Expedited Message Handler (EMH).	ISC supports the use of Fast Path Expedited Message Handler (EMH) between IMS subsystems.

**Related Reading:** For more information on MSC, see “Chapter 9. Administering Multiple Systems Coupling” on page 193.

## IMS Facilities Available to ISC

This section describes a variety of IMS facilities that are available to ISC.

### Distributed Transaction Processing

When supporting distributed transaction processing, IMS can be either the front-end or back-end processor. A front-end subsystem originates communications traffic (transactions, commands, or message switches) that are to be processed by the back-end subsystem. A message is typically the result of data entered at a terminal. However, the message can also result from processing that occurred in an application within the front-end subsystem. The back-end subsystem processes input messages from another subsystem.

In the most typical configuration, IMS is the back-end system and processes IMS transactions that another subsystem enters.

## IMS Facilities Available to ISC

In addition, special support exists for front-end or back-end system utilization is provided by the Front-End Switch exit routine.

**Related Reading:** For more information on the Front-End Switch exit routine (DFSFEBJ0), see *IMS/ESA Customization Guide*.

Sometimes, transactions entered into an IMS front end are sent with an ISC message switch to the other subsystem. In this case, the terminal operator or MFS provides the message routing information. The transaction can also be sent as alternate PCB output, in which a user-written application program or MFS format definition provides the routing information.

The fact that IMS is a queued subsystem is a key factor in determining the ISC functions that it supports, particularly when IMS acts as a front-end processor. Messages created by a terminal operator or an application in a front end IMS are queued for transmission to the receiving subsystem and sent asynchronously with respect to the terminal or application that sent the message. A terminal attached to an IMS front-end is not held in response mode during the receiving (back-end) subsystem's processing, unless it uses the Front-End Switch exit routine (DFSFEBJ0).

## Distributed Services

IMS supports distributed services by providing remote access to an IMS message queue. Although IMS does not support distributed DL/I calls, a DL/I database in one subsystem can be updated by another subsystem; this action requires the sending subsystem to invoke an updating application in the receiving subsystem.

## IMS Transaction Types

IMS supports the following transaction types:

- Recoverable-update (includes Fast Path)
- Recoverable-inquiry
- Irrecoverable-inquiry

**Related Reading:** For more information on transaction types, see *IMS/ESA Administration Guide: System*.

## IMS Execution Modes

IMS supports the following execution modes:

- Response mode (includes Fast Path)
- Conversational mode
- Exclusive mode
- Transaction preset mode
- Test mode
- Nonresponse or nonconversational mode

**Related Reading:** For more information on execution modes, see *IMS/ESA Administration Guide: System*.



## IMS Editing Facilities

Messages transmitted on the ISC session and processed within an IMS subsystem are edited by the following editing facilities:

- ISC edit (the default editor)
- Message Format Service (MFS)
- Basic edit

These editors can be selected on a message-by-message basis.

**Related Reading:** For a description of these editors, see “IMS Editing for Intersystem Communication (ISC)” on page 102.

## ISC Message Integrity

Message integrity is provided to prevent loss or duplication of input or output messages between IMS and another half session during session restart.

Message integrity and recovery are increased by log write-ahead (LWA). This facility is invoked during system definition by an option on the TRANSACT macro. LWA ensures that sync point information is written to the log (and thus available to IMS restart procedures) before IMS acknowledges the message.

**Related Reading:** For more information on message integrity, see “Resynchronizing Sessions” on page 270 and “Sync Point and Response Requirements” on page 289.

## ISC Security

IMS security facilities control access to IMS resources by another subsystem. Before implementing ISC security in IMS, examine the data protection facilities of the subsystems and associated operating system components. Some important ISC-environment security factors are described here.

Terminal operator verification and authorization are provided by the subsystem controlling that terminal connection.

Both IMS terminal and password security can be defined for static ISC terminals by using the Security Maintenance utility (SMU). If password security is defined, a password must be provided on the input message.

**Related Reading:** For more information on password security, see:

- “Password Security” on page 42
- “ETO and LU 6.1 (ISC) Terminals” on page 167
- *IMS/ESA Administration Guide: System*

## ISC in an XRF Complex

ISC sessions are handled as class–2 or class–3 terminals. If the active IMS system fails, all LU 6.1 sessions are terminated. The alternate (now active) IMS system reestablishes the sessions after it takes control of the complex. Users might be without service for a short time.

**Related Reading:**

## IMS Facilities Available to ISC

- For information on establishing an ISC session with an XRF complex, see “Establishing Connection with the XRF Complex” on page 268.
- For more information on XRF, see *IMS/ESA Operations Guide* and *MVS/ESA Planning: Extended Recovery Facility (XRF)*.

---

## Sample System Configurations

The following figures illustrate some IMS ISC configurations. The first two figures use the Customer Information Control System/Virtual Storage (CICS) product as an example of an ISC node.

Figure 42 illustrates ISC’s distributed transaction processing capability—the ability to invoke existing or new IMS transactions from a CICS application or from a CICS application on behalf of a CICS-attached terminal (using transaction routing).

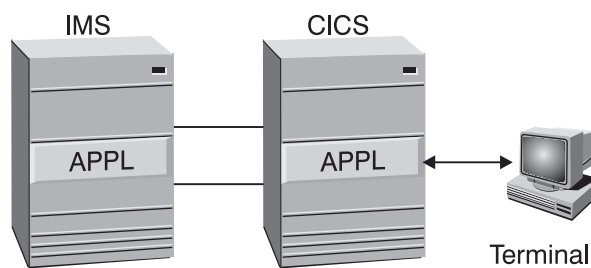


Figure 42. Existing or New IMS Transactions Invoked from CICS

In Figure 42, ISC provides transaction-to-transaction capability, because an application program can be written as two complementary transactions: one executing in an IMS system, the other in a CICS system. User functions can be distributed between systems as required.

Figure 43 on page 227 illustrates ISC’s ability to distribute device mapping function from an IMS Message Format Service (MFS) system to a CICS Basic Mapping Support (BMS) system. MFS partially maps the data stream that is sent to a CICS application. The application is responsible for processing the input data stream to a form that is acceptable to BMS. BMS can then be used to complete the device mapping.

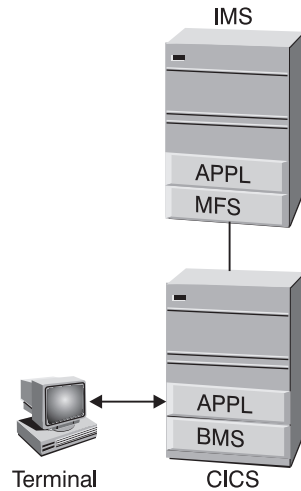


Figure 43. IMS MFS DPM Mapping Function Distributed to CICS's BMS

Figure 44 illustrates how two IMS systems can be connected using ISC. Interconnection of two IMS systems allows a new or modified transaction (TRAN1) in one IMS system to invoke an existing transaction (TRAN2) in another IMS system. Using MFS, replies can be routed to a new transaction (TRAN3) in the initiating IMS system, or to a new instance of the originating transaction (TRAN1). Without MFS, replies are treated as message switches and routed to the source terminal.

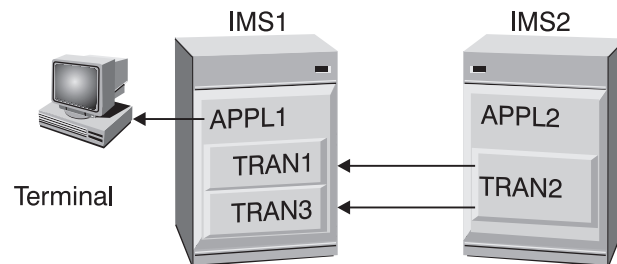


Figure 44. IMS-to-IMS with ISC

Figure 45 on page 228 illustrates how a CICS subsystem can communicate with an IMS subsystem that has MSC links to other IMS subsystems. In this case, MSC is used to couple the IMS subsystems, while the CICS-to-IMS session is supported by ISC. The CICS subsystem has a single-system view of the multiple IMS subsystems with which it can communicate. MSC distributes the load between the multiple IMS subsystems.

## Sample System Configurations

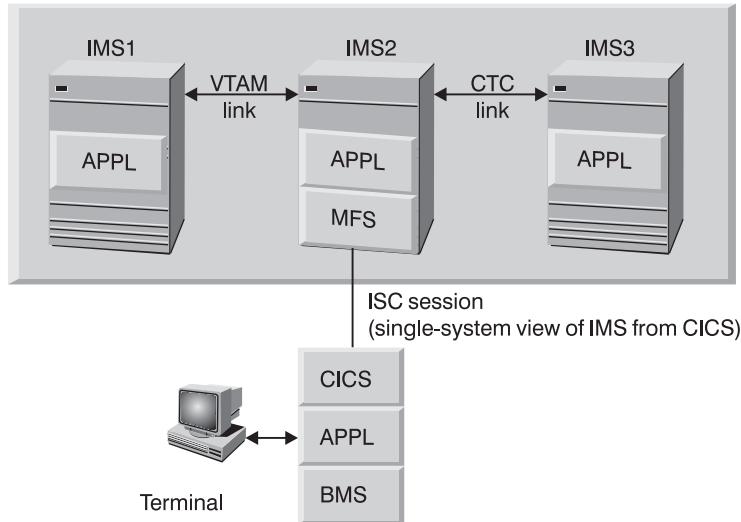


Figure 45. CICS-to-IMS MSC Using ISC

The intersection of functions provided by IMS and CICS is summarized in the next section.

**Related Reading:** For a sample program illustrating the use of ISC between IMS and CICS, see “Appendix J. Sample Program for IMS-CICS ISC” on page 551.

## ISC between IMS and CICS

Table 14 identifies the ISC functions that are provided by CICS and IMS, and the IMS facilities that are available to users of CICS START or RETRIEVE, and SEND or RECEIVE.

**Related Reading:** For more information on the interaction of IMS and CICS in an ISC session, see “Appendix G. How IMS and CICS Use the ISC Interface” on page 497.

Table 14. Facilities Available to CICS START RETRIEVE and SEND RECEIVE USER

Function	Front-End Processor	Back-End Processor
Distributed Transaction Processing Initiate remote transaction	CICS <sup>1</sup>	CICS <sup>1</sup>
	CICS <sup>1</sup>	IMS nonresponse type
	IMS all transaction types <sup>2</sup>	CICS <sup>1</sup>
	IMS all transaction types except conversational <sup>2</sup>	IMS all transaction types except conversational <sup>2</sup>
Distributed Transaction Processing Application-to-Application Conversation	CICS <sup>3</sup>	CICS <sup>3</sup>
	CICS <sup>3</sup>	IMS Response Conversational Fast Path

Table 14. Facilities Available to CICS START RETRIEVE and SEND RECEIVE USER (continued)

Function	Front-End Processor	Back-End Processor
Distributed Presentation Management	IMS MFS map	IMS MFS map
Using distributed transaction processing	CICS user application <sup>4</sup> IMS MFS map	IMS MFS map CICS user application <sup>4</sup>

**Note:**

1. Using CICS START or RETRIEVE commands.
2. If the transaction in the processing subsystem is a response mode or Fast Path transaction, a nonresponse transaction type is required in the originating system to handle the reply from the processing subsystem.
3. Using CICS SEND or RECEIVE commands.
4. The user application can optionally invoke BMS to aid in mapping functions between CICS and the terminal.

## Terminal Device-Dependent Data

ISC, when used with IMS, is a technique that allows programs in various subsystems (IMS, CICS, and user-written) to exchange messages with application programs in other subsystems. To minimize or eliminate the need to modify these programs, the messages that are exchanged should not contain device control characters that are unique to the originating or destination terminal.

The device control editing facilities of IMS (MFS) or CICS (BMS) should be used to remove these device control codes prior to the initiating program creating its ISC message. Likewise, the program that sends the response to a terminal should use the formatting service of its own subsystem to properly format the message before sending it. This way, all inter-program message exchanges can be independent of the originating and destination terminals. This means that each terminal's owning system should understand all formats for transactions that can be entered or received by terminals on that system.

## Passing CICS Data to IMS

Users want their front-end application programs to receive 3270 data streams from an attached terminal and to simply pass through this data stream (device control characters) to the back-end IMS. The resulting output from IMS should then be a 3270 data stream that could be passed through the front-end application and out to the 3270.

### Pass-Through Input to IMS

It is possible to send a 3270 data stream from a front-end application to IMS. However, IMS does not recognize the ISC session as a 3270 device and does not edit the device control characters from the data stream. Your IMS application program must provide this editing function.

### Output from IMS

Similarly, IMS does not place 3270 device control characters in an output data stream that is destined for an ISC LTERM, because the ISC LTERM is not a 3270 device. Your application program must then construct a 3270 data stream to be sent to a front-end application that can, with minimal editing, send this data stream to an attached terminal.

## **ISC between IMS and CICS**

This effort needs to be repeated for each device and each application program, and again after any change to a device format.

---

## Chapter 11. VTAM Facilities Used for ISC

VTAM controls the physical transmission of data between IMS and a logical unit. Both VTAM applications (such as IMS) and other subsystems can be viewed as VTAM logical units.

**Definition:** A *logical unit* is an addressable resource, such as an application program or a subsystem. A logical unit can also be a component of a general-purpose terminal system.

**Related Reading:** For more information on the communication concepts and facilities that govern data transmission between a VTAM application program (such as IMS) and another subsystem, see *VTAM V4R3 for MVS/ESA Programming*.

### **In this Chapter:**

- “VTAM Commands and Indicators” on page 232
- “Using the VTAM Application Program Interface” on page 233
- “Specifying Logon Modes When Establishing Connection” on page 233
- “Design Considerations for Secondary Logical Units” on page 234

The VTAM concepts and facilities used by IMS that are particularly relevant to ISC include:

- Connection, disconnection, and establishing logon mode.
- Messages and responses.
- Request definite response 1 (RQD1) or request definite response 2 (RQD2) and the associated definite responses (DR1 or DR2).<sup>29</sup>
- Sequencing and chaining.
- Orderly session disconnection via stop bracket initiation (SBI) and bracket initiation stopped (BIS) commands.
- Facilities for ensuring orderly communication, including the use of brackets and change-direction indicators.
- Sequence number recovery.
- Receiving input and sending messages.
- Conditional bracket termination.
- Extended Error Recovery Procedure (EERP).
- Use of parallel sessions between ISC nodes.
- Support for both negotiable and nonnegotiable session bind parameters.

### **Related Reading:**

- For more information on SBI and BIS commands, see “Symmetrical Session Shutdown for LU 6.1 (SBI and BIS)” on page 320.
- For more information on using parallel sessions between ISC nodes, see “Using the VTAM Application Program Interface” on page 233.

IMS also supports the class-of-service (COS) and session-outage-notification (SON) facilities.

---

29. Definite response 1 and definite response 2 have been separated and redefined for LU 6.1 protocols. RQD2 requests and DR2 responses are now known as sync-point requests and responses and are functionally independent of those responses associated with DR1. “Resynchronizing Sessions” on page 270 explains response requirements in detail.

## VTAM Facilities Used for ISC

### Related Reading:

- For more information on the COS and SON facilities, see *IMS/ESA Administration Guide: System*.
- For more information on all of these communications concepts and facilities, see *Systems Network Architecture Format and Protocol Reference: Architectural Logic*.

The ISC system programmer and system analyst must be familiar with these concepts and facilities in order to design and implement a communications interface between IMS and a remote subsystem using SNA LU 6.1 protocols.

---

## VTAM Commands and Indicators

VTAM commands and indicators (communication control information) are necessary for data transmission between an IMS application program and another VTAM logical unit. Table 15 shows which VTAM commands and indicators that IMS sends and receives during an IMS session. Results are unpredictable if any unsupported commands or indicators are used.

*Table 15. VTAM Commands and Indicators Sent and Received by IMS*

<b>VTAM Command or Indicator</b>	<b>IMS Sends as Primary Half Session, Receives as Secondary Half Session</b>	<b>IMS Sends as Secondary Half Session, Receives as Primary Half Session</b>
	<b>X = Supported</b>	<b>X = Supported</b>
Independent of Session Initiate session Procedure error Terminate session	Note 1	Note 4 Note 1 Note 5
Session Control Commands Bind Bind response Set-and-Test-Sequence- Numbers (STSN) STSN response Start data traffic (SDT) Start data traffic response Unbind	X  X X X	X  X X
Normal (synchronous) Flow Indicators Begin-Bracket (BB) End-Bracket (EB) Change-Direction (CD)	X X X	X X X
Commands BID CANCEL CHASE Logical Unit Status (LUSTATUS) Ready-to-Receive (RTR) Bracket Initiation Stopped (BIS)	Note 3 X X X X	X X X Note 2 X
Expedited (asynchronous) Flow Commands Request-Shutdown (RSHUT) Signal Stop Bracket Initiation (SBI)	X X	Note 2 X X



Table 15. VTAM Commands and Indicators Sent and Received by IMS (continued)

VTAM Command or Indicator	IMS Sends as Primary Half Session, Receives as Secondary Half Session	IMS Sends as Secondary Half Session, Receives as Primary Half Session
------------------------------	--	--

**Note:**

1. Sent by VTAM.
2. Not sent by IMS, but optionally sent by the other subsystem and received by IMS when IMS is a primary half session.
3. Not sent by IMS, but optionally sent by the other subsystem and received by IMS when IMS is a secondary half session.
4. Supported by internal VTAM forms of CINIT (LOGON exit).
5. Supported by internal VTAM forms of CTERM (NS and LOSTERM exits).

## Using the VTAM Application Program Interface

The VTAM application program interface (API) used by IMS for ISC consists of macro instructions and control blocks to allow IMS to:

- Establish connection or disconnection.
- Request and control the transfer of data between IMS and another logical unit.

To provide application-to-application communication, the VTAM API provides both a primary and a secondary session application interface. The primary interface is essentially the same as that used by IMS to support other VTAM nodes. However, the interface introduces some new macros and some additional parameters on existing macros.

**Related Reading:** For more information on defining IMS to VTAM, see:

- “Special Considerations If Your System Requires VTAM” in *IMS/ESA Installation Volume 1: Installation and Verification*
- *VTAM V4R2 for MVS/ESA Resource Definition Reference*

The VTAM API supports single sessions as well as parallel (multiple, simultaneously active) sessions between two logical units. The IMS interface for this support is primarily through the session qualifier fields of CINIT and BIND, which are available through the LOGON and SCIP exit routines. Single-session support requires static definition of message queues during IMS system definition; parallel-session support permits message queues to be dynamically allocated at session initiation.

## Specifying Logon Modes When Establishing Connection

When establishing connection, IMS requires that session parameters define the rules a logical unit must follow when communicating with IMS. These session parameters are contained in the VTAM mode table.

The use of the default logon mode table entry can be overridden in one of two ways:

- At system definition, the logon mode entry can be specified on the TERMINAL macro using the MODETBL=keyword or an ETO logon descriptor.
- The MODETBL keyword on the /OPNDST command can specify the logon mode table entry to replace the table entry defined at system definition.

**Related Reading:**

## Specifying Logon Modes When Establishing Connection

- For information on specifying the MODETBL keyword, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.
- For information on specifying the /OPNDST command, see *IMS/ESA Operator's Reference*.

If neither method is used to specify a logon mode table entry, VTAM uses the first entry in the default logon mode table, unless it is overridden by VTAM node mode table entry definition. If a logon mode table entry is specified, VTAM searches the default or user-specified logon mode table for the specified entry.

IMS examines the set of session parameters within the indicated VTAM mode table entry and overlays only those parameters on which IMS has dependencies. The remaining bytes are not changed. IMS ignores any unformatted user data transmitted with the session parameters.

### **Related Reading:**

- For more information on the session parameters, see "Appendix B. Bind Parameters for SLU P and LU 6.1" on page 471.
- For more information on establishing logon mode tables and defining logon mode table entries, see *VTAM V4R2 for MVS/ESA Resource Definition Reference*.

---

## Design Considerations for Secondary Logical Units

When IMS and another subsystem establish an ISC session, one session partner is the bidder and the other the first speaker (the contention winner). In an IMS ISC session, the bidder is always the primary logical unit (the bind sender), and the first speaker is always the secondary logical unit (the bind receiver). When providing a system design for a secondary logical unit that is to communicate with IMS within an ISC network, consider the following points:

- The first speaker (secondary logical unit) must resolve bracket contention and be prepared to receive interrupts (for example an VTAM command or bracket indicator) at all times. If the first speaker is not prepared to do so, a deadlock can result (for example, if both the primary logical unit and the secondary logical unit use the VTAM SEND macro, and issue POST=RESPONSE rather than POST=SCHED).
- When terminating a session using the VTAM TERMSESS macro, the system designer should be aware of the differences involved in issuing CONDITIONAL versus UNCONDITIONAL. Sending CONDITIONAL permits the primary logical unit to clean up and control session termination. Sending UNCONDITIONAL causes immediate termination of the session by VTAM prior to notification of the primary logical unit. If the primary logical unit is in a suspended state as, for example, when waiting for a response, the system designer should be aware that UNCONDITIONAL session termination can have unanticipated results.

Throughout the remainder of this book, the term secondary logical unit is synonymous with first speaker and the term primary logical unit with bidder.

---

## Chapter 12. IMS Facilities Affected by ISC

This chapter explains how IMS facilities are affected by ISC processing.

**Related Reading:** For more information on IMS facilities available to an ISC session, see:

- *IMS/ESA Administration Guide: System*
- *IMS/ESA Application Programming: Transaction Manager*

**In this Chapter:**

- “Editing Messages”
- “Issuing IMS Commands from an ISC Session” on page 236
- “Effects on Parallel Sessions” on page 236
- “Using IMS Test Mode for ISC Sessions” on page 237
- “IMS Control Block Storage on ISC Parallel Sessions” on page 237
- “Relationship of ISC and IMS Execution Modes” on page 238

---

### Editing Messages

Within IMS, the communication interface is transparent to application programs. In support of this, IMS provides some common editing that:

- Appends protocols and function management headers to outbound messages
- Strips function management headers from incoming messages and moves the messages to appropriate points within the IMS control blocks

Subsequent to this common editing, messages received by IMS are edited by basic edit, ISC edit, or Message Format Service (MFS) in accordance with the input ATTACH or SCHEDULER DPN parameter. Messages sent by IMS are edited by Message Format Service before they are sent. The edit functions of ISC edit and basic edit are unique. However, the associated protocols are the same.

- ISC edit is the default editor. When ISC edit is requested and that request does not contain the SNA-defined PRN parameter following the input data, the message is edited with partial input data transparency. That is, only the IMS-required destination code and optional password, which must occur at the beginning of the message, are subjected to editing by ISC edit. The balance of the input message is not edited prior to processing unless the transaction is defined to translate to uppercase. Full input data transparency is provided through ISC edit using the SNA-defined PRN parameter on the ATTACH or SCHEDULER FM header. This enables the IMS transaction code to be external to the message and enables IMS to receive input and route it to its proper destination (terminal or transaction) without examining the input message itself. Because the transaction code is external to the data stream, the optional password is not available to the session. Therefore, password security should not be defined. If password security is defined, an error results.

**Related Reading:**

- For more information on the SNA-defined PRN parameter, see “FM Headers for Message Routing” on page 246.
- For more information on ISC edit, see “Using FM Headers to Invoke ISC Edit” on page 323.
- Message Format Service’s Distributed Presentation Management (DPM) option divides responsibility for message formatting between MFS and a program

## Editing Messages

residing in another subsystem. When using DPM-Bn, physical terminal characteristics are not defined to MFS. Instead, IMS sends MFS-formatted data to a program in the other subsystem. That program must complete the formatting, if necessary, and present the data to the physical device. DPM also allows user involvement in specifying input and output ISC message routing parameters. You can also use the PRN parameter on the ATTACH or SCHEDULER FM header to specify the IMS destination externally, relative to the resulting MFS-formatted input data stream.

**Related Reading:** For more information on MFS and the DPM option, see *IMS/ESA Application Programming: Transaction Manager*

- Basic edit provides standard editing for terminal input. It is used to edit device and operator control characters and can be used in an ISC session when it is important to maintain input compatibility for existing applications. Protocols for basic edit are the same as those described above for ISC edit. Use of the ATTACH or SCHEDULER FM header PRN parameter has no effect when basic edit is used. The message destination is determined from the first field of the input data stream.

**Related Reading:** For more information on the editing functions of basic edit, see “Basic IMS Edit Functions” in *IMS/ESA Administration Guide: System*.

IMS permits basic edit, ISC edit, or MFS to be invoked in order to handle communication of transactions, commands, and message switches. MFS DPM facilities are defined as optional on a component-by-component basis. Any of these available processes are selectable on a message-by-message basis by using the ATTACH or SCHEDULER FM header.

ISC input and output can be edited by most IMS data communication exit routines. The exceptions are the MSC-related exit routines, and the hardware-required routines.

---

## Issuing IMS Commands from an ISC Session

Although available to the ISC session, IMS operator commands are primarily intended for use by appropriately authorized operators of IMS master terminals and remote terminals that are directly attached to an IMS system. Terminal operators of other (non-IMS) subsystems do not communicate directly with IMS, but rather with the control program or user-provided applications within the other subsystem. That subsystem must determine and provide procedures for its terminal operators and application programmers to follow when communicating with IMS. If an application in a non-IMS subsystem is permitted to issue IMS commands via the ISC session, IMS dependencies are introduced into that application. An ISC half session cannot be defined as the primary or secondary master terminal.

---

## Effects on Parallel Sessions

**Recommendation:** Be sure to include the USER keyword when using authorized IMS commands with the NODE keyword. Equivalent action is taken against all active, or potentially active, parallel sessions of the indicated node when a specific parallel session instance is not identified to IMS via the USER keyword.

**Example:** Issuing the /CLSDST NODE x command without specifying USER schedules termination of all active parallel sessions of NODE x. Issuing the /STOP NODE x

command without specifying USER schedules termination of all active parallel sessions of NODE x and also prevents any new parallel sessions from being initiated on NODE x.

---

## Using IMS Test Mode for ISC Sessions

You can test ISC communication protocols and editing facilities by putting a back-end IMS system into test mode.

While in test mode, IMS receives an input message, checks input protocols and FM header parameters, performs input and output editing, inserts appropriate output protocols and FM header parameters, and returns the message to the front-end subsystem. Noncommunication error analysis, transaction code verification, and transaction scheduling are bypassed. Also, IMS does not send asynchronous output while in test mode.

You can place one or all of the ISC sessions within a specific subsystem in test mode when the IMS /TEST command is received on the ISC sessions. The /TEST command is entered locally in the back-end IMS processor by an authorized terminal operator. You can terminate test mode by sending an end-bracket (EB) indicator on an SNA LUSTATUS or CHASE command. Test mode can also be terminated when an /END command is received on the ISC session or is entered locally in the back-end IMS processor by an authorized terminal operator.

In an IMS-to-IMS environment, use the ISC message switch facility to place the back-end IMS system into test mode, to send the message and receive the echo reply, and to terminate test mode. This provides an efficient means of testing without requiring a user application in either subsystem.

---

## IMS Control Block Storage on ISC Parallel Sessions

The storage required for the IMS control block structure representing potential sessions is greater than the storage required for control blocks representing input and output message queues. Therefore, installations needing only a few logical units type 6.1, each having a relatively small number of associated parallel sessions, but each requiring that a large number of ISC users (subpools or LTERM sets) be dynamically allocated to them, can have lower storage requirements. For static ISC terminals, these users (subpools) are still statically defined. The IMS control block structure for ISC single session (statically defined and allocated LTERMs) is the same as for other static VTAM terminal types within IMS.

**Related Reading:** For information on the IMS control block structure for ISC parallel sessions, see “Appendix H. ISC Data Flow Control Examples” on page 539.

You do not need to define a maximum number of ETO-ISC sessions. You can continue to add sessions until you run out of storage and processor capacity. Therefore, the SESSION= keyword is not one of the supported keywords on the ETO descriptor. To define the maximum number of sessions for static ISC sessions, use the SESSION keyword on the TERMINAL macro.

**Related Reading:** For more information on actual storage requirements for ISC, see *IMS/ESA Administration Guide: System*.

---

### Relationship of ISC and IMS Execution Modes

Because the terms “synchronous” and “asynchronous” have slightly different connotations within IMS and ISC, this section explains the relationship of these execution modes.

### External Specification of Execution Modes

Messages in ISC use the SNA-defined function management headers. ISC-routed messages can be processed either synchronously or asynchronously, as determined primarily by the type of FM header and secondarily by the VTAM bracket protocols sent with the message. A summary of supported protocols appears in Table 16. Using FM headers and VTAM bracket protocols to establish the mode of the message (synchronous or asynchronous) can be considered to be external to the message.

#### **Related Reading:**

- For more information on the SNA-defined function management headers, see “Function Management Headers” on page 321.
- For more information on VTAM send, receive, and bracket protocols that are sent with ISC messages, see “Controlling Data Flow (DFC Protocols)” on page 285.

*Table 16. Processing Mode Requested by FM Headers*

	<b>Synchronous</b>	<b>Asynchronous</b>
ATTACH with CD	X	
ATTACH with EB		X
ATTACH without either EB or CD	X	
SCHEDULER with CD	X	X
SCHEDULER with EB		X
SCHEDULER without either EB or CD	X	X

Messages received by IMS with the ATTACH FM header are processed:

- Synchronously, if the session state is left “in-brackets” after the message has been received.
- Asynchronously, if either of the following occurs:
  - The session state is left “between-brackets” after the message has been received.
  - The message is sent with EB on the first- or only-in-chain.

During synchronous processing, unsolicited or asynchronous output is not sent.

All messages received by IMS with the SCHEDULER FM header are processed asynchronously by the receiver with respect to the session.

### Internal Definition of Execution Mode

The IMS internal execution mode determines how IMS processes a transaction. Response mode and conversational mode, for example, are used to ensure synchronism between a given input message and its associated reply. Although a transaction uses an IMS input and output message queue between the session and the application, these modes ensure that associated messages are always

## Relationship of ISC and IMS Execution Modes

processed synchronously with relation to the source session. While these messages are being processed, IMS does not send asynchronous or unsolicited output on that session.

**Related Reading:** For more information on IMS message types and internal modes of execution, see *IMS/ESA Administration Guide: System* .

Within IMS, internal definitions are either synchronous or asynchronous, as indicated in Table 17.

Table 17. Internal Definitions

Synchronous	Asynchronous
Response mode	Nonresponse or nonconversational mode
Conversational mode	Message switch
Fast Path	
Commands	
Test mode	

## Resultant Processing Mode

During ISC communications, when the relationship between the externally requested execution mode and the internally understood processing mode are consistent, the message is processed exactly as requested. In the case in which the two specifications (internal versus external) are not consistent, the message's execution with respect to the session is synchronous.

In either case, if the generated reply is returned on the same session as the request, it is sent using the same type of FM headers that were received with the input message. If the reply is returned on a session other than the one on which the input message was received, it is considered unsolicited asynchronous output and is sent with the SCHEDULER FM header. If the subsystem that is to receive the reply does not support receipt of the SCHEDULER header, the reply is sent with an ATTACH header that terminates the bracket at the end of the IMS message. Therefore, with the exception of a nonlast conversation, nonlast MFS demand-paged messages, or test mode output, ATTACH is used instead of ATTACH SCHEDULER and EB is indicated on the first or only chain of the message. A Nonlast conversation, nonlast MFS demand-paged messages, and test mode output are sent carrying ATTACH with CD indicated on the first or only chain of the message.

However, when a message is sent from the front-end subsystem to the back-end subsystem that is externally requesting synchronous processing, no assumptions should be made by the front-end subsystem as to the timing of the output reply or the availability of any other output for that session. As a result, requests and replies cannot be correlated within the front-end subsystem, even though execution in the back-end subsystem might have been synchronous. This is summarized in Table 18 on page 240.

The special support for front-end and back-end system utilization Front-End Switch exit routine provides.

**Related Reading:** For more information on the Front-End Switch exit routine, see *IMS/ESA Customization Guide*.

## Relationship of ISC and IMS Execution Modes

Table 18. Internal versus External Execution Mode Specification

Internal Specification (System Definition)	External Specification (FMH)	
	Synchronous <sup>1</sup>	Asynchronous <sup>2</sup>
Synchronous	Synchronous	Synchronous <sup>3</sup>
Asynchronous	Synchronous <sup>4, 5</sup>	Asynchronous

**Notes:**

1. Synchronous specification does not include ATTACH with EB indicated on the first or only chain of the message.
2. Asynchronous specification includes ATTACH with EB indicated on the first or only chain of the message.
3. Synchronous operation in IMS is allowed in this case, because the SNA-defined external asynchronous format allows IMS as the message receiver to execute an inbound message according to its own interpretation. Thus, IMS's internally synchronous definition overrides the external asynchronous format. If, however, IMS's internal (system definition) mode of execution is synchronous and the message is received with ATTACH EB, IMS generates an error message and terminates the session.
4. This support allows response mode to be established dynamically (on a message-by-message basis), even though the message was internally defined to be asynchronous (nonresponse mode).
5. Messages cannot be externally defined as synchronous when OPTIONS=NORESP is specified internally on the IMS TERMINAL macro or ETO user descriptors. This conflict between system definition and FM header intent causes session termination.

Traffic between two IMS systems is always sent in asynchronous format. If the internal definition of the transaction within the back-end system is synchronous, it is processed synchronously and the ISC session is held for synchronous output. However, this same relationship does not apply for the source terminal. The front-end IMS does not hold that terminal in response mode until the reply is received from the back-end subsystem, unless you are using the Front-End Switch exit routine.

**Related Reading:** For more information on the Front-End Switch exit routine, see *IMS/ESA Customization Guide*.

## LTERM Users (Subpools) and Components

**Definition:** *IMS user blocks* are sets of IMS logical terminals (LTERMs) defined by the SUBPOOL macro during IMS system definition or dynamically created from ETO user descriptors. Subpools defined at system definition cannot be used with ETO LU 6.1 terminals. Users (subpools) defined for ISC are separate from the users defined for dial-type terminals and are permitted only in conjunction with ISC parallel session support.

**Definition:** The collection of all static ISC users is known as the *VTAMPOOL*. LTERMs defined within the VTAMPOOL are reassignable only between users within the VTAMPOOL. LTERMs not defined within the VTAMPOOL cannot be assigned into the VTAMPOOL.

ISC users are dynamically assigned to an ISC session instance as a result of session initiation. That is, these parameters define the user that is available to the given session instance. This user remains allocated to the named parallel session instance even across session and subsystem failures until released through normal



## Relationship of ISC and IMS Execution Modes

termination by mutual agreement of IMS and the other subsystem. For a single nonparallel session, the allocation of a set of LTERMs is fixed during system definition.

**Related Reading:** For information on system definition for single and parallel sessions, see *IMS/ESA Installation Volume 2: System Definition and Tailoring* .

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 can cause problems with input component determination or output presentation.

IMS uses the input component ID to identify the LTERM that is to be associated with the input message. For other terminal support, IMS assumes that all input is from the first LTERM in the list that passes the necessary operational and security checks. However, for input from an ISC node, the input component is determined based on the component that is indicated in the ATTACH FM header. If no FM header is received, IMS assumes the input is to be associated with the LTERM for input component (ICOMPT) one. After the component value is determined, if the associated LTERM cannot be found, is stopped, or is not ready, or if that LTERM cannot pass security checks, the message is rejected.

When output is sent, it is sent to the component (COMPT) identified with the output LTERM. Message switches, broadcast messages for specific LTERMs, and data replies from transactions are directed to the component that is associated with the specified output LTERM. The user-written MPP can insert to the I/O PCB and default to the output component that is associated with the selected input LTERM. It can also address specific components through the appropriate LTERM name by an insert to an alternate PCB.

You can establish proper relationships between input and output components through the NAME macro during IMS system definition or LTERM keyword on the ETO user descriptor. This enables a logical unit to indicate its input component and causes output to be returned to the associated output component that was indicated during IMS system definition or on an ETO user descriptor. Proper definition and use of components can reduce or eliminate the need for LTERM naming conventions, DL/I CHNG calls, and inserts to alternate PCBs.

**Recommendation:** Incorrectly specifying the message delete system definition parameter, MSGDEL, or the ETO user descriptor while defining ISC users (subpools) can prevent a session from being initiated. An ISC session can be initiated only if the MSGDEL specifications on the TERMINAL or ETO user descriptor and SUBPOOL macros match.

**Related Reading:** For more information on the TERMINAL macro statement, the SUBPOOL macro, and ETO user descriptor information, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Also, when using the ISC message switch support or an alternate PCB insert from a local transaction in order to route messages to another subsystem and return replies to a source terminal operator, MSGDEL=SYSINFO must be specified on the IMS system definition TERMINAL macro or ETO logon descriptor associated with both the ISC session and the source terminal. Specifying MSGDEL=NONIOPCB for

## Relationship of ISC and IMS Execution Modes

the ISC session prevents ISC-message routing to other subsystems. Specifying MSGDEL=NONIOPCB for the source terminal prevents message replies from being routed to the source terminal operator.

**Related Reading:** For more information on using an ISC message switch or an alternate PCB insert from a local transaction to route messages, see “IMS Use of Routing Parameters” on page 247.

The output bracket and send and receive protocols used by IMS depend on a combination of the system definition output component specifications, the protocol used to input the message to IMS, and the type of IMS message received. For example, IMS synchronous message types have a predefined protocol for output replies. Associated with the output component is a definition of send and receive protocols for asynchronous output.

**Related Reading:** For more information on the IMS output bracket protocol, see “Bracket Protocol for IMS Output” on page 299.

---

## Chapter 13. Designing Communications Using the ISC Protocol

This chapter is an overview of the topics that help you to design subsystem-to-subsystem communications that use the ISC protocol.

### In this Chapter:

- “Determining Output Protocols”
- “Accessing Existing Application Programs with ISC” on page 244
- “Statically Defining an ISC Node to IMS” on page 257
- “Choosing Parameters: System Design Considerations” on page 260
- “System Definition Summary” on page 263

---

### Determining Output Protocols

Traffic on the ISC session is controlled by the VTAM protocols associated with the message. The primary bracket protocols used are:

- Begin-bracket (BB)  
Signals the beginning of a bracket. Begin-bracket is an unconditional request when issued by the first speaker (secondary half session) and a conditional request when issued by the bidder (primary half session).
- End-bracket (EB)  
Signals the end of a bracket, places the session in contention state, and allows either session partner to request a new bracket.
- Change-direction (CD)  
Turns control of the session over to the session partner and allows the session partner to send traffic on the session.

Traffic across an ISC session can be sent and received synchronously to the session. The processing of messages is performed either synchronously or asynchronously to the session. That is, the sending subsystem initiates a message and waits for a reply (synchronous) or does not wait (asynchronous).

**Related Reading:** For more information on synchronous and asynchronous processing, see “Relationship of ISC and IMS Execution Modes” on page 238.

Within ISC, the external synchronous mode of operation (as specified via FM headers) requires transmitted messages and associated replies, if applicable, to occur within a single bracket. The end of synchronous mode is signaled by EB. When in synchronous mode, IMS must be able to reply, if required, within the same bracket (before EB is received) to any outstanding message traffic with the exception of a message switch. If the input message protocol prohibits IMS from sending any required replies within the same bracket, the input message is rejected.

Within IMS, the protocols used for output that must be sent with ATTACH (because the other subsystem lacks SCHEDULER support) are predefined, regardless of whether the originating input transaction and resulting output replies are synchronous (ATTACH) or asynchronous (SCHEDULER). This output includes:

- Nonlast chains (pages) of MFS paged output

## Determining Output Protocols

- The last (MFS paged) or only chain of response mode, conversation mode, test mode, and IMS command replies
- Asynchronous output

**Related Reading:** For more information on these predefined bracket and send/receive protocols, see “Bracket and Half-Duplex Protocol” on page 296.

You must define to IMS the protocols to be used with the last (MFS paged) or only chain of other asynchronous output. These are defined on the `COMPTn` keyword of the `TERMINAL` macro or on an ETO logon descriptor. Four parameters are supplied:

### **SINGLE1**

Asynchronous output for this component is sent one message per bracket.

### **SINGLE2**

Asynchronous output for this component is sent one message at a time with the VTAM begin-bracket (if necessary) and change-direction indicators to allow the receiving subsystem to optionally send its communication traffic.

### **MULT1**

All asynchronous messages for a given LTERM are sent before the bracket is ended.

### **MULT2**

All asynchronous messages for a given LTERM are sent before change-direction is sent.

**Related Reading:** For more information on these parameters, see “Bracket Protocol for IMS Output” on page 299.

For IMS-to-IMS sessions, the selection of `SINGLE1`, `SINGLE2`, `MULT1`, or `MULT2` can be related to the transaction types and characteristics of the receiving IMS subsystem. The following considerations apply:

### **SINGLE1**

This protocol is appropriate for sending message switches, nonresponse transactions, or nonconversational transactions to the receiving IMS.

### **SINGLE2**

This protocol is appropriate for sending response mode (including Fast Path) transactions, commands, and test mode, because synchronous communication assumes that input generates corresponding output. Conversational mode is not supported on an IMS-to-IMS session.

### **MULT1 or MULT2**

These protocols treat asynchronous traffic the same way as `SINGLE1` or `SINGLE2` treat synchronous traffic. They are useful in suppressing contention when large amounts of asynchronous traffic are queued for transmission on one (`MULT1`) or both (`MULT2`) subsystems.

---

## Accessing Existing Application Programs with ISC

During an ISC session, you can access application programs written prior to your installation's support of ISC. Consider the following information when planning to access these programs.

### Accessing Programs That Use MFS

If your application program uses MFS, is it sensitive to the MFS MODname? Your program is sensitive, for example, if the program tests the MODname in the IOPCB as part of its logic, or uses ISRT calls to the IOPCB that include a MODname as part of the call. If the program is sensitive, MFS must be used when input is received from an ISC session.

If your program is sensitive to cursor position, it cannot be used to receive ISC session input. An ISC LTERM is not a hardware device and, as a result, cursor position has no meaning.

For output, MFS provides support for attribute and extended attribute bytes.

**Related Reading:** For more information on support for attribute and extended attribute bytes for output, see the ATTR= keyword description of the DFLD statement for DPM-Bn in *IMS/ESA Application Programming: Transaction Manager*.

ISC supports the system control area (SCA) and default system control area (DSCA) in the following manner. All functions allowed for the 3270 display can be specified by the application program in a message field (MFLD) defined as an SCA. A device field (DFLD statement) can be defined as an SCA in the device output format (DOF). For ISC subsystems, MFS does not interpret the SCA specifications; it merely relays them in the user-defined device field SCA that it sends to the remote program or ISC subsystem. The MFS-supported SCA, or DSCA, for ISC session output can be used by the receiving application program to intelligently handle a terminal device attached to its subsystem.

If your application program is not sensitive to MFS formatting, it is possible for an ISC-connected subsystem to send an input data stream to your program without invoking MFS. To accomplish this, the sending subsystem must construct an input data stream that is exactly the same as your program's I/O area. This "bypassing" of MFS would result in a reduction of the processor overhead associated with MFS. Bear in mind that bypassing MFS in this manner can result in the transmission of inefficient data streams from the sending subsystem and also makes the sending subsystem sensitive to changes in your program's I/O area.

It is probably easiest, from an implementation point of view, to use MFS with ISC when MFS is already used for input to your program. In this case, you simply create ISC (DPM-Bn) device input formats (DIF) and DOFs.

### Accessing Programs That Do Not Use MFS

The two input edit choices for existing application programs that do not use MFS are ISC edit and basic edit.

ISC Edit is the default edit process for ISC input to IMS. An existing application program expects the transaction code to be at the start of the input message and to be followed by a blank. When this is true, the sending subsystem should ensure that the transaction code is placed at the start of the data stream, followed by a blank. Password security is for statically defined terminals only and can only be invoked by placing the password after the transaction code in the data stream and ensuring that the SNA-defined PRN in the FM header is not present. When ISC Edit is used, the input data stream sent to IMS should match the application program's I/O area exactly.

## Accessing Existing Application Programs with ISC

Basic edit for ISC input provides standard editing as if the input comes from a terminal. It is used to edit device and operator control characters. The sending subsystem could simply pass through to IMS a data stream from a device that would normally use basic edit if it were attached directly to IMS. In this situation, IMS's ISC basic edit support eliminates application-provided editing in the sending subsystem. Basic edit is selected by the sending subsystem when it places the characters 'BASICEDT' in the DPN field of the FM header.

## Routing Messages

This section introduces the routing fields in the SNA-defined FM headers and describes how IMS uses these fields to route messages (and thus support distributed processing).

### FM Headers for Message Routing

ISC message routing information is provided within SNA-defined function management (FM) headers. On input, IMS automatically processes the information in these FM headers and then removes them from the input message. IMS then passes the message to the appropriate input edit process. On output, IMS automatically builds the necessary routing FM headers based upon information provided by IMS system definition, session bind parameters, the input FM header, or MFS.

**Related Reading:** For more information on FM headers, see "Function Management Headers" on page 321.

For this explanation of distributed transaction processing, the portion of the ATTACH and SCHEDULER FM headers of interest are the routing parameters that they carry. Four fields (described more fully in sections "ATTDPN" on page 343 through "ATTDQN and ATTDQ" on page 347) are used to route messages to appropriate control blocks. These routing fields, present in both headers, are:

#### DPN (Destination Process Name)

The DPN parameter names an input process to be invoked synchronously to the session. Within IMS, the processes that can be invoked are basic edit, ISC edit, or MFS. For MFS, the input DPN value is the MFS message input descriptor (MID) name. For other subsystems, the named process can be the message destination (such as a transaction code).

#### PRN (Primary Resource Name)

The PRN parameter names a resource to be associated with the attached process. This parameter occurs on a reply returned as the result of processing a message in a remote subsystem.

Within IMS, the input PRN can name either an LTERM message queue for terminal output (IMS message switch) or an input transaction message queue. When the PRN is not available on input, IMS defaults to the normal method of using the message destination contained in the first data field (or optionally inserted by MFS formatting) or a preset transaction code established by an IMS operator command.

#### RDPN (Return Destination Process Name)

The RDPN parameter defines a suggested return destination process name to be associated with an output message. It should be returned as the DPN on resulting message replies to facilitate the processing of replies returned to the source subsystem. The input RDPN is discarded when replies or message switch output is sent on another session. The RDPN is sent by IMS

## Accessing Existing Application Programs with ISC

only for MFS-formatted output, wherein the RDPN is the optional chained MID name specified on the MFS message output descriptor (MOD).

### RPRN (Return Primary Resource Name)

The RPRN parameter defines a suggested return primary resource name to be associated with an output message. It should be returned on resulting message replies to the subsystem that was the source of the message to facilitate return reply processing within that subsystem. The input RPRN is discarded when replies or message switch output is sent on another session. The RPRN can optionally be set by use of MFS formatted output if specified on the MFS MOD.

**Related Reading:** For a summary of IMS actions relative to the DPN, PRN, RDPN, and RPRN fields in the ATTACH and SCHEDULER FM headers, see Table 38 on page 346.

## IMS Use of Routing Parameters

IMS ISC without the Front-End Switch exit routine does not provide for automatic transaction routing when IMS is the front-end subsystem. These transactions can be sent as alternate PCB output from a user-written application program within IMS or by using an ISC message switch. In the latter case, the source terminal operator provides the output routing information through an ISC destination LTERM name which is the required first data field on IMS message switches.

Except when the output is formatted by MFS (by using a MOD chained from the input MID associated with the message switch), IMS message routing support for ISC removes the first data field before sending the balance of the message text to the back-end subsystem. When MFS is used to format message switch output, you should use MFS editing to remove the output LTERM name.

Further, ISC message routing support within IMS defaults the RPRN within the output FM header sent with the message to the LTERM name associated with the terminal that was the source of the message switch. For alternate PCB output, the RPRN is set to the I/O PCB name. The RPRN allows IMS to automatically route message replies from the remote subsystem back to the source terminal in the form of a message switch, provided the other subsystem has returned the RPRN parameter as the PRN parameter on the reply. MFS DPM can be used to alter the FM header destination and return routing parameters. When using IMS as a front-end subsystem to route messages to another subsystem or route replies back to the source terminal, MSGDEL=SYSINFO must be specified on the IMS system definition TERMINAL macro statement or an ETO user descriptor associated with both the ISC session and the source terminal.

## Routing Examples

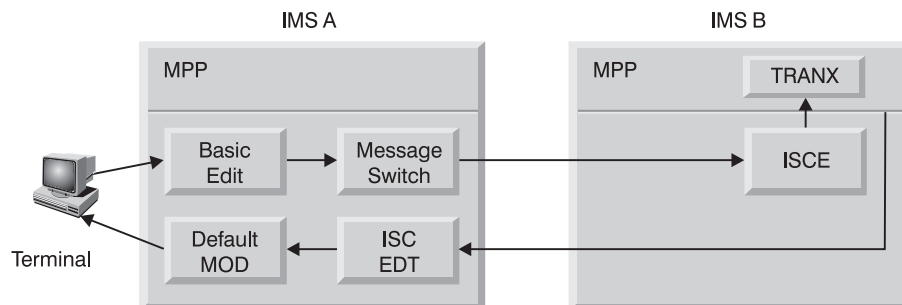
The following routing examples show typical functions that can be accomplished between two subsystems using ISC and how routing parameters are used to accomplish these functions. In the examples:

- The ISC edit alias is ISCE.
- The LTERM name for input and output associated with the terminal on the left in the diagrams is T.
- The LTERM name for input and output associated with the terminal on the right in the diagrams is T2.
- The LTERM name for the ISC session between the two subsystems is LTISC1.

## Accessing Existing Application Programs with ISC

**Related Reading:** For an example of the Front-End Switch exit routine, see *IMS/ESA Customization Guide*.

### Example 1. IMS-to-IMS Message Switch Routing:



**Note to Figure:** The name ISCE was defined for ISC edit during system definition by specification of EDTNAME=ISCE on the COMM macro in both subsystems.

The terminal attached to IMSA uses the ISC message switch to input TRANX to be executed in IMSB. An MPP in IMSB processes TRANX, and the reply is routed back to the terminal in IMSA, which was the source of the message switch. The output terminal reply appears as an output message switch.

1. Assume the terminal is a 3270 display device. Based on this assumption, basic edit can only be invoked by entering input from a cleared screen. Therefore, in this example, the terminal operator enters:

LTISC1 is the IMSA logical terminal name associated with an ISC session

LTISC1	TRANX	Data...
--------	-------	---------

between IMSA and IMSB.

2. Basic edit edits the input data stream from the terminal, and the message is placed on the IMSA message queue with a destination of LTISC1.
3. On output to LTISC1, ISC support in IMSA:
  - a. Strips the destination LTERM name (LTISC1) from the data stream
  - b. Builds the FMH required to send the transaction to IMSB
4. The data stream that is sent to IMSB looks like:

FMH: DPN=SCHEDULER
--------------------

FMH: DPN=ISCE, PRN=, RDPN=, RPRN=T	TRANX	Data...
------------------------------------	-------	---------

- a. DPN=ISCE because IMS ISC support supplies this value as a default if no DPN is supplied when output is to be sent to another subsystem.
- b. PRN= is not supplied and also not required for this example. TRANX is part of the user data. Only through the use of MFS can PRN= be supplied.
- c. RDPN= is not supplied and also not required for this example. Only through the use of MFS can RDPN= be supplied.
- d. RPRN=T is automatically inserted by IMSA as a default function of the message switching logic incorporated in IMS ISC.



## Accessing Existing Application Programs with ISC

5. The TRANX data stream is edited by ISC edit (ISCE) on input to IMSB, because the FMH specified DPN=ISCE. After editing, the data stream is placed on the message queues, and looks like:

```
TRANX Data...
```

6. When scheduled, an MPP retrieves TRANX from the message queues and processes the transaction. Output inserted (ISRT) by the MPP to the originating input terminal looks like:

```
Data...
```

This output can be inserted (ISRT) by the MPP to its I/O PCB (on the same parallel session) or to an alternate PCB. The I/O PCB or an alternate response PCB must be used if TRANX is a response-mode transaction. An alternate PCB could be used if TRANX were a nonresponse transaction. Alternate PCB output is sent on the same or a different parallel session, depending on the session to which the output LTERM is assigned.

7. On output from IMSB to IMSA, the FMH is built and sent with the data:

```
FMH: DPN=SCHEDULER
```

```
FMH: DPN=ISCE, PRN=T, RDPN=, RPRN Data...
```

The output is sent on the same session as that on which the input was received.

- a. DPN=ISCE is specified, because ISC support supplies this value as a default if no DPN is available from input, or through MFS when asynchronous output is to be sent to another system.
  - b. PRN=T is supplied, because the input FMH to IMSB specified RPRN=T. IMS ISC support automatically wraps an input FMH RPRN value to the output FMH PRN field.
  - c. RDPN= is not required but can be added by MFS DPM. The RPRN is not supplied, because the reply is returned on the same session as the input transaction. However, MFS can also be used to set the RPRN.
8. On input to IMSA, the reply from IMSB is edited by ISC edit (ISCE), because the FMH specified DPN=ISCE.

Because a PRN is supplied in an input FMH, ISC edit uses the PRN as the IMS destination and appends the PRN value to the input data. After ISC edit, the message is placed on the input message queues and looks like:

```
T Data...
```

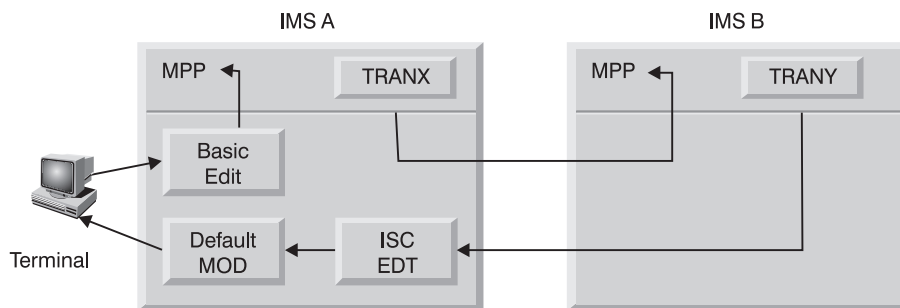
If MFS had been used to process the reply within IMSA, the insertion of the LTERM name can be suppressed. See Example 6.

9. An IMS default MOD is used to format the data for output to the terminal. The data displayed to terminal T is:

## Accessing Existing Application Programs with ISC

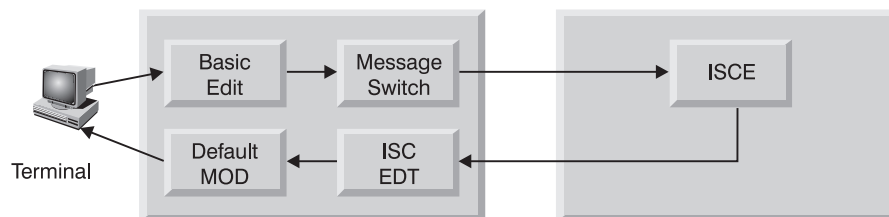
```
T|Data...
```

### Example 2. IMS-to-IMS Application Routing:



This example illustrates IMS-to-IMS routing using an application program in both IMS subsystems. The input format from the terminal operator differs from that in Example 1 in that the routing can be supplied by TRANX in IMSA rather than by the terminal operator. Processing in IMSB and on the session is the same as that described for Example 1. Also see Examples 6 and 7 later in this section to understand the interaction of MFS in the routing scenario.

**Example 3. IMS-to-Other Subsystem Message Switch Routing:** The function to be accomplished with the example that follows is the same as that in Example 1. However, the back-end subsystem is either CICS or a user-written subsystem.



The description of the activities taking place in IMS for this example parallels that in Example 1. The “other” subsystem is of interest and is described below.

1. Assume the other subsystem is CICS. Remember, from Example 1, the data stream sent from IMS looks like this:

```
FMH: DPN=ISCE,PRN=,RDPN=,RPRN=T |Data...| *
```

\* The data is formatted in a way that ISCE expects to receive it. The IMS terminal operator who is entering the transaction should understand what that format is to be.

2. CICS must have a user-written transaction defined with transaction code ISCE, because the DPN represents a transaction code to CICS. CICS, upon receiving the above data stream, invokes a transaction named ISCE, which must be specifically written for the ISC environment. This transaction examines the FMH values supplied and must use the RETRIEVE interface to obtain the user data and process TRANX.

## Accessing Existing Application Programs with ISC

Because CICS supports transaction codes having a maximum of 4 characters, ISC edit has been given the alias ISCE.

3. The output from the transaction in CICS must look like the output that IMSB sent back to IMSA in Example 1. That is:

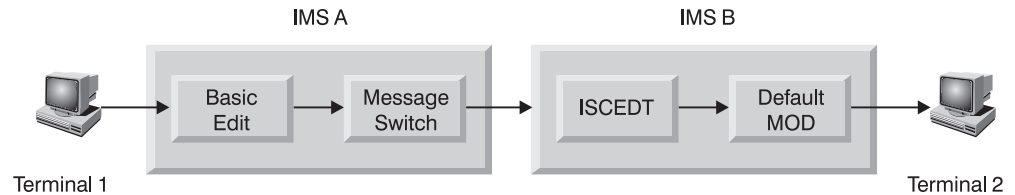
```
FMH: DPN=ISCE,PRN=T,RDPN=,RPRN | Data...
```

The CICS transaction ISCE actually gives CICS the format to use to build the output FMH (in contrast to Example 1, wherein IMS ISC support builds the FMH). In this example, the CICS transaction must supply DPN=ISCE and PRN=T to CICS to enable the proper FMH to be built.

4. When IMS receives this output data stream from CICS, all activities parallel those in Example 1.

A user can implement the functions of this example in a user-written subsystem by understanding the previous explanation of CICS functions and relating it to the functions implemented in the system.

### Example 4. IMS-Terminal-to-IMS Terminal Message Switch Routing:



The figure above shows a terminal T connected to IMSA effecting a message switch to terminal T2, which is connected to IMSB.

1. The terminal T enters the following data stream:

```
LTISC1 | T2 | Data...
```

LTISC1 is an LTERM name that IMSA has associated with an ISC session to IMSB. The terminal operator (T) must know this ISC LTERM name as well as the LTERM name of the destination terminal (T2) attached to IMSB.

2. ISC message switch support removes the LTERM name (LTISC1) before sending the balance of the message on the ISC session to IMSB. Therefore, the data stream that is sent on the ISC session looks like:

```
FMH: DPN=SCHEDULER |
FMH: DPN=ISCE,PRN=,RDPN=,RPRN=T | T2 | Data...
```

- a. DPN=ISCE is specified, because IMS ISC support supplies this value as a default if IMS does not supply a DPN when output is sent to another subsystem.
- b. PRN= is not required, because the destination, terminal T2, is part of the data stream, but could be supplied or modified by MFS.

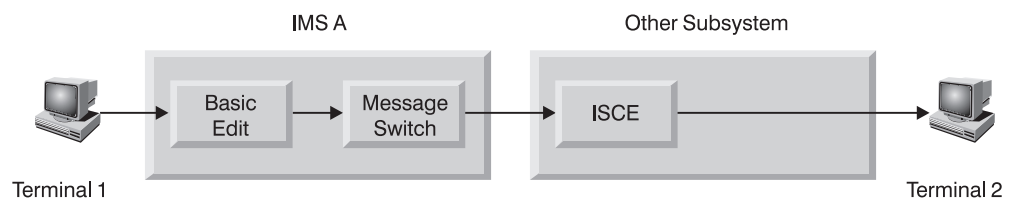
## Accessing Existing Application Programs with ISC

- c. RDPN= is not supplied and is also not required for this example. This value could have been supplied by MFS.
  - d. RPRN=T, as in Example 1, is automatically inserted by IMSA as a default function of the message switching capability of IMS ISC. Because no reply is returned, this parameter is discarded by IMSB. MFS can be used in IMSB to make this parameter available to the operator of terminal T2.
3. Before the data stream is placed on IMSB's message queues, it is edited by ISC edit. In IMSB, ISC edit examines the data (because no PRN is available in the input FMH) to determine the destination. T2 is found to be the destination and the input message is placed on the message queues with a destination T2.
  4. On output from IMSB to terminal T2, the data stream now looks like this:

```
T2|Data...
```

5. A default system MOD would be used for output if T2 were a required MFS device.

### Example 5. IMS-Terminal-to-Other Terminal Message Switch Routing:



Example 5 is similar to Example 4; however, the back-end subsystem is CICS or a user-written subsystem. The description of the activities taking place in IMS in this example parallels the description in Example 1 exactly. The other subsystem of interest is described.

1. Assume the other subsystem is CICS. Remember from Example 1 that the data stream sent from IMS looks like this:

```
FMH: DPN=ISCE,PRN=,RDPN=,RPRN=T | Data...
```

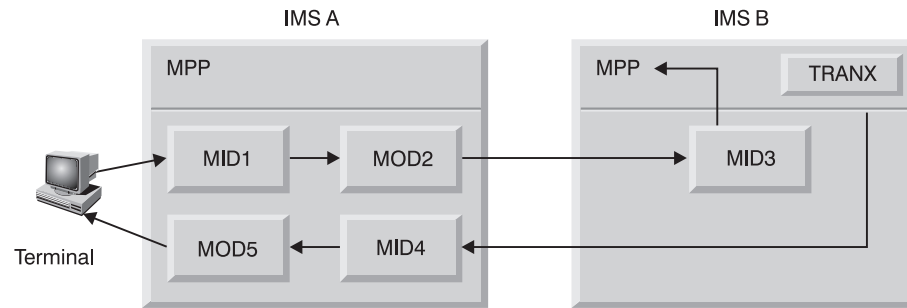
2. As in Example 3, CICS must have a transaction code defined as ISCE. The data is formatted in the way in which ISCE expects to receive it. The IMS terminal operator who is entering the transaction should understand what that format is to be. CICS, upon receiving the data stream, must invoke a transaction named ISCE, specifically written for the ISC environment. This transaction uses the data in the input FMH and obtains the input data via the RETRIEVE interface. Because the PRN field in the input FMH is not initialized by IMSA to be equal to T2, the CICS application must initiate a message switch to terminal T2 via the transient data queue or start a new transaction naming T2 as the primary resource.

In this example, the output terminal T2 must be identified in the input data stream. If IMSA had initialized the PRN field in the input FMH to T2, CICS would have attached the transaction ISCE with T2 as the primary resource, and a SEND to terminal T2 could have been made directly.

3. In order for IMSA to set the PRN field, MFS would be required.

## Accessing Existing Application Programs with ISC

### Example 6. IMS-to-IMS Message Switch Routing with MFS:



In this example, terminal T enters a transaction to be processed in the back-end IMS subsystem. The reply from the back-end subsystem is to be routed back to the original terminal that entered the transaction (T).

1. Terminal T enters its input from a formatted screen. The terminal only needs to enter data.
2. After editing by MID1, the message is placed on the message queues and looks like this:

```
|LTISC1|TRANX|Data...|
```

The values LTISC1 and TRANX were appended to the data by MFS.

3. MOD2, to be used for editing the data stream that is sent to IMSB, is chained from MID1, and is chained to another MID (MID4) within IMSA. The data stream to be sent to IMSB looks like this:

```
|FMH: DPN=SCHEDULER|
|FMH: DPN=MID3, PRN=, RDPN=MID4, RPRN=T|
|TRANX|Data...|
```

- a. DPN=MID3 can be supplied by MOD2 or from a literal in the DOF associated with MOD2.
  - b. PRN= is not needed, because TRANX is already in the data stream. It could have been supplied from MOD2 or its associated DOF.
  - c. RDPN=MID4 is specified, because MID4 is chained to MOD2.
  - d. RPRN=T is again automatically inserted by IMSA as a default function of message switching and not overridden by MFS.
  - e. LTISC1 again has been stripped from the output message by the MFS MOD.
4. Upon receipt of the data stream from IMSA, IMSB edits the data using MID3, because the DPN in the FMH specified MID3.
  5. The application receives and processes the following:

```
|TRANX|Data...|
```

6. Output from the application program is sent by IMSB to IMSA in the following format:

## Accessing Existing Application Programs with ISC

```
FMH: DPN=SCHEDULER
```

```
FMH: DPN=MID4,PRN=T,RDPN=,RPRN= Data...
```

7. DPN=MID4 and PRN=T are automatically wrapped by IMS from the RDPN and RPRN values in the original FMH.
8. On receipt of this data stream, MID4 in IMSA edits the input and places the message on the queues for final output to terminal T in the following format:

```
Data...
```

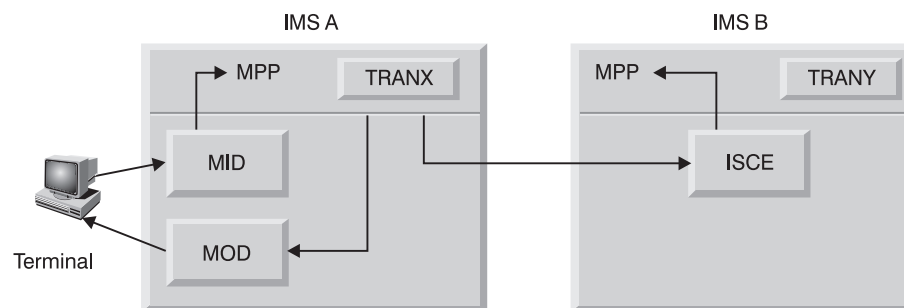
Because PRN=T is supplied, IMS uses it as the destination. Because MFS is also used to format the input reply, the PRN is not appended to the data.

9. Because MOD5 is chained to MID4, MOD5 is used to format the output to terminal T.

Several observations can be made about this example:

- MID3 in IMSB is not chained to a MOD. Therefore, the I/O PCB that the application program in IMSB sees does not contain a MODname to be used for output. If MID3 should have a chained MOD, the application ISRT call should have a blank MOD name to negate MFS editing on output.
- The application in IMSB has not changed the MOD name for output. If it does, additional MFS format design is required.
- The MPP in IMSB can be an already-existing program that also handles transactions from terminals connected to IMSB. The use of MFS can make the use of an ISC session transparent to the application.
- The original input terminal is not held in response mode.

### **Example 7. IMS-to-IMS Application Routing with MFS:**



This example can be understood in two contexts based upon an understanding of previous examples—the case shown in which ISC edit is used to edit the input in IMSB or the case in which MFS DPM-Bn is used to edit the input in IMSB. Example 7 emphasizes several points:

- Terminal T is completely handled by the front-end subsystem. Therefore, in this example, Terminal T can be held in response mode with IMSA.
- The data sent from IMSA is inserted (ISRT) via an alternate PCB and is asynchronous to the response mode transaction.
- IMSA could be connected to many IMS subsystems and, based on the input transaction, route the ISC traffic to the appropriate back-end IMSB.

## Accessing Existing Application Programs with ISC

- The MPP in IMSA can provide any required message routing information, such as selecting the appropriate back-end subsystem through an ISRT to an alternate PCB.
- MFS can be used in either or both subsystems to provide data stream formatting and routing.

### Routing Messages through MSC to an ISC LTERM

You can route messages across an MSC link to an ISC half session. Suppose you have a setup similar to Figure 46. SystemA and SystemB are IMS subsystems connected by an MSC link. SystemC is connected to SystemB with an ISC link. If you want to route input messages and output replies between SystemA and SystemC, place the ISC routing parameters in the IMS prefix portion of the message. The MSC link-processing preserves these routing parameters.



Figure 46. Routing of MSC to an ISC LTERM

## Considerations for IMS-to-IMS ISC Sessions

Consider the following information before configuring an ISC session between two IMS subsystems using static terminals.

### Coexistence of ISC and MSC VTAM within One IMS Subsystem

During IMS system definition, the TERMINAL, the MSPLINK macro statement, or an ETO logon descriptor of one IMS subsystem must be defined with the same name as that defined on the COMM (ACB name) macro of the other IMS system.

IMS system definition allows both the MSPLINK and TERMINAL macros or an ETO logon descriptor to indicate the same remote subsystem name. However, the session qualifier information (the name used on the SUBPOOL macro and the partner-id parameter on the MSPLINK macro) must be unique. This allows MSC and ISC sessions to be defined and simultaneously active between two IMS subsystems. The number of ISC parallel sessions that can be active simultaneously between two IMS subsystems is the smaller of the values defined for the SESSION= parameter of the TERMINAL macro statements of the two IMS subsystems. (The SESSION= parameter is for statically defined terminals only.) The number of MSC parallel sessions that can be active simultaneously between two IMS subsystems is the smaller of the values defined for the SESSION= parameter of the MSPLINK macros of the two IMS subsystems. The total number of MSC and ISC parallel sessions that can be simultaneously active is the sum of these two smaller numbers.

### Defining Single and Parallel Sessions

During IMS system definition, the TERMINAL macro statements or the ETO logon descriptors defining the ISC session(s) between the IMS subsystems must be defined identically as either single session or parallel sessions. However, when both subsystems are defined for parallel sessions, the number of concurrently active sessions or number of SUBPOOL macro statements need not be identical.

## Accessing Existing Application Programs with ISC

Without ETO, you can define up to 4,095 parallel sessions. With ETO, the number of ISC parallel sessions is limited primarily by the processor capacity and the amount of virtual storage above and below the line available to the control region.

### Ensuring Compatible Buffer Sizes

The VTAM output buffer size defined on the `TERMINAL` macro statement or an ETO logon descriptor of one IMS subsystem must be compatible with the “receive-any” buffer size defined on the `COMM` macro statement of the other subsystem.

### Remote Control of IMS via ISC

The IMS ISC message switch support, user-written applications, or appropriate Security Maintenance Utility (SMU) specifications allow a terminal operator within one IMS subsystem to effectively control the operation of another IMS subsystem. The IMS automated operator interface (AOI) facility can also be used to aid in the control of either IMS subsystem.

For statically defined terminals, you can authorize commands or transactions to ISC sessions within the remote IMS subsystem; all terminal operators and application programs within the local IMS subsystem that are authorized to send data on the ISC session can then access the remote subsystem. Therefore, use password security to restrict remote subsystem access to only specific authorized individuals and applications as a master terminal within the IMS subsystem.

You can provide additional security by specifying that all commands on the ISC session be issued between automated operator interfaces in both subsystems or issued to a single automated operator interface in the back-end subsystem.

A logical unit type 6.1 cannot be defined as the IMS master terminal during IMS system definition, nor can it be assigned as a master terminal via the `IMS /ASSIGN` command.

### Restriction on IMS-to-IMS Conversation Mode

Conversation mode between two IMS subsystems is not supported. This is because conversations between IMS subsystems that are connected using ISC produce unpredictable session protocols and conversational transaction input when only one half session is in conversation mode. Further, the conversation terminates if both half sessions are in conversation mode.

### Routing Transactions to the Back-End IMS

IMS ISC support allows the following to route transactions to a back-end IMS subsystem:

- Terminal operators using ISC message switches.
- Application programs using alternate PCB inserts within the front-end IMS subsystem.

In either case, the default action by the ISC support is to route a resulting transaction reply, in the form of a message switch, back to the originating terminal operator or transaction. Under these conditions, the transactions accessed within the back-end IMS subsystem should generally be defined as recoverable, because the response required for the message switch reply is also recoverable.



## Accessing Existing Application Programs with ISC

Irrecoverable transactions can be accessed, but require the use of MFS DPM-Bn to change the default destination set in the ISC message routing parameters from the LTERM associated with the originating terminal operator to an irrecoverable transaction within the front-end IMS. This avoids the protocol errors that occur when sending irrecoverable reply messages. The destination can be changed by MFS DPM either within the front-end IMS to modify or delete the default RPRN parameter sent on the transaction to the back-end IMS, or within the back-end IMS to modify or delete the wrapped PRN parameter sent on the reply to the front-end IMS. This irrecoverable transaction can then route the reply to the appropriate terminal operator by inserting to a modifiable alternate PCB.

Special support exists for front-end/back-end system utilization provided by the Front-End Switch exit routine.

**Related Reading:** For more information on the Front-End Switch exit routine, see *IMS/ESA Customization Guide* .

### Sending Messages between the Subsystems

Communication from an IMS front-end is always asynchronous. Therefore, messages are required to be sent and received with both the ATTACH and SCHEDULER FM headers. The exception to this is that system messages are sent with the ATTACH FM header only.

**Related Reading:** For more information on the system message process, see “System Message Process (SYMSG) and Related FM Headers” on page 326.

### Protocol Restrictions on IMS-to-IMS Sessions

When an ISC session is between two IMS subsystems, certain types of protocols are not sent between the subsystems. This section provides these protocols. If you are designing IMS-to-IMS sessions, you can skip the information on these protocols and headers provided in the next chapter.

- The data flow control commands BID, RTR, and RSHUT are not sent between the two IMS subsystems.
- The reset-attached process (RAP) FM header is not sent between IMS subsystems.
- The queue model (QMODEL) headers are not sent between IMS subsystems. This precludes IMS output demand-paging and all associated input QMODEL paging requests. Also, the ATTDQN parameter on the ATTACH FM header and the SCDDQN parameter on the SCHEDULER FM header are not supported in IMS-to-IMS sessions.
- The following error codes are not sent between two IMS subsystems:
  - LUSTATUS commit and function abort X'0864' and X'0866'
  - All sender ERP sense codes except X'0813' and X'0846'
  - Selective receiver ERP sense codes, except as listed under “Selective Receiver ERP” on page 312.

---

## Statically Defining an ISC Node to IMS

A subsystem that is statically defined at system definition to IMS as an ISC node appears to IMS to be a terminal and to an IMS application program to be one or more logical terminals (LTERMs). Thus, an ISC node is defined using those system definition macros applicable to defining other VTAM terminals.

## Statically Defining an ISC Node to IMS

The list below shows the macro statements that must be coded to define an ISC node:

Macro Statement	Use
<b>COMM</b>	Names the IMS subsystem to VTAM; names the ISC edit process and defines the receive-any (RECANY) buffer size.
<b>TYPE</b>	Identifies the unit type as an ISC node (UNITYPE=LUTYPE6).
<b>TERMINAL</b>	Identifies the other ISC node to IMS. Also provides sizes for the output and Fast Path buffers, limits for the input segment size, and processing options. Defines the number of sessions possible between two ISC nodes. Describes component characteristics.
<b>VTAMPOOL</b>	Begins the definition of the ISC LTERM users (subpools) used with parallel sessions.
<b>SUBPOOL</b>	Defines a set of LTERMs within the VTAMPOOL.
<b>NAME</b>	Names the LTERMs associated with a given terminal for a single session or a given user (subpool) for parallel sessions.

**Related Reading:** For more information on these system definition macros, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

This section more fully describes the macros that are key to defining an ISC session between two systems and some of the implications of selecting certain system definition options instead of others.

Figure 47 shows two IMS systems, each of which is defined to the other as an ISC node. The following paragraphs detail the way in which these systems would be defined during IMS system definition. For ease of understanding, this description focuses on defining the IMS system SFIMS to the IMS system NYIMS. A similar system definition would be required to define system NYIMS to subsystem SFIMS. Only those parameters of interest on the COMM, TERMINAL, and NAME macro statements are described.

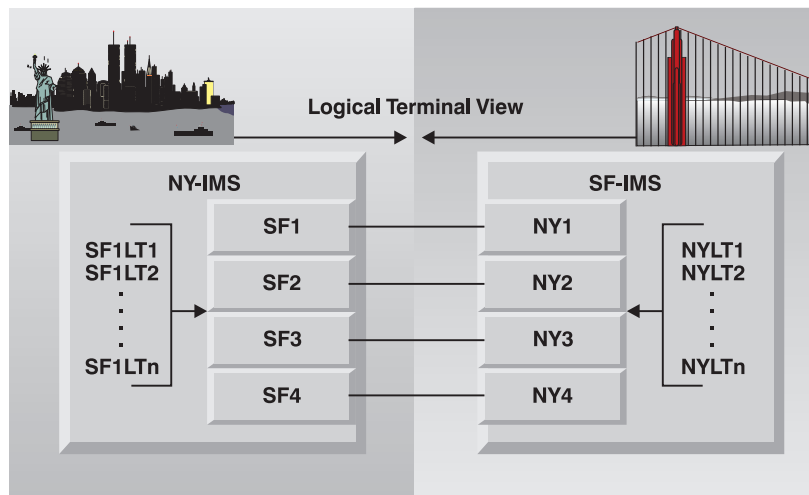


Figure 47. Communication between Two IMS Subsystems

## Statically Defining an ISC Node to IMS

The parameter `APPLID=NYIMS` on the `NYIMS` system definition `COMM` macro makes that subsystem known by the node name `NYIMS` to `VTAM`, to itself, and to any other subsystems that wish to access it. A similar system definition must occur for the San Francisco subsystem `SFIMS`.

`NAME=nodename` on the `TERMINAL` macro statement of the `NYIMS` system definition is coded to show the node name of the `SFIMS` system (`NAME=SFIMS`). If `SFIMS` is an XRF complex, the `nodename` should be the `USERVAR` associated with the `SFIMS` system.

The `SESSION=` keyword is related to parallel sessions rather than to a single session. (The `SESSION=` keyword applies to statically defined terminals only. It does not apply to the `ETO` environment.) `SESSION=` specifies the maximum number of parallel sessions that are allowed from `NYIMS` to `SFIMS`. It is possible to code one “parallel” session. A single session and one parallel session are not the same.

**Related Reading:** For more information on single and parallel sessions, see “`LTERM` Users (Subpools) and Components” on page 240.

By coding `SESSION=` as greater than one, the `NYIMS` might maintain multiple concurrent sessions with `SFIMS`, up to the number specified by the `SESSION=` parameter. In Figure 47 on page 258, four paths are drawn between the two `IMS` systems. Each path represents a parallel session. To allow four parallel sessions in the `NYIMS`, code `SESSION=4`.

A major advantage of parallel session support is the ability to dynamically allocate `LTERMs` to that session based on the `IMS /OPNDST` command parameters.

**Related Reading:** For more information on dynamic `LTERM` allocation, see “`LTERM` Users (Subpools) and Components” on page 240.

Each parallel session is required to be identified to `NYIMS` using the `SUBPOOL` macro statement. In the example, to allow four concurrently active parallel sessions, four users (subpools) must be defined as follows in the `NYIMS` system definition:

```
VTAMPOOL
  SUBPOOL  NAME=SF1
  SUBPOOL  NAME=SF2
  SUBPOOL  NAME=SF3
  SUBPOOL  NAME=SF4
```

The `VTAMPOOL` macro statement begins the definition of the `ISC` subpools and must be coded if parallel sessions are to be used. The `VTAMPOOL` macro statement has no operands.

In Figure 47 on page 258 `SFLT1`, `SFLT2`, `SFLT3`, . . . `SFLTn` represent `LTERM` names that have been defined to the first `SUBPOOL`, `SF1`. The definition of `LTERMs` within a given subpool is accomplished by coding the `NAME` macro. Defining multiple `LTERMs` to a subpool provides several advantages. The first is that an `LTERM` or `LTERMs` defined to a given subpool can be reassigned (using the `/ASSIGN` command) to any other subpool. This capability gives flexibility in the operation of an `ISC` network.

The second advantage is with the `COMPTn=` keyword on the `TERMINAL` macro statement. This keyword can be used to define or assign certain characteristics to a given `LTERM`. `COMPT=` and `ICOMPT=` are the keywords on the `NAME` macro statement and are used to associate a component (`COMPTn`) with an `LTERM`.

## Statically Defining an ISC Node to IMS

The definition of SFIMS to NYIMS now looks like Figure 48:

```
TYPE          UNITYPE=LUTYPE6
TERMINAL     NAME=SFIMS
            SESSION=4

VTAMPOOL
  SUBPOOL NAME=SF1
    NAME SF1LT1
    NAME SF1LT2
  :
  :
  NAME SF1LTN
  SUBPOOL NAME=SF2
  :
  :
  SUBPOOL NAME=SF3
  :
  :
  SUBPOOL NAME=SF4
  :
  :
```

Figure 48. The Definition for SFIMS and NYIMS

---

## Choosing Parameters: System Design Considerations

This section further describes the parameters of the COMM, TERMINAL, NAME, and SUBPOOL macro statements.

### COMM Macro Statement

The key parameters on the COMM macro statement are:

APPLID=NYIMS

RECANY=XXX

EDTNAME =                    *ISCEDT*  
                             user-specified name

The APPLID parameter defines the VTAM ACB name for IMS. The name specified for APPLID= must be the same as the ACBNAME= parameter of the VTAM APPL definition statement. If the ACBNAME= parameter is not specified, the APPLID= name must be the same as the name of the APPL definition statement that is used when defining IMS to VTAM. If the other subsystem is also an IMS, the name specified for APPLID= must be the same as that supplied on the NAME= keyword parameter of that IMS TERMINAL macro statement.

If you choose not to define the APPLID parameter on the COMM macro statement and instead use the JOBSTEP name of the EXEC statement when initiating the IMS control region, VTAM attempts to match that name with either the ACBNAME= parameter or the name of the VTAM APPL definition statement, as noted above.

When specifying the receive-any (RECANY) buffer size on the COMM macro, remember that a 28-byte overhead applies to the size of the receive buffers for ISC sessions. Therefore, the RECANY buffer size specified on the NYIMS system definition

## Choosing Parameters: System Design Considerations

must be at least 28 bytes larger than the size specified for OUTBUF= on the TERMINAL macro of the SFIMS system definition for NYIMS. When defining SFIMS, the same considerations apply.

Other factors affect the size of the RECANY buffers. Because the specified RECANY buffer size applies to all VTAM devices, the size specified for RECANY might be larger than the size required for ISC due to the needs of other VTAM devices. Also, the FM header size must be added to the maximum data record size for both systems to determine the maximum ISC RECANY size.

The EDTNAME parameter specifies the alias that can be used for ISCEDT in your IMS system. In this system definition, the default name, ISCEDT, is used, making the specification of the EDTNAME= parameter unnecessary.

### Related Reading:

- For information on the interaction of the receive-any buffer size and the output buffer size, see “Specifying the OUTBUF Parameter” on page 262.
- For more information on setting buffer sizes, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

## NAME Macro

The key system definition options for defining an ISC session on the NAME macro are those in Figure 49.

```
      1
      2
COMPT = 3
      4

OUTPUT=terminal label

EDIT=ULC/UC

      1
      2
ICOMPT = 3
      4
```

Figure 49. System Definition Options

COMPT= and ICOMPT= specify the output and input components respectively to be associated with the LTERM being defined. The system definition example uses components defined as COMPT=1 and COMPT=2.

OUTPUT= should not be used on an ISC session.

**Recommendation:** If you do not desire translation of ISC output to the session, specify ULC. If session output can be uppercase or lowercase data, or binary data, use ULC.

## SUBPOOL Macro

The SUBPOOL macro identifies ISC subpools to a system.

```
NAME=subpool name
MSGDEL=SYSINFO/NOIOPCB
```

## Choosing Parameters: System Design Considerations

The NAME parameter identifies the ISC subpool.

In this example, the MSGDEL specification is the default, SYSINFO. When MSGDEL is specified on the SUBPOOL macro, it must match the MSGDEL specification on the TERMINAL macro.

**Related Reading:** For more information on specifying MSGDEL, see “LTERM Users (Subpools) and Components” on page 240.

## TERMINAL Macro

The key system definition options for defining an ISC session on the TERMINAL macro are:

OUTBUF=XXX

MSGDEL=SYSINFO/NONIOPCB

COMPT1		MULT1		VLVB		IGNORE
COMPT2	=	MULT2		DPM1		1
COMPT3		SINGLE1	,	:	,	2
COMPT4		SINGLE2		DPM15		:
						10

### Specifying the OUTBUF Parameter

IMS does not negotiate buffer sizes when initiating an ISC session, nor does IMS support RU truncation. When sending a negotiated bind, IMS sets its send (OUTBUF=) and receive (RECANY=) sizes in the bind.

When receiving the bind reply, IMS verifies that:

- Its send (OUTBUF=) size has not been reduced by the other subsystem.
- Its receive (RECANY=) size is not exceeded by the other subsystem's send (OUTBUF=) size.

When receiving a negotiated bind, IMS:

- Verifies that the send (OUTBUF=) size of the other subsystem does not exceed IMS's receive (RECANY=) size.
- Inserts the send (OUTBUF=) size into the bind reply.

The specified OUTBUF size must be large enough to include the largest output segment size plus the overhead for message headers. When defining the SFIMS system, the same considerations apply.

**Related Reading:** For more information on the RECANY buffer size, see “COMM Macro Statement” on page 260.

### Specifying the MSGDEL Parameter

A description of the ramifications of choosing SYSINFO or NONIOPCB is found in “LTERM Users (Subpools) and Components” on page 240. In the example in that section, the default for SYSINFO is used for both the TERMINAL and the SUBPOOL macros.

### Specifying the COMPTn Parameter

Up to four components can be defined for a given ISC node on the TERMINAL macro. The parameters that can be chosen are shown above. All parameters are optional. “COMPTn=” allows flexibility in assigning characteristics to LTERMs. In the

## Choosing Parameters: System Design Considerations

system definition example, component 1 is specified as having the characteristics SINGLE1, DPM-B1, and IGNORE. Component 2 is specified as having the characteristics SINGLE2, DPM-B2, and IGNORE.

### Specifying the SINGLE or MULT Parameter

The characteristics of SINGLE1, SINGLE2, MULT1, and MULT2 are described in “Determining Output Protocols” on page 243.

### Specifying the VLVB or DPM-B1...DPM-B15 Parameter

Selecting VLVB or DPM-Bn determines whether the component can use the Distributed Presentation Management feature of MFS. Although MFS is specified for a component by specifying DPM-Bn, its use is optional, on a message-by-message basis. DPM-Bn is used for this system definition example.

The selection of VLVB precludes the use of MFS-DPM for a component and indicates that variable-length, variable-blocked format is to be used instead of MFS for both input and output.

**Related Reading:** For more information on VLVB, see *IMS/ESA Application Programming: Transaction Manager*.

### Specifying the IGNORE or 1, 2,...10 Parameter

This parameter is used to specify a user-defined feature code for MFS DPM-Bn. The designated feature is used to select an MFS format (DPM-B1...DPM-B15) that contains the matching feature specification. IGNORE is used to specify that an MFS format (DPM-Bn) with the FEAT=IGNORE of the DEV statement is to be selected. IGNORE has been used for this system definition example.

---

## System Definition Summary

Referring back to our example in Figure 47 on page 258, SFIMS can be defined to NYIMS as shown below.

**Related Reading:** For more information on the parameters not described above, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

```
TYPE          UNITYPE=LUTYPE6
TERMINAL NAME=SFIMS
  COMPT1=(SINGLE1, DPM-B1, IGNORE)
  COMPT2=(SINGLE2, DPM-B2, IGNORE)
  SESSION=4
VTAMPOOL
SUBPOOL NAME=SF1
  NAME SF1LT1, COMPT=1
SUBPOOL NAME=SF2
  NAME SF2LT1, COMPT=2
SUBPOOL NAME=SF3
  NAME SF3LT1, COMPT=2
SUBPOOL NAME=SF4
  NAME SF4LT1, COMPT=2
```

This definition does not correspond exactly to Figure 47 on page 258. In that figure, the SUBPOOL SF1 contains multiple LTERMs (to indicate that multiple LTERMs can be assigned to a SUBPOOL). Based on SINGLE1, SINGLE2, MULT1, and MULT2, the definition can now be simplified to show just one LTERM being defined per SUBPOOL.

The first SUBPOOL, SF1, has LTERM SF1LT1 defined with COMPT=SINGLE1. For asynchronous output from NYIMS to SFIMS, LTERM SF1LT1 is used as the destination LTERM for all:

## System Definition Summary

- Alternate PCB output from application programs in NYIMS that generates asynchronous transactions in SFIMS
- Message switches originated by a terminal connected to NYIMS
- Replies to nonresponse transactions received by NYIMS from SFIMS

The characteristics of SINGLE1 (BB/message/EB) should allow one parallel session to handle all asynchronous input and output between NYIMS and SFIMS.

Three SUBPOOLS (SF2, SF3, SF4) have been defined with single LTERMs, each defined with a COMPT defined as SINGLE2. These three SUBPOOLS (sessions) can be used to send response mode transactions to SFIMS. Because of the nature of response mode transactions (session tied up from transaction origination until receipt of reply), three parallel sessions are defined to allow the user to minimize response mode bottlenecks between NYIMS and SFIMS.

In most IMS ISC definitions, the assignment of one LTERM to each SUBPOOL is sufficient to handle the communications traffic between the two nodes.

Not shown, but certainly to be considered, is the possibility of defining a fifth SUBPOOL within NYIMS to handle incoming SINGLE1 traffic from SFIMS. By design then, NYIMS could receive all SINGLE1 input from SFIMS on the session defined by the fifth SUBPOOL and send all SINGLE1 output on the session defined by the first SUBPOOL (SF1). This plan would result in minimal contention for asynchronous output between the two systems.



---

## Chapter 14. ISC Protocol Guide and Reference

This chapter contains the protocols IMS uses to control sessions, data flow, and message routing. It has the specific protocol information you need to send and receive data with an ISC link.

### **In this Chapter:**

- “Operating the Network” on page 266
- “Controlling the Session (Session Control Protocols)” on page 267
- “Resynchronizing Sessions” on page 270
- “Completing Session Initiation” on page 278
- “Running the Session” on page 279
- “Terminating the Session” on page 279
- “Using STSN to Resynchronize Sessions” on page 280
- “STSN Command Format” on page 283
- “Controlling Data Flow (DFC Protocols)” on page 285
- “Normal Conversation Termination Extension with ISC” on page 287
- “Keeping Half Sessions Synchronized” on page 287
- “Data Flow Control Protocol Reference” on page 295
- “BID Protocol” on page 295
- “Bracket and Half-Duplex Protocol” on page 296
- “CANCEL Protocol” on page 304
- “Chaining Protocol” on page 304
- “CHASE Protocol” on page 305
- “ERP Purging” on page 305
- “LUSTATUS Protocol” on page 308
- “Paged Messages ERP” on page 311
- “Ready-to-Receive Protocol” on page 312
- “RSHUT Protocol” on page 312
- “Selective Receiver ERP” on page 312
- “Sender ERP” on page 316
- “Sense Codes that IMS Receives” on page 319
- “Sense Codes that IMS Sends” on page 319
- “SIGNAL Protocol” on page 320
- “Symmetrical Session Shutdown for LU 6.1 (SBI and BIS)” on page 320
- “Function Management Headers” on page 321
- “Using FM Headers to Invoke ISC Edit” on page 323
- “Initiating a Process: ATTACH FM Header” on page 323
- “Error Recovery Procedure FM Header” on page 324
- “Resetting the Active Process: RAP FM Header” on page 324
- “Requesting Asynchronous Transaction Processing: SCHEDULER FM Header” on page 325
- “System Message Process (SYSMSG) and Related FM Headers” on page 326

### Operating the Network

Because ISC permits multiple sessions between logical units, the balance of this explanation of ISC differentiates between the terms logical unit, session, and half session:

#### **Definitions:**

- The term *logical unit* is used when referring to characteristics common to all sessions between two session partners or when it is not necessary to differentiate between individual sessions.
- The term *session* is used when describing a characteristic unique to a given connection (session instance) between logical units.
- A *half session* describes characteristics unique to one of the session partners.

### Making IMS Ready

Use the IMS /START command with the DC keyword to make IMS ready to receive VTAM logon or BIND SCIP (session initialization) requests. The DC keyword initiates IMS data communications processing, opens the VTAM access method control block (ACB) if it is not already open, and enables the IMS VTAM logon exit. Any logon requests received by VTAM before the IMS /START DC command is issued but after the ACB has been opened are queued in VTAM until the /START DC command is completed. If VTAM is active when IMS is initialized, the IMS VTAM ACB is opened; otherwise, the ACB is opened by the /START DC command.

The /START DC command also tells VTAM to pass any queued VTAM logon requests to IMS.

### Bringing Up an IMS Network

Before any sessions can be established, VTAM and NCP must be active. In addition, all logical units must be online (activated by the VARY command).

**Related Reading:** For information on establishing individual sessions, see “Controlling the Session (Session Control Protocols)” on page 267.

You do not need to perform message resynchronization after a cold start.

**Related Reading:** For information on cold starting a network, see *IMS/ESA Operations Guide*.

### Shutting Down an IMS Network

Use the IMS /CHECKPOINT command to terminate the network and shut down IMS. The format of the /CHECKPOINT command determines whether the network termination occurs immediately or waits for processing to complete:

- /CHECKPOINT FREEZE|DUMPQ|PURGE immediately terminates the session for all logical units, as follows:

**FREEZE**           Immediately after current input/output message

**DUMPQ**            After control blocks are past checkpoints

**PURGE**            After all queues are empty

- /CHECKPOINT FREEZE|DUMPQ|PURGE QUIESCE allows all network nodes to complete normal processing before shutting down.

The IMS command `/STOP DC` also shuts down an IMS network at completion of processing.

Also, certain VTAM commands, such as `VTAM HALT NET`, shut down the network.

**Related Reading:** For more information on shutting down an IMS network, see *IMS/ESA Operations Guide*.

---

## Controlling the Session (Session Control Protocols)

Session initiation includes initiating a session instance, binding the session, and, if necessary, ensuring that the half sessions are in sync. When both half sessions have agreed to the bind and are in sync, normal traffic flow is initiated using the VTAM start data traffic (SDT) command.

### Initiating a Session

A session must be established before data can be transmitted between another logical unit type 6.1 and IMS. If message resynchronization is required, it is performed after the bind.

IMS can be requested to initiate a session in one of the following ways:

- An ISC logical unit requests session initiation by sending the “initiate-self” command. VTAM verifies the command and passes the request to IMS.
- An ISC logical unit sends `BIND` to initiate a session with IMS.
- The MVS VTAM network operator requests session initiation on behalf of the logical unit by using the `MVS VARY` command with the `LOGON` option. VTAM processes the request and passes it to IMS. The `VARY` command cannot be used to initiate parallel sessions.
- VTAM passes to IMS a logon or `BIND SCIP` request for each logical unit defined to VTAM as belonging to IMS. (This method cannot be used to initiate parallel sessions.)
- An authorized IMS terminal operator requests session initiation for an ISC logical unit by entering the `IMS /OPNDST` command.

IMS can initiate (send `BIND`) or accept (receive `BIND`) a session if all the following conditions are met:

- The IMS master terminal operator has issued a `/START DC` command.
- The logical unit name is known by IMS.
- The logical unit is not stopped (`/STOP` command) within IMS.
- The logical unit has not reached the maximum allowable sessions defined to IMS. (This applies to statically defined terminals only.)
- If the session is to be parallel, the `CINIT` or `BIND` session qualifier field must contain a valid `LTERM` subpool name that defines the message queue set to be used. A valid subpool name is one that is not stopped or allocated (except during session restart). The session qualifier fields must not be supplied when requesting session initiation with a logical unit defined to IMS as single session.
- For statically defined terminals, the `MSGDEL` option for the `LTERM` subpool as specified on the `SUBPOOL` macro does not conflict with the `MSGDEL` option specified for the session on the `TERMINAL` macro.

**Related Reading:** For more information on specifying the appropriate parameters, see the descriptions of these macros in “Statically Defining an ISC Node to IMS” on page 257.

## Controlling the Session (Session Control Protocols)

- No invalid or conflicting parameters are indicated on either a CINIT, BIND, or negotiable BIND reply.

**Related Reading:** For information on the ISC bind parameters, see “Appendix B. Bind Parameters for SLU P and LU 6.1” on page 471.

## Establishing Connection with the XRF Complex

If one session partner is to be an XRF complex, the following actions must be taken at ISC session initiation:

- If the non-XRF half session initiates the session with IMS, the non-XRF half session should:

1. Identify the APPLID of the currently active IMS system by issuing the following macro:

```
INQUIRE OPTCD=USERVAR,AREA=area,AREALN=8
```

If the VTAM network is Release 3.2 or later and USERVAR Management Enhancement is installed, you do not need to issue the INQUIRE macro.

With this request, the AREA parameter specifies the address of the area with the USERVAR name for the active XRF system. The USERVAR is then replaced with the APPLID of the currently active IMS system.

2. Issue a session-initiation request by specifying the APPLID of the active IMS system.

If the VTAM network is Release 3.2 or later and USERVAR Management Enhancement is installed, issue a session-initiation request by specifying the USERVAR of the XRF system.

- If the IMS system in the XRF complex initiates the session, the IMS system:

1. Issues the session-initiation request, adding the USERVAR assigned to the BIND in the user data field.

**Related Reading:** For the format of the BIND parameter, see Note 6 in Table 48 on page 477.

2. Validates the session-initiation request by referencing the USERVAR in the user data field located at the end of the request. Initiation is not apparent to the user.

If the active IMS system fails, the alternate system re-establishes the session during takeover. Until the ISC session between the old active IMS and the remote system terminates completely, session initiation fails. The new active system retries session initiation every 30 seconds (for a maximum of 20 attempts) until the session initiates successfully.

## Binding the Session

In an ISC session, IMS can assume either the primary half-session role (send the BIND) or the secondary half-session role (receive the BIND). However, when session restart and recovery are required, session polarity must be maintained.

**Definition:** *Session polarity* means that the same session role (primary versus secondary) is in effect at the point of failure and is reestablished by the session initiation request. Otherwise, the request is rejected. When the session is initiated using an IMS /OPNDST command and session restart and recovery are not required, IMS requests to be the primary half session.

## Controlling the Session (Session Control Protocols)

### Negotiable versus Nonnegotiable BIND

As the primary half session, IMS sends either a negotiable or nonnegotiable BIND, depending on parameters in the VTAM mode table. The mode table entry is indicated on the VTAM CINIT or IMS /OPNDST commands, or defined on the TERMINAL macro during IMS system definition.

#### **Definitions:**

- When sending a *negotiable BIND*, IMS sets the bind parameters, as indicated in “Appendix B. Bind Parameters for SLU P and LU 6.1” on page 471. Because the secondary half session can change some parameters, all the parameters are checked for validity by IMS when a negotiable BIND response is received.
- Prior to sending a *nonnegotiable BIND*, IMS checks all parameters from the mode table entry for validity. The session is terminated if IMS finds any incompatible parameters.

**Related Reading:** For information on the parameters that can be set in the mode table entries or changed on a negotiable BIND response, see “Appendix B. Bind Parameters for SLU P and LU 6.1” on page 471.

As the secondary half session, IMS receives both the negotiable and the nonnegotiable form of BIND. When receiving a negotiable BIND, IMS checks the bind parameters, except the secondary network addressable unit (NAU) protocol field, for validity. IMS then sets the secondary NAU protocol field for the negotiable BIND response to the values indicated in “Appendix B. Bind Parameters for SLU P and LU 6.1” on page 471.

When receiving a nonnegotiable BIND, IMS checks the bind parameters, except for “STSN required” and “BIS sent,” for validity prior to accepting the BIND. IMS must then operate under the secondary NAU protocol definition provided in the bind.

### Binding Single or Parallel Sessions

Requirements for binding a session differ for single and parallel sessions. To bind a single session, either negotiable or nonnegotiable BIND is sent. The ISC logical unit with which IMS is communicating must have been defined with a static set of LTERMs during IMS system definition. IMS ignores the session qualifier pair (SQP) field on both CINIT and BIND.

To bind a parallel session, the CINIT and BIND parameters must include a SQP field to identify the specific parallel half-session instance between IMS and a logical unit. (All sessions created dynamically using the ETO feature are parallel sessions.) This field in the bind parameters contains both the primary and the secondary session qualifiers. The half-session name of the ISC node communicating with IMS consists of the logical unit name concatenated with its associated session qualifier. The IMS half-session name is the IMS ACB name concatenated with the session qualifier associated with IMS. The IMS session qualifier is the subpool name. Both half-session names are saved across session and IMS subsystem failures and are used to allocate, or validity-check when warm starting, the LTERM subpool to be used for the session.

### Resolving a Bind Race

A race occurs when IMS and another logical unit simultaneously send BIND requests to each other and the two half-session names are mirror images. That is, the primary logical unit name and session qualifier concatenated with the secondary logical unit name and session qualifier of one side is equal to the primary logical

## Controlling the Session (Session Control Protocols)

unit name and session qualifier concatenated with the secondary logical unit name and session qualifier of the other side. The logical unit that wins and becomes primary is the one whose name (the primary logical unit name taken from each half-session name) is at the top of the standard collating sequence. This test prevents both sides from rejecting each other, resulting in no session.

### After a Successful Session Bind

Following a successful session bind, both half sessions must perform a resynchronization process. Message resynchronization is not performed when IMS is cold started or when a previous session instance between IMS and another logical unit has been normally shut down using the Stop Bracket Initiation/Bracket Initiation Stopped (SBI/BIS) procedure.

#### Related Reading:

- For more information on the resynchronization process, see “Resynchronizing Sessions”.
- For more information on session shutdown, see “Terminating the Session” on page 279 and “Symmetrical Session Shutdown for LU 6.1 (SBI and BIS)” on page 320.

---

## Resynchronizing Sessions

To maintain the integrity of recoverable resources, messages, and queues in IMS across both subsystem and session failures, both half sessions must maintain the session information required for the resynchronization process. Message integrity cannot be maintained without user intervention when either or both subsystems incur an error or when a user restart procedure destroys this information.

**Related Reading:** For more information on resynchronization process session requirements, see “Sync Point and Response Requirements” on page 289.

Resynchronization is required when it is possible for a recoverable work unit to be in-doubt on a flow (primary-to-secondary or secondary-to-primary).

**Definition:** A *work unit* includes all transmissions between sync points within a bracket, as illustrated in Figure 50 on page 271. A sync point might have been requested by one or both half sessions without having been acknowledged. These conditions can be caused by a subsystem or session failure or an abnormal completion of a shutdown sequence.

Performing message resynchronization is unnecessary following a normal shutdown sequence (unless nonnegotiated BIND was sent), because both half sessions can come to a controlled, mutual understanding that no additional normal message traffic or sync-point requests are to occur prior to session termination. Message resynchronization is always required following nonnegotiable BIND to allow error conditions detected by the secondary half session to be communicated to the primary half session.

The half-session pairs resynchronize with the VTAM BIND, set-and-test-sequence-numbers (STSN), and start data traffic (SDT) commands. The STSN command allows both half sessions to reestablish sync-point information (session sequence numbers), which is being maintained by both half sessions.

## Resynchronizing Sessions

When message resynchronization is necessary, it must be completed successfully before either half session can resume normal data transmission.

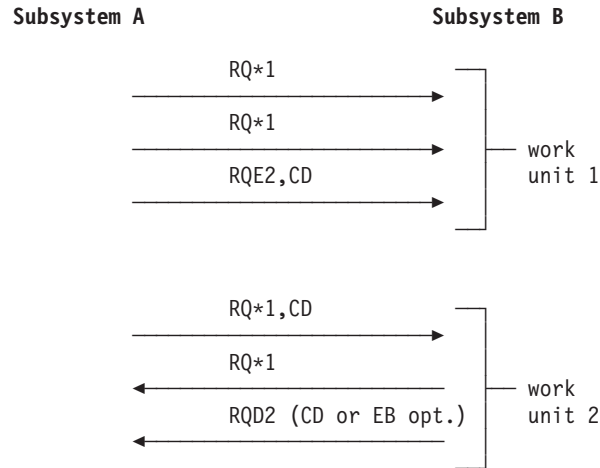


Figure 50. Work Unit Examples

In work unit 1 of Figure 50, a reply to the exception response request and CD creates an implied sync point. In work unit 2, sending the DR2 response to the RQD2 creates a sync point.

**Related Reading:** For more information on half-session synchronization, see “Keeping Half Sessions Synchronized” on page 287.

## Designing Restart Resynchronization Procedures

Resynchronization might or might not be required when IMS is restarted and when a session is established or reestablished. This section addresses considerations for resynchronization at IMS restart and at session restart.

IMS restart provides for message recovery during:

- A normal IMS subsystem restart procedure, by restoring the IMS message queues and session restart information to the last or specified checkpoint
- An emergency IMS subsystem restart procedure by restoring the IMS message queues and session restart information to their states just prior to failure

If the failure occurred prior to a normal shutdown sequence between IMS and another logical unit, ensure that the session is restarted using the same half-session pairs (half-session names) that were in session at the time of failure. Further, when the session is resumed, the half-session pairs must maintain their previous relationship—that is, the former primary half session must again be primary, and the former secondary half session must be the current secondary.

### Maintaining Sequence Numbers

To allow session recovery and message resynchronization across session and subsystem failures, both half sessions must maintain a checkpoint of the following:

- Three sequence numbers that are produced when the session was last active

## Resynchronizing Sessions

The three sequence numbers are the potential (pending) and committed sequence numbers for the flow sent by the half session and the last committed sequence number for the flow received by the half session.

- An indicator produced when a unilateral decision was made by a half session to back out or commit a work unit that was left pending during the session outage
- An indication of the direction of the decision

Sequence number mismatches and incorrect decisions to commit or back out a recoverable work unit are detected by comparing the sequence numbers sent or received on the VTAM set-and-test-sequence-number (STSN) command with the checkpointed resynchronization information. Each half session can detect invalid sequence number mismatches on their outbound flow and reject the resynchronization request. Each half session can also agree or disagree to unilateral decisions made by the other half session to commit or back out a work unit. This agreement or disagreement is based on the detection of a sequence number mismatch on the inbound flow and involves a second STSN when an incorrect decision is made on the STSN receiver's inbound flow. The STSN command receiver does this before responding to the second STSN command (which is sent in response to the receipt of TEST NEGATIVE on the first STSN). This second STSN informs the STSN receiver that a wrong decision had been made on its inbound flow.

## Recovering Sessions with Cold Start

Invalid sequence number mismatches and mismatches where no unilateral decisions have been made often indicate a subsystem restart from an incorrect log or incorrect checkpoint on the log. This situation can require forcing the recovery of the ISC session by cold starting one or both half sessions.<sup>30</sup> IMS allows authorized terminal operators to change IMS ISC session state from recoverable to cold start when necessary by using the IMS /ASSIGN (subpool to VTAMPOOL) command before attempting to initiate a session. When a session is changed to a cold-start state, the subpool associated with that session is made available for allocation to any ISC cold-startable parallel session with any node.

## Controlling Unilateral Decisions about Pending Work Units

To prevent resources from accidentally getting out of synchronization between subsystems, you can specify whether to continue session resynchronization after a session outage, where the other half session has made a unilateral decision to commit or back out a pending unit of work. IMS allows system definition to allow resynchronization without regard to inbound sequence number mismatches (OPTIONS=FORCESS on the TERMINAL macro statement or an ETO user descriptor), or to only allow resynchronization when the inbound sequence numbers agree (OPTIONS=SYNCSESS). A keyword on the IMS /CHANGE command allows an authorized terminal operator to override the system definition specification for a single attempt to initiate a session. The effects of the CHANGE command are reset to the original system definition specification after one session initiation attempt. IMS does allow unilateral decisions to back out during session outages. However, using the IMS /DEQUEUE command for a session (terminal, and optionally, the subpool parameter) or LTERM during a session outage where an output-message sync point is pending is considered to be a unilateral decision by an authorized terminal operator to commit the pending output.

---

30. Session recovery where one or both half sessions are cold started is not considered a major error because, by definition, the cold starting half session has no information from the previous active session with which to negotiate (agree or disagree) recovery or resynchronization.



## Resynchronizing Sessions

A pending output sync-point response can be determined by using the IMS /DISPLAY command. When two IMS subsystems are connected via an ISC session, the FORCESS option must be in effect on the opposite IMS subsystem from one where a pending output message was dequeued (committed) using a /DEQUEUE command, or session resynchronization fails with message DFS2065. Using FORCESS rather than SYNCSESS has no other effect between two IMS subsystems.

### Recovering from In-Bracket Failures

When a session has failed while in-brackets, it might be necessary to restart the failed bracket when the session is restarted. The bracket state manager and the half-duplex reset flags in the bind can be set in either the BIND request or the negotiable bind response by either half session. These flags are set to in-brackets/SEND or RECEIVE to indicate the possibility that the process previously attached must be restarted.

### Response or Conversational Output Available at Restart

If, during session initiation, IMS has response mode or conversational output available to be sent or is in a sync-point response pending state, the bracket state manager reset and half-duplex flags within the IMS portion of the bind or bind response are set to in-brackets/SEND by IMS. If IMS is in conversation mode but has no conversational output available or pending, these same bind or bind reply flags are set to in-brackets/RECEIVE, because input is required to continue the conversation. IMS does not accept a nonnegotiated BIND or negotiated BIND response indicating IMS to be in-brackets/RECEIVE when conversation or response mode output is available. Unless a sync point is pending from a previous session, the session is resumed and the STSN that is sent must receive a TEST POSITIVE reply to the SET AND TEST option in order for processing to continue.

When IMS is in conversation or response mode and receives a nonnegotiated BIND or a response to a negotiated BIND indicating between-brackets, IMS attempts to terminate the conversation or response mode. This termination is just like that which occurs if end-bracket is received on an FMH7 or LUSTATUS while IMS is in conversation or response mode during normal data flow active state (after SDT).

Conversation and response mode termination is only assured when the output reply message is available to be sent or the half session is waiting for the next conversational input. Therefore, IMS restricts attempts to terminate conversation or response mode by binding between-brackets when the output reply is available to be sent or no conversational input is required. If no output reply is available to be sent or no conversational input is required, a warning message is sent to the IMS master terminal operator with instructions to retry session initiation later.

If IMS receives a negotiated BIND indicating in-brackets/SEND and is not in conversation or response mode, a BIND response is sent indicating between-brackets. IMS sends LUSTATUS - NO-OP indicating end-bracket immediately after SDT when IMS is not in conversation or response mode and when receiving a non-negotiated BIND or negotiated BIND response indicating in-brackets/SEND. LUSTATUS - NO-OP is sent because transaction restart is not possible under these circumstances.

It is possible that IMS is left in-brackets/SEND after a BIND or BIND response, because a conversation or response mode output sync-point response is pending from the previous session. The output message might be dequeued if STSN processing indicates that the response was sent. In this case, an LUSTATUS - NO-OP

## Resynchronizing Sessions

indicating end-bracket is to be sent after SDT if the output message is a response mode reply. An LUSTATUS - NO-OP indicating change-direction is sent if the output message is a conversational reply, because input is required to continue the conversation.

If STSN indicates that the conversation or response mode message was not acknowledged, IMS again places the message in a sync-point response-pending state as if it had been sent with change-direction and as if it had requested an exception sync-point response. Any normal flow reply is then considered implicit acknowledgment of the pending output message and causes it to be dequeued by IMS even if the reply was an LUSTATUS - Abort or an ATTACH ATTDPN=SYSMSG. These replies indicate additional error conditions associated with the message's scheduling or execution. Had the session not failed, these error conditions would have been reflected by an exception response to the sync-point request rather than the LUSTATUS - Abort or ATTACH ATTDPN=SYSMSG. Depending on the sense code used, the exception response might have caused the message to be returned to the message queue (backed out) for retransmission rather than dequeued (committed) as occurs for LUSTATUS - Abort or ATTACH ATTDPN=SYSMSG resulting when both a session and an application failure occur together.

### Session Failures without IMS Failure

A session failure not also involving an IMS subsystem failure might occur before IMS returns a requested sync-point response for a response mode or conversational input message. When the session is reestablished, the DFC state set by IMS using the bind is IMS in-brackets/SEND. The subsequent STSN sequence number recovery might reflect either that the input message has been committed (the transaction has reached a sync point after inserting a reply message) or that the input message has not yet been committed. In either case, IMS being bound as in-brackets/SEND indicates that the input message has been received and the other half session should wait for the reply message. Because session restart resets the original sync-point request, the reply message now becomes an implicit sync-point response to the original input message.

IMS recovers the fact that the session was in conversational mode (and also recovers associated input messages and output replies) across IMS outages. The fact that the session was in response mode (and any associated output replies) is only recovered across an IMS outage if the failure occurs after the transaction sync point that made the response mode reply available.

### Session Failures because of IMS Failure

A session failure that also involves an IMS subsystem failure might occur before IMS responds to a synchronization request on response mode (if the message was recoverable) or conversational mode input. When the session is reestablished, the DFC state set by IMS by the bind indicates IMS as bound in-brackets/SEND. In either case, because of the IMS restart process from the subsystem log, the subsequent STSN sequence number recovery reflects that the input message has been committed even if that commit has not yet actually occurred. However, this condition produces inconsistent results only if the transaction subsequently abnormally terminates after being restarted. In this case, the response mode output reply message is made available for asynchronous (ATTACH EB or ATTACH ATTDPN=SCHEDULER) delivery on the recovered ISC session.

### Recoverability of Commands and Execution Modes

IMS commands (except /DIS, /RDIS, and /FOR) and test-mode input are executed immediately without being placed on an input message queue. These commands complete all processing prior to causing output to be made available using a transaction sync point. Any failure causes the command or test mode transaction input and associated output message to be backed out and discarded.

IMS commands /DIS, /RDIS, and /FOR produce asynchronous queued output. Output that is enqueued prior to the failure is recovered and made available for asynchronous transmission when the session is reestablished.

Fast Path, recoverable response mode, and recoverable conversational mode transactions are backed out and discarded if the failure occurs before a transaction sync point. The fact that the session was in response mode is not recovered.

### Coordinating the Restart Process

The following rules are used by the half sessions to coordinate restart after a session failure:

- The restart always occurs from the most currently completed sync point.
- The session bind establishes the half-duplex state at the current sync point.
- When the primary half session wants to restart:
  - It sends a BIND request to select the proper half-duplex state.
  - The secondary half session cannot change this state in the BIND response, unless the secondary half session does not want to restart.
  - When the secondary half session does not want to restart, it sends a BIND response to preclude restart by setting the bracket state to between-brackets.
- When the primary half session does not want to restart:
  - The primary half session sends a BIND request setting the bracket state to between-brackets.
  - If the secondary half session wants to restart, it sends a BIND response that establishes the proper half-duplex state and sets the bracket state to in-brackets.
  - If the secondary half session does not want to restart, the secondary half session's response to the BIND does not change (it matches the state set by the primary half session).
- STSN is used to resynchronize any pending requests for sync-point responses.
  - If no mismatch occurred, agreement exists on the current sync point. The restart is attempted after sending start data traffic (SDT).
  - It might be necessary for the half session in send state to send LUSTATUS - NO-OP, CD to adjust the half-duplex state to the current sync point based on the STSN.

**Related Reading:** For more information on the LUSTATUS protocol, see "LUSTATUS Protocol" on page 308.
  - If a mismatch occurred and the session continues, the half session in send state sends LUSTATUS - NO-OP, EB to place the session into contention.
- The session in send state at the final restart point reestablishes the session by sending an explicit ATTACH.

Half sessions assuming a secondary role must reject a session bind in the following situations:

## Resynchronizing Sessions

- Session bind parameters indicate in-brackets, “other half session speaks first”, and response mode or conversational output (or equivalent) are immediately available at data traffic active state. This is a session restart logic error.
- On restart following an abnormal failure or shutdown sequence, session bind parameters indicate half-session names different from those in effect during the previous session between IMS and another logical unit.

In an ISC session, either subsystem can assume a primary or a secondary half-session role, and either subsystem can request message resynchronization; therefore, a set of rules must be established by which these half-session pairs can maintain message or sync point integrity.

The following sections describe the format of the STSN and how the STSN is used to complete resynchronization and recovery.

## Determining Session Synchronism Using STSN

The need to resynchronize can be communicated during bind negotiation; two flags in the BIND request are used to determine the requirement to resynchronize and to return the half sessions to the state that existed at the time of session termination.

If both half sessions have terminated normally (that is, neither had any outstanding traffic to handle), the session is restarted as though it were a new session—bind parameters are sent, and no requirement exists that the relative positions of the session partners be maintained, nor that the same half-session names be used. The two flags upon which this determination is made are:

- Sequence number indicator
  - 1 = Sequence numbers available
  - 0 = Sequence numbers not available
- Bracket initiation stopped (BIS) indicator
  - 1 = BIS sent
  - 0 = BIS not sent

**Related Reading:** For more information on the BIS indicator, see “Symmetrical Session Shutdown for LU 6.1 (SBI and BIS)” on page 320.

Table 19 on page 277 is a matrix describing the states set by these two flags in relation both to the bind sender (primary half session, or PHS) and the bind receiver (secondary half session, or SHS).

When both half sessions are in a “COLD START” state, no sequence numbers are available or required to be sent. Session shutdown is such that resynchronization is not required. The bind is negotiated and the session started in accordance with the bind parameters. For a nonnegotiable BIND, some information required by both half sessions is not available until STSN flows exist. Therefore, for a nonnegotiable BIND, STSN is always sent.

**Related Reading:** For more information on the resynchronization process that occurs for these conditions, see “Performing the Resynchronization” on page 277.

Table 19. BIND Action/Response Matrix

Bind Sender (PHS)	Bind Receiver (SHS)		
	Numbers Not Available	Numbers Available	
		BIS Sent	BIS Not Sent (See Table 20)
<b>Numbers Not Available</b>	COLD STSN not required Send RESET to NOBB	COLD STSN not required Send RESET to NOBB	COLD/WARM mismatch STSN sent
<b>Numbers Available BIS Sent</b>	COLD STSN not required Sent RESET to NOBB	COLD STSN not required Send RESET to NOBB	WARM STSN sent
<b>Numbers Available BIS Not Sent (Table 21 on page 282)</b>	WARM/COLD mismatch STSN sent	WARM STSN sent	WARM STSN sent

IMS sets the “BIS sent” and “sequence numbers available” flags in the negotiable BIND and BIND response (or PHS and SHS, respectively) as follows:

- Session cold start following IMS cold start or normal session termination:
  - BIS not sent
  - Sequence numbers not available
- Session restart
  - Previous session terminated normally following attempted normal termination by IMS:
    - BIS sent
    - Sequence numbers available
  - Previous session terminated abnormally (normal termination by IMS not attempted):
    - BIS not sent
    - Sequence numbers available

**Related Reading:** For information on the format of the STSN command, see “STSN Command Format” on page 283.

## Performing the Resynchronization

When a session is reestablished (or established for the first time), the sequence numbers available for synchronization in each half session can, for each flow, represent the following states:

- COLD
 

This session has no sequence numbers to send or to compare. The primary half session indicates this by resetting the sequence number indicator bit in the BIND request. If the BIND is negotiable, the secondary half session indicates this by the same bit setting in the BIND response. If STSN is required (as for nonnegotiable BIND), the primary half session sends a SET AND TEST action code with zeros as the sequence number values.
- NOT PENDING
 

This session has no work units that are in-doubt (that is, waiting to be committed)
- PENDING
 

One work unit is in doubt (waiting to be committed). The sequence numbers available to the session are the committed number (the sequence number of the last message to be committed) and the potential number (the sequence number of the last message to be issued).

## Resynchronizing Sessions

- **DECISION TO COMMIT**  
An in-doubt work unit is committed during the session outage. Potential and committed numbers are still available. (A decision to commit occurs only as the result of a /DEQ command being issued during a session outage.)
- **DECISION TO BACK OUT**  
An in-doubt work unit is backed out during the session outage. Potential and committed numbers are still available. (IMS does not back out work units, but other session partners might).
- **INVALID**  
Invalid sequence numbers are detected. (An invalid sequence number is one that should not occur, as, for example, a sequence number sent on the secondary-to-primary flow that does not match the committed or potential numbers saved by the STSN receiver.) This type of condition usually indicates an incorrect log data set; the master terminal operator must intervene to recover the session.
- **RECEIVED PENDING (Sync-Point Response Lost)**  
The receiver of this flow receives the entire work unit that the sender sends and generates a DR2 that might be lost in the network.
- **NOT RECEIVED PENDING (Sync-Point Response Not Lost)**  
The receiver of this flow never receives the RQ\*2 that the sender sends and therefore backs out the work unit that the sender is pending on.

The half sessions exchange information on their respective states by using bind negotiation and the action codes in the STSN request and response.

**Related Reading:** For information on the codes sent on STSN and the consequences, see “Using STSN to Resynchronize Sessions” on page 280.

---

## Completing Session Initiation

After the half sessions are in sync, either half session can decide to accept or reject the session:

- If the primary half session does not want to continue the session, it sends UNBIND.
- If the primary half session wants to continue the session, it sends start data traffic (SDT).
  - If the secondary half session responds positively, the session is formally bound and traffic on the session can begin.
  - If the secondary half session wants to discontinue the session, it rejects the primary half session’s SDT, whereupon the primary must respond with an UNBIND and the session terminates.

Session initiation, including allocation of the LTERM subpool, is completed when a start data traffic (SDT) response has been received if IMS is the primary half session or when an SDT response is sent if IMS is the secondary half session. The LTERM subpool remains allocated to the other half-session name, even across session or subsystem failures, until the session is terminated by mutual consent of both half sessions (symmetrical shutdown). This requires that, after they have been allocated, all subsequent session binds for the same half session must specify the same bind session qualifiers that were active at abnormal session termination until the subpool is released through normal session termination.

## Completing Session Initiation

If the session is terminated prior to the completion of SDT, any newly attempted subpool allocation caused by a session cold start is backed out, and the subpool is returned to an available-for-allocation status.

In IMS, the master terminal operator must always be notified of any session that is rejected prior to a start data traffic (SDT) completion. In an ISC session, notification to the IMS master terminal operator is optional for normal initiation and termination sequences. Operator notification is specified by the OPTIONS keyword parameters MTOMSG and NOMTOMSG on the TYPE and TERMINAL macros or by an ETO logon descriptor.

---

## Running the Session

Use the data flow control protocols to control the flow of data in an ISC session.

### **Related Reading:**

- For information on the data flow protocols, see “Controlling Data Flow (DFC Protocols)” on page 285 and “Data Flow Control Protocol Reference” on page 295.
- For information on response and conversation mode errors and transaction sync points, see “Controlling Data Flow (DFC Protocols)” on page 285.

---

## Terminating the Session

**Definition:** *Session termination* releases a logical unit from its current connection to the VTAM application program, making the LU available for sessions with other VTAM applications, or terminating communications altogether.

The two types of session termination are normal and abnormal.

### **Definitions:**

- *Normal termination* allows both half sessions to complete normal processing before the session is terminated.
- *Abnormal termination* forces the session to terminate unconditionally.

Session termination can be invoked by the IMS master terminal operator, the VTAM network operator, or either half session.

Because of this variety of session termination methods, each IMS network installation must determine specific procedures for session termination. When developing these procedures, consider the requirements for session-termination processing.

## Normal Termination

Normal termination of an ISC session occurs with the flow of the data flow control indicators stop bracket initiation (SBI) and bracket initiation stopped (BIS). Normal termination can be initiated by either half session.

**Related Reading:** For more information on SBI and BIS, see “Symmetrical Session Shutdown for LU 6.1 (SBI and BIS)” on page 320.

Normal session termination also occurs when the MTO invokes an orderly termination of the IMS network by the IMS /CHECKPOINT command with the FREEZE,

## Terminating the Session

PURGE, or DUMPQ parameter and the QUIESCE parameter. The QUIESCE parameter ensures that message queues are emptied before the session is terminated.

When all terminals have indicated that shutdown is complete, IMS:

- Performs checkpoint processing and then issues the VTAM CLSDST macro instruction, when acting as primary half session.
- Awaits UNBIND, if acting as secondary half session.

CLSDST causes VTAM to send the UNBIND command. CLSDST or UNBIND releases the logical units from session with IMS. Any further data transmission to the logical units is prohibited.

During the processing of an orderly session termination, the IMS master terminal operator can terminate the network unconditionally by using an IMS /CLSDST, /STOP, or /CHECKPOINT command.

## Abnormal Termination

Abnormal session termination can occur as a result of transmission or protocol errors, or errors in data that make that data unacceptable to the receiving message processing program. Because an ISC session involves two peer-level systems, error recovery processing and abnormal session termination processes can differ. IMS-detected error conditions requiring abnormal session termination result in IMS issuing a VTAM CLSDST macro when IMS is the primary half session, or TERMSESS with OPTCD=UNCOND when IMS is the secondary half session.

**Related Reading:** For information on the effects of error recovery processing and some situations that can result in abnormal session termination, see “Data Flow Control Protocol Reference” on page 295.

---

## Using STSN to Resynchronize Sessions

Table 20 and Table 20 show the results and actions related to using STSN for primary-to-secondary (P-S) and secondary-to-primary (S-P) flows. The first result in each table cell represents the STSN command action; the second result represents the response returned to the STSN command. These are separated by a slash (/). When two STSNs are sent, they are separated by a comma. The results are further explained in the notes following the tables.

## Primary-to-Secondary Flow Matrix

Table 20 illustrates the actions taken when resynchronizing a session and the STSN command flows from the primary half session to the secondary half session.

*Table 20. STSN Primary-to-Secondary Flow*

Primary	Secondary		
	Session Cold	Not Received: Pending	Received: Pending
Pending	<b>1</b> Set and Test / Reset	<b>2</b> Set and Test / Negative	<b>3</b> Set and Test / Positive
Decision to Commit	<b>4</b> Set and Test / Reset	<b>5</b> Set and Test / Negative, Set/Positive	<b>6</b> Set and Test / Positive
Decision to Back Out	<b>7</b> Set and Test / Reset	<b>8</b> Set and Test / Positive	<b>9</b> Set and Test / Negative, Set/Positive
Not Pending	<b>10</b> Set and Test / Reset	<b>11</b> Set and Test / Positive	Not Applicable



## Using STSN to Resynchronize Sessions

### Notes To Table 20 on page 280:

- 1** The STSN sender is PENDING. The STSN receiver has no sequence numbers. A COLD/WARM mismatch has occurred. The session can continue. The STSN sender should continue the session.
- 2** The STSN sender is PENDING. The STSN receiver never received the pending RQ\*2 and returns a TEST NEGATIVE in the STSN response. The STSN sender backs out the pending work unit to the last commit point.
- 3** The STSN sender is PENDING. The STSN receiver received the pending RQ\*2 and sent a + DR2. The STSN receiver responds TEST POSITIVE, and the STSN sender commits the pending work unit.
- 4** The STSN sender wants to release the locks and commit resources; however, an RQ\*2 is outstanding. The STSN receiver has no sequence numbers. A COLD/WARM mismatch has occurred. The session can continue. The STSN sender should continue the session.
- 5** The STSN sender wants to release the locks and commit resources; however, an RQ\*2 is outstanding. The STSN receiver never received the pending chain, as indicated by TEST NEGATIVE in the STSN response. The STSN sender sends a second STSN with the SET action code to inform the STSN receiver that a wrong decision was made. The STSN receiver can continue the session by sending TEST POSITIVE or decline by sending INVALID.
- 6** The STSN sender wants to release locks and commit resources; however, an RQ\*2 is outstanding. The STSN receiver received the pending chain and committed it, as indicated by the TEST POSITIVE in the STSN response.
- 7** The STSN sender wants to back out the pending work unit, but a pending RQ\*2 is outstanding. The STSN receiver has no sequence numbers. A COLD/WARM mismatch has occurred. The session can continue. The STSN sender should continue the session.
- 8** The STSN sender wants to back out the pending work unit, but an RQ\*2 is outstanding. The STSN receiver never received the pending chain, so the sequence numbers match. A TEST POSITIVE is returned on the STSN response.
- 9** The STSN sender wants to back out the pending work unit, but an RQ\*2 is outstanding. The STSN receiver received the pending chain and committed resources. Because the STSN sender sent out the old committed sequence numbers and the STSN receiver received the chain, a TEST NEGATIVE is returned. The STSN sender informs the STSN receiver that a wrong decision has been made by sending a second STSN with a SET action code. The STSN receiver can continue the session by sending TEST POSITIVE or decline by sending INVALID.
- 10** The STSN sender is not pending on the P-S flow. The STSN receiver has no sequence numbers. The session continues.
- 11** The STSN sender is not pending on the P-S flow. The STSN receiver agrees with the sequence numbers on the STSN and returns a TEST POSITIVE.

## Using STSN to Resynchronize Sessions

### Secondary-to-Primary Flow Matrix

Table 21 illustrates the actions taken when resynchronizing a session and the STSN command flows from the secondary half session to the primary half session.

Table 21. STSN Secondary-to-Primary Flow

Primary	Secondary					
	Session Cold	Pending	Decision To Commit	Decision To Back Out	Invalid	Not Pending
Not Received: Pending	<b>1</b> Set and Test / Reset	<b>2</b> Set and Test/ Positive	<b>3</b> Set and Test/ Negative	<b>4</b> Set and Test/ Positive	<b>5</b> Set and Test / Invalid	<b>6</b> Set and Test / Positive
Received: Pending	<b>7</b> Set and Test / Reset	<b>8</b> Set and Test/ Positive	<b>9</b> Set and Test/ Positive	<b>10</b> Set and Test/ Negative	<b>11</b> Set and Test / Invalid	Not Applicable

#### Notes To Table 21:

- 1** The STSN sender is not COLD. The STSN receiver has no sequence numbers. The session continues.
- 2** The STSN receiver has a pending work unit and has held onto locks and not committed resources. The sequence number sent on the SET AND TEST action code indicates that the STSN sender did not receive the pending RQ\*2. Therefore, the STSN receiver backs out the pending work unit to the last commit point and returns a TEST POSITIVE on the STSN response.
- 3** The STSN receiver had a work unit pending and decides to commit resources and release locks. The STSN sender did not receive the RQ\*2 sent by the STSN receiver. The STSN receiver returns a TEST NEGATIVE on the STSN response to inform the STSN sender that a wrong decision was made. The STSN sender can continue the session by sending SDT or decline by sending UNBIND.
- 4** The STSN receiver had a work unit pending and decides to back out the pending work unit. The STSN sender never received the pending work unit, and the STSN receiver returns a TEST POSITIVE on the STSN response.
- 5** The STSN receiver detects a severe loss of synchronization (possible log data set mismatch) and returns an INVALID on the STSN response.
- 6** The STSN receiver has no pending work unit and agrees with the sequence numbers sent on the STSN; therefore, it replies with a TEST POSITIVE.
- 7** The STSN sender is not COLD. The STSN receiver has no sequence numbers. The session continues.
- 8** The STSN receiver has a pending work unit and still holds resource locks. The number sent by the STSN sender on the S-P SET AND TEST indicate to the STSN receiver that the STSN sender received the pending work unit. The STSN receiver commits the pending work unit and responds TEST POSITIVE.
- 9** The STSN receiver had a pending work unit and decides to commit the pending work unit. The number sent by the STSN sender on the S-P flow indicates that the STSN sender received the RQ\*2 and a TEST POSITIVE is sent on the STSN response.
- 10** The STSN receiver had a pending work unit. The receiver decides

## Using STSN to Resynchronize Sessions

to back out that work unit and release the locks. The number sent by the STSN sender on the S-P flow indicates to the STSN receiver that the RQ\*2 was received and processed by the STSN sender. The STSN receiver indicates the wrong decision by a TEST NEGATIVE on the STSN response. The STSN sender can continue the session by sending SDT or decline by sending UNBIND.

### 11

The STSN receiver detects a severe loss of synchronization (possible log data set mismatch) and indicates this to the STSN sender by an INVALID in the STSN response.

---

## STSN Command Format

Both the STSN command and STSN response contain a 5-byte data field. The format of the STSN command is:

<b>Byte 0</b>	Action code
<b>Bytes 1, 2</b>	Sequence number of the last inbound sync point message sent to the PHS
<b>Bytes 3, 4</b>	Sequence number of the last outbound sync point message sent from the PHS

The primary half session uses the action code to ask the secondary half session to verify the VTAM sequence numbers. The bits of the action code byte are as follows:

<b>Bits 0 and 1</b>	Refer to the inbound sequence-number field
<b>Bits 2 and 3</b>	Refer to the outbound sequence-number field
<b>Bits 4 through 7</b>	Reserved

The following values are acceptable for bits 0, 1, 2, and 3 of the STSN command action code:

<b>00 IGNORE</b>	Do not alter value. IMS does not send this action code value as PHS. IMS as SHS returns an INVALID response code for this action code when resynchronization is required and TEST POSITIVE when resynchronization is not required.
<b>01 SET</b>	Set the appropriate sequence number to the value indicated in the sequence number field. For ISC, this code only occurs when the PHS must send a second STSN to indicate a unilateral decision was made to commit or back out a pending work unit.
<b>10 SENSE</b>	Do not alter value. The SHS should return its version of the sequence number in the command response. IMS as SHS returns an INVALID response to this action code.
<b>11 SET AND TEST</b>	Set the appropriate sequence number to the value indicated in the sequence number field. For ISC, this code always occurs for both flows on the first STSN sent by the PHS. The SHS must indicate in the command response whether the sequence number values are acceptable.

When the secondary half session receives the STSN command, it must be able to:

- Verify the SHS-outbound (PHS-inbound) sequence number and arrange to retransmit a message to the PHS if required.

## STSN Command Format

- Verify the SHS-inbound (PHS-outbound) sequence number and inform the PHS whether the number matches the SHS-saved number. Further, the SHS might or might not agree with unilateral decisions by the PHS to commit or back out a pending work unit from the previous session.
- Return to the PHS a DR1 and a 5-byte data response to the STSN command.

The format of the STSN response has the same format as the STSN command:

<b>Byte 0</b>	Action code
<b>Bytes 1, 2</b>	Sequence number of the last recoverable message the SHS sent to the PHS
<b>Bytes 3, 4</b>	Sequence number of the last recoverable message the SHS received from the PHS

The secondary half session (SHS) uses the action code to indicate the test results. The action code must be returned to the primary half session (PHS). Returning the sequence numbers (bytes 1 through 4) is optional. However, for debugging recovery and restart problems, always return these to the PHS whenever the STSN numbers do not match those maintained by the SHS. The bits of the action code byte are as follows:

<b>Bits 0 and 1</b>	Refer to the SHS inbound sequence number
<b>Bits 2 and 3</b>	Refer to the SHS outbound sequence number
<b>Bits 4 - 7</b>	Reserved

The following values are acceptable for bits 0, 1, 2, and 3 of the STSN response action code:

**00 RESET** Returned on both sequence number flows in response to the SET AND TEST option to indicate that the SHS must cold start and has no sequence number information. The PHS should continue session initiation and should not treat this response as an error. IMS sends this code when cold-starting and replying to STSN, and continues the session when receiving it.

### 01 TEST POSITIVE

Returned in response to the SET AND TEST option to indicate that the sequence number agrees with the sequence number that the SHS checkpointed. This code should be returned in response to the SET option when the SHS agrees to continue session initiation following a unilateral decision by the PHS to commit or back out a pending work unit on its outbound flow during the session outage.

**Related Reading:** For more information on IMS action for unilateral decisions, see “Controlling Unilateral Decisions about Pending Work Units” on page 272.

### 10 INVALID

Returned in response to the SET AND TEST option to indicate that the STSN SHS outbound sequence number does not agree with the sequence number that the SHS checkpointed (major session restart mismatch). INVALID is also returned in response to the SET option to indicate disagreement with a unilateral action by the PHS to commit or back out a pending work unit on its outbound flow. The PHS should not continue the session when receiving an INVALID response to either flow.

**Related Reading:** For more information on IMS action for unilateral decisions, see “Controlling Unilateral Decisions about Pending Work Units” on page 272.

### 11 TEST NEGATIVE

Returned in response to the SET AND TEST option on the SHS inbound flow to indicate that the sequence number does not agree with the sequence number checkpointed by the SHS. The PHS responds with SDT if the reason for the mismatch is a pending work unit sent by the PHS, but is not received by the SHS; with a second STSN indicating the SET option for both flows if the PHS had made a unilateral decision to commit or back out a pending work unit; or UNBIND if no work unit was pending and no unilateral decision had been made (major session restart mismatch). TEST NEGATIVE can also be returned in response to the SET AND TEST option to indicate that the SHS had made a unilateral decision to commit or back out a pending work unit on its outbound flow. In this case, the PHS can optionally agree to continue session initiation by sending SET or not to continue session initiation by sending UNBIND.

**Related Reading:** For more information on IMS action for unilateral decisions, see “Controlling Unilateral Decisions about Pending Work Units” on page 272.

---

## Controlling Data Flow (DFC Protocols)

This section describes how IMS handles response and conversational mode errors during an ISC session and how to keep the half sessions in sync.

**Related Reading:** For information on the DFC protocols, see:

- “Data Flow Control Protocol Reference” on page 295. This reference provides brief explanations and format tables for individual data flow control protocols.
- *Systems Network Architecture Format and Protocol Reference: Architectural Logic*

After an exception response to a DFC command, you might need to purge before further sends or receives.

**Related Reading:** For information on ERP purging, see “ERP Purging” on page 305.

## Handling IMS Response Mode or Conversational Output Errors

IMS messages cannot be cancelled by session termination or protocols after they have been received and enqueued (made available for scheduling). However, errors can be detected by either half session after the input message has been enqueued. If the error is detected by IMS while processing the transaction, but before the reply message is committed (for example, a DFS555 transaction abend occurs), both half sessions are able to back out, because IMS holds the input sync-point response until the reply message is available for output. An exception response and appropriate error recovery process (ERP) message are returned. Errors occurring after the reply message is made available for output result in backout to the last application sync point (the one that made the reply message available). These errors are not communicated to the other half session.

## Controlling Data Flow (DFC Protocols)

### Response Mode Errors

Errors detected by the other half session cannot be communicated to IMS until IMS attempts to send the reply. At this point, errors must be communicated to IMS before the requested sync-point response is returned, because all response mode output is sent indicating RQD and EB. Errors are reflected by the return of an exception response with an optional ERP message, which might contain appropriate sense data and protocols.

The exception response or ERP FM header sense codes can cause the output message to be backed out (dequeued) or retransmitted, or they can cause the session to terminate with the message still on the queue. ERP message protocols can cause the message to be backed out (dequeued) regardless of the sense code via use of EB or can leave IMS in send state via use of CD. This latter case is only allowed if the sense code used results in the output message being dequeued or retransmitted.

An LUSTATUS - abort with EB is sent as the next output on the session if CD is used on the ERP message and the sense code results in dequeuing the response mode output message. However, in all of the cases where the error is detected by the other half session, the reply message is the only resource that IMS can back out. The database updates and other messages initiated by the response mode transaction cannot be backed out by IMS.

### Conversational Mode Errors

Errors detected by the other half session cannot be communicated to IMS until IMS attempts to send the output reply. Errors must be communicated to IMS before the requested sync-point response is completed. For nonlast conversational output, the sync-point requested using RQE2 with CD and any returned normal flow data implicitly completes the sync point. The last conversational output message is sent requesting RQD2 with EB. Errors are reflected either by returning an exception response with an optional ERP message (which might contain appropriate sense data and protocols) or by returning an LUSTATUS with appropriate sense codes and protocols.

The sense codes can cause the message to be backed out (dequeued) or retransmitted, or they can cause the session to terminate with the message still on the queue. ERP and LUSTATUS message protocols can cause the message to be backed out (dequeued) using EB, or they can leave IMS in send state using CD. This latter case is only allowed if the sense code retransmits the message or if the last conversational output for the sense code dequeues the output message. An LUSTATUS - abort with EB is sent as the next output on the session if CD is used on the LUSTATUS or ERP message, and the sense code results in dequeuing the last conversational mode output message.

When the error is detected by the other half session, the output reply message is the only resource that IMS can automatically back out. The database updates, conversational SPA, and other messages initiated by the conversational transaction cannot be automatically backed out by IMS. However, for conversations, an internal IMS /EXIT command is scheduled to invoke the user's Conversational Abnormal Termination exit routine, which can optionally schedule a transaction to reverse necessary database changes based on the conversational SPA content. The interface to the exit routine is the same as if an /EXIT command had been received on the session.

**Related Reading:** For more information on this interface, see “Conversational Abnormal Termination Exit Routine (DFSCONE0)” in *IMS/ESA Customization Guide*.

---

### Normal Conversation Termination Extension with ISC

In a non-ISC environment, normal termination of IMS conversation mode occurs when the IMS transaction creates a blank in the transaction-code field of the conversational SPA prior to committing the reply message. The conversation ends when the reply message has been successfully dequeued. An extension for normal termination has been made for ISC, because the peer-level application in the other half session can also now end the conversation. However, IMS supports this request for normal termination by receiving only an SNA-defined commit request - LUSTATUS X'0006' with EB. This occurs because of a stand-alone commit request (or normal termination) by the remote application to its subsystem. This form of conversational termination requires notification of the user using the Conversational Abnormal Termination exit routine (new input vector of X'28' in byte 3 of register 1), because committing changes might have been deferred by the completed conversational steps. This is possible by recording the deferred information within the conversational SPA. The exit routine can schedule an appropriate transaction to commit the deferred changes upon being invoked with the new vector.

**Restriction:** IMS does not support an input message with EB to terminate the conversation.

---

### Keeping Half Sessions Synchronized

This section describes the synchronization requirements for ISC sessions and the attached application programs.

**Related Reading:** For information on sync points and associated commit or backout processes for the IMS application program interface, see *IMS/ESA Application Programming: Transaction Manager*.

Sync-point responses (DR2) are used between ISC session partners to ensure that both partners' sync-point managers can commit or back out recoverable resources synchronously. All messages sent or received on an IMS ISC session are defined as either recoverable or irrecoverable, depending on the message type. The session-response protocols are used to ensure that both ends of an ISC session mutually understand the recoverability attributes associated with each message. The response protocol used must be consistent with the IMS message type.

The ISC sync point can be explicit or implicit:

#### **Explicit**

The input requests an RQD2 response.

#### **Implicit**

The input requests the sending of an exception response (RQE2) and change-direction on output made available by the transaction, or the sending of an LUSTATUS in lieu of output data from the transaction).

To further increase integrity and recoverability, IMS can supplement the sync point facility by logging the information. Use of log write-ahead ensures that the sync point indication is recorded on the log. This makes sync point information available

## Keeping Half Sessions Synchronized

to IMS restart procedures before the sync-point response is actually sent or change-direction is replied to (implied sync point).

### Related Reading:

- For more information on the sync point requirements, see “Sync Point and Response Requirements” on page 289.
- For tables showing the VTAM indicators used in IMS messages to request and respond to sync points, see “Sync-Point Indicators on Messages” on page 292.

Before reading these sections, be sure to understand the definitions and relationships of the ISC sync point and associated commit and backout processes to the IMS application program, described in the following paragraphs.

## Sync Points Requested on Input to IMS

For any type of input, IMS does not schedule the intended transaction until the complete input message is successfully received. Sender-detected errors, errors resulting from processing of IMS input, and session failure prior to the receipt of the complete input message cause the entire message to be discarded or backed out. However, the input message cannot be cancelled by ISC session failures or protocols after the complete message is received and made available for scheduling.

When an input message is backed out because of errors detected during IMS input processing or during synchronous transaction execution, the session partner is notified, either by means of session termination or by an exception response to the ISC input. The following events occur for this backout:

- Backout results in resetting the associated DFC and ATTACH states to those of the last sync point.
- Backout during transaction execution results in backing out application updates and messages, except express messages made after the last application sync point.
- The application sync point and ISC sync point are not necessarily the same.
- Backout during input to IMS has no effect on other recoverable IMS resources, such as databases, because input messages are not available for scheduling or execution until they are received completely and without error.
- The result of the backout is only the current input message, even if several consecutive input (irrecoverable) messages were received and executed or enqueued for scheduling after the last input sync point was requested from IMS.

The definition and relationship of successful ISC input sync points to IMS application sync points depend upon whether the ISC input was synchronous or asynchronous. For asynchronous input, the sync point reflects only that IMS is now responsible for message recovery. No implications exist relative to the scheduling or execution of transactions or to the availability of transaction output. After the message is successfully enqueued, the DFC and ATTACH sync point information is updated and the requested sync-point response is returned to the session partner.

Although some IMS exceptions for synchronous input exist, the sync point is intended to reflect the success of the IMS transaction execution and sync point. IMS updates the DFC and ATTACH sync point information as appropriate and commits all associated transaction resources (for example, DL/I databases) and the output transaction message reply when responding to the ISC attached input sync-point request.



## Keeping Half Sessions Synchronized

The following exceptions apply to sync points for synchronous input to IMS:

- The ISC sync-point response is returned by IMS when a transaction-inserted response mode or conversational reply message (first message inserted to the I/O PCB or alternate response PCB) is made available for output using a transaction sync point. Additional transaction processing and sync points are not reflected in the ISC session. No ISC sync point-response occurs for transaction sync points prior to the transaction's inserting the reply message, as in a program-to-program switch. Nonfirst messages inserted to the I/O PCB or alternate response PCB, messages inserted to nonresponse PCBs, and IMS express messages intended for the ISC session are queued for future asynchronous delivery after successful delivery of the reply message.
- Except during program-to-program switches to another conversational transaction, IMS generates a subsystem error message as the result of abnormally terminating a conversational transaction that attempts to cause a transaction sync point without first having inserted an output conversational reply message. The subsystem error message causes an exception response to be sent to the input sync-point request and the input message to be backed out.

## Sync Points Requested on Output by IMS

IMS commits the output message when the requested sync-point response is returned by the other session partner. The message might also be committed as the result of some sense codes that are returned on an exception response to the requested sync-point response.

**Definition:** The term *commit* means that the message has been successfully sent and dequeued, and sync point information has been updated as appropriate.

Use of the IMS /DEQUEUE command during a session outage while an output sync-point response is pending is considered a unilateral decision by an authorized terminal operator to commit the sending output message.

Depending upon the sense code used, IMS backs out the output message when an exception response is returned to the sync-point request or when the IMS /DEQUEUE command is issued by an authorized terminal operator before IMS requests a sync-point response.

**Definition:** The term *backout* means that a recoverable message is returned to the message queue (unless dequeued by a /DEQUEUE command) for subsequent retransmission. An irrecoverable message is either dequeued or returned to the message queue for subsequent retransmission, depending upon the type of error. Normally, an exception response or IMS failure causes an irrecoverable message to be dequeued. Some session failures that do not result from a subsystem failure cause an irrecoverable message to be retransmitted at the first opportunity after resynchronization. Backout results in resetting the associated DFC and ATTACH states to those of the last sync point.

## Sync Point and Response Requirements

The IMS input/output message flow can be represented as an input/output flow from a sequential queue data set. In order to maintain integrity and recoverability of this queue data set, IMS requires that each recoverable input and output message establish a sync point between both half sessions before continuing the flow. This allows both half session sync-point managers to commit or back out resources in synchronism.

## Keeping Half Sessions Synchronized

The sync point facility for ISC redefines and separates the DR1 and DR2 requests and responses. The DR2 requests and responses are known as sync-point requests and responses and are functionally independent from those associated with DR1.

### Recoverable Messages

To ensure that a recoverable transaction can be recovered, IMS requires the following response protocol for each recoverable message sent or received:

- Messages that are not MFS-paged messages:
  - An exception DR2 (RQE2) must be requested on each nonlast (or when it is not the only) RU of an SNA chain.
  - Except when change-direction is sent, each last or only RU of an SNA chain must request a DR2 (RQD2). Either exception DR2 (RQE2) or RQD2 can be requested when change-direction is sent.
  - A sync-point response on the last or only RU of an SNA chain always indicates end-of-message. End-of-message always occurs at end-of-chain for single chain (non-MFS-paged) messages.
- First chain of asynchronous (ATTACH SCHEDULER) demand-paged message:
  - The first chain (OIC) of asynchronous demand-paged messages is an ATTACH for the SCHEDULER model. This OIC requests a DR2 (RQD2) with end-bracket. This allows the application to be asynchronously scheduled to receive the message and leaves the session in a state to be allocated to the scheduled application.
- Nonlast pages of MFS demand-paged (output using ATTACH or ATTACH SCHEDULER) messages, and last pages of MFS-operator logical-paged (OLP) output:
  - An exception DR1 (RQE1) is requested on each nonlast (or when it is not the only) RU of an SNA chain (MFS demand-page).
  - Each last or only RU of an SNA chain (MFS demand-page) requests exception DR1 (RQE1) with change-direction.
- Last pages of MFS demand-paged (output) messages:
  - An exception DR2 (RQE2) is requested on each nonlast (or when it is not the only) RU of an SNA chain (MFS demand-page).
  - Except when change-direction is sent on the last page, each last or only RU of an SNA chain (MFS demand-page) requests a DR2 (RQD2). Exception DR2 (RQE2) with change-direction is requested.
  - A sync-point response on the last or only RU of an SNA chain always indicates end-of-message. A sync point must always be requested on the last page of MFS demand-paged messages to ensure end-of-message.
- First, nonlast pages of MFS-autopaged messages:
  - An exception DR1 (RQE1) is requested on each nonlast (or when it is not the only) RU of an SNA chain (MFS autopage) sent to IMS and must be requested on each nonlast (or when it is not the only) RU of an SNA chain (MFS autopage) received by IMS.
  - RQD1 is requested on the last or only RU of the first SNA chain of an MFS-autopaged output message sent by IMS.
  - RQD1 must be requested on the last or only RU of the first SNA chain of an MFS-autopaged input message received by IMS when the other half session is initialized as the primary half session (bidder). Either exception DR1 (RQE1) or DR1 (RQD1) can be requested on the last or only RU of this SNA chain when the other half session is initialized as the secondary half session (first speaker).

## Keeping Half Sessions Synchronized

Change-direction is not sent and must not be requested on nonlast pages of MFS-autopaged input and output messages.

- Nonfirst, nonlast pages of MFS-autopaged messages:

An exception DR1 (RQE1) is requested on each RU (including last or only) of an SNA chain (MFS autopage) sent by IMS. An exception DR1 (RQE1) must be requested on each nonlast (or when it is not the only) RU of an SNA chain received by IMS. Either exception DR1 (RQE1) or RQD1 can be requested on each last or only RU of an SNA chain received by IMS. Change-direction is not sent and must not be requested on nonlast pages of MFS-autopaged input and output.

- Last pages of MFS autopaged messages:

An exception DR2 (RQE2) is requested on each nonlast RU of an SNA chain (MFS autopage) sent by IMS and must be requested on each nonlast RU of an SNA chain (MFS autopage) received by IMS.

Except when change-direction is sent on the last page, each last or only RU of an SNA chain (MFS autopage) requests a DR2 (RQD2). Exception DR2 (RQE2) is requested when change-direction is sent.

Except when change-direction is received on the last page, each last or only RU of an SNA chain of an MFS-autopaged input must request a DR2 (RQD2). Either exception DR2 (RQE2) or definite response 2 (RQD2) can be requested when change-direction is indicated.

A sync-point response on the last or only RU of an SNA chain or on an LUSTATUS - commit always indicates end-of-message. A sync point must always be requested on the last page or on an LUSTATUS - commit following the last page of autopaged messages to ensure end-of-message.

**Related Reading:** For more information on LUSTATUS - commit, see “LUSTATUS Protocol” on page 308.

For the case when allowing RQE1 on the last or only RU of an SNA chain that does not indicate a change-direction for MFS-autopaged input or output, an exception to the preceding protocols occurs when the session bind indicates DEFINITE RESPONSE CHAINS. If the definite response chains parameter is set for the IMS half session, each last or only RU defined above that does not also indicate change-direction is sent requesting RQD1 or RQD2. RQD1 or RQD2 must be requested by the other half session under these same conditions if its session bind indicates DEFINITE RESPONSE CHAINS. Both exception and definite response (DR1 and DR2) are valid if change-direction is indicated under definite response chain rules.

### Irrecoverable Messages

IMS treats an irrecoverable message in the same manner as a recoverable one, except that all processing required to achieve recoverability is eliminated. As a result, irrecoverable messages require less processing time, but can be lost in the event of a failure. Irrecoverable, non-MFS input and output messages basically have the same requirements as those for recoverable messages, except that DR1 and exception DR1 can optionally be requested instead of DR2 and exception DR2, respectively. Irrecoverable MFS-paged input and output messages have the same requirements as recoverable ones.

When messages are being sent, only one message can be outstanding at a time. This means that the sender can send one message and must wait for the response or reply before sending another.

## Keeping Half Sessions Synchronized

The required response and sync-point protocol allows message integrity to be maintained by allowing the half sessions' sync-point managers to mutually understand when messages are accepted (committed) or rejected (backed out). This also allows change-direction to be solicited using SIGNAL RCD as required. RQE1 or RQE2 is recommended where change-direction is indicated, because these capabilities are automatic. An RQD1 or RQD2 are valid, but reduce performance because of the unnecessary response. A response or sync point is implied when a reply is received to a sent message indicating change-direction and RQE1 or RQE2. That is, the reply is an implied DR1 or DR2 response. Requesting either a definite response (RQD1 or RQD2) or an exception response (RQE1 or RQE2) with change-direction is valid for the session bind option DEFINITE RESPONSE CHAINS.

A failure can occur between sending a recoverable message and receiving the sync-point response (or reply) and receiving the sync-point response (or reply message). During the session restart procedure, the STSN command is used to inform both half sessions of the sequence number of the last sent or received sync-point message. If either half session has not completely received a given message, that message can then be retransmitted.

## Sync-Point Indicators on Messages

This section describes the sync-point indicators sent on IMS input and output messages.

### Requests on IMS Input Messages

Table 22 summarizes the response and sync-point requests for input messages to IMS.

**Related Reading:** For the description of CD and -CD, see "Bracket and Half-Duplex Protocol" on page 296.

Table 22. Response and Sync-Point Requests for IMS Input Messages

Input Message Types X = IMS supported S = Suggested	VTAM Indicators with Message							
	RQE1 <sup>1</sup>		RQD1		RQE2 <sup>1</sup>		RQD2	
	CD	-CD	CD	-CD	CD	-CD	CD	-CD
Nonlast MFS page (autopage)		S		X				
Last MFS page (autopage)								
Recoverable and irrecoverable					S		X	S
IMS transactions:								
Fast Path Conversational Recoverable Irrecoverable								
Response mode	S		X	X	S		X	X
Irrecov. nonresp., Nonconvers. mode	S	S <sup>2</sup>	X	X	S	Note 2	X	X

## Keeping Half Sessions Synchronized

Table 22. Response and Sync-Point Requests for IMS Input Messages (continued)

Input Message Types X = IMS supported S = Suggested	VTAM Indicators with Message							
	RQE1 <sup>1</sup>		RQD1		RQE2 <sup>1</sup>		RQD2	
	CD	-CD	CD	-CD	CD	-CD	CD	-CD
Recov. nonresp., Nonconvers. mode						S	X	S
MFS paging control requests								
SNA-formatted QMODEL FMHs	S							
SNA RAP FMH					S		X	
IMS message switch								
Using ATTACH SCHEDULER					S		X	S
Using ATTACH (no SCHEDULER)								Note 3
ATTACH SYSMMSG <sup>4</sup>								Note 3
IMS command	S		X	X	S		X	X
VTAM command <sup>5</sup>								
LUSTATUS - commit					S		X	S
Other	X		X	X				
Test mode <sup>6</sup>	S		X	X	S		S	X
FMH7 (ERP) messages	X	Note 3	X	Note 3				

**Notes:**

1. A response is implied by a reply to the change-direction indicator; therefore, RQE1 or RQE2 is recommended when change-direction is sent. Either method of requesting a response is supported for the session BIND option of DEFINITE RESPONSE CHAINS. When IMS is running as a secondary system, the other half session (as the primary system), if sending a BB--CD chain, must indicate RQD1 to allow proper DFC bracket and send/receive synchronization.
2. Supported only for irrecoverable-inquiry input requesting EB or BB/EB.
3. Sent with EB. BB is also sent as needed.
4. Because IMS SYSMMSG is handled in the same manner as is a message switch, the same protocols apply.
5. LUSTATUS might indicate RQE1 with EB or BB/EB. CHASE, LUSTATUS, and CANCEL commands can optionally request RQE1 with CD. LUSTATUS - commit can also request RQE2 or RQD2 for specific conditions. All other normal flow VTAM commands must request RQD1.
6. Applies to test "echo" mode only. Not applicable to /TEST MFS.

### Requests on IMS Output Messages

Table 23 summarizes the response and sync-point requests for IMS output messages

#### Related Reading:

- For information on CD and -CD, see "Bracket and Half-Duplex Protocol" on page 296.
- For information on using the abort and commit sense codes to ensure that both half sessions maintain a mutual understanding of the current sync point and the status of recoverable resources, see "LUSTATUS Protocol" on page 308.

## Keeping Half Sessions Synchronized

Table 23. Response and Sync-Point Requests for IMS Output Messages

Output Message Types X = IMS supported	VTAM Indicators with Message							
	RQE1 <sup>1</sup>		RQD1		RQE2 <sup>1</sup>		RQD2	
	CD	-CD	CD	-CD	CD	-CD	CD	-CD
Update, recov.					X			X
Inquiry, recov.					X			X
Inquiry, irrecov.	X			X				
Fast Path (recov.)								X
Nonlast MFS page (autopaged)		Note 2		Note 2				
First chain demand-paged output								
ATTACH (no SCHEDULER)	X							
ATTACH SCHEDULER								X
Nonlast MFS page (demand-page)	X							
Last MFS page (autopaged or w/o OLP) recov., irrecov.					X			X
Last MFS page (with OLP) <sup>3</sup>	X							
IMS command reply:								
/DISPLAY, /RDISPLAY, /FORMAT					X			X
/TEST	X							
Other commands				X				
Test mode output <sup>4</sup>	X			X				
Broadcast output (ATTACH SYSMMSG)					X			X
System error messages (ATTACH SYSMMSG)	X			X	X			X
Message switch output					X			X
VTAM commands <sup>5</sup>	X			X				
FMH7 (ERP) messages	X			Note 6				
QSTATUS (QMODEL) <sup>7</sup>	X				X			X

Table 23. Response and Sync-Point Requests for IMS Output Messages (continued)

Output Message Types X = IMS supported	VTAM Indicators with Message							
	RQE1 <sup>1</sup>		RQD1		RQE2 <sup>1</sup>		RQD2	
	CD	-CD	CD	-CD	CD	-CD	CD	-CD
<b>Notes:</b>								
<ol style="list-style-type: none"> <li>1. A response is implied by a reply to the change-direction indicator; therefore, IMS indicates RQE1 or RQE2 when sending the change-direction indicator. Either method of requesting a response is supported for the session BIND option DEFINITE RESPONSE CHAINS.</li> <li>2. RQD1 occurs on each chain rather than RQE1 when IMS is bound as "RQD only," and on the first page of MFS autopaged output to prevent unnecessary ERP overhead if errors or contention is detected by the receiver.</li> <li>3. MFS operator logical paging is in effect if the MOD specifies PAGE=YES and autopaged output is not indicated in the MFS system control area.</li> <li>4. Applies to test "echo" mode only. Not applicable to /TEST MFS.</li> <li>5. CHASE is always sent RQE1/CD. See "LUSTATUS Protocol" on page 308 and "CANCEL Protocol" on page 304 for the DFC bracket, send/receive, and response requirements for LUSTATUS and CANCEL, respectively.</li> <li>6. Sent with EB.</li> <li>7. CD/RQE1 is sent on QSTATUS, which results from an invalid cursor in an MFS DPM demand-page request for output sent as ATTACH SCHEDULER. CD/RQE2 or EB/RQD2 is sent on QSTATUS, which results from the receipt of QPURGE during demand-paged output. CD/RQE2 is sent: <ul style="list-style-type: none"> <li>• When operating in IMS conversation or in test mode</li> <li>• When the input is sent with ATTACH SCHEDULER and the associated output component is SINGLE2 or MULT2.</li> </ul> </li> </ol> <p>All other cases result in EB/RQD2.</p>								

## Data Flow Control Protocol Reference

This section describes the byte-level protocols for data flow control. The protocols are presented in the following order:

- "BID Protocol" on page 295
- "Bracket and Half-Duplex Protocol" on page 296
- "CANCEL Protocol" on page 304
- "Chaining Protocol" on page 304
- "CHASE Protocol" on page 305
- "ERP Purging" on page 305
- "LUSTATUS Protocol" on page 308
- "Paged Messages ERP" on page 311
- "Ready-to-Receive Protocol" on page 312
- "RSHUT Protocol" on page 312
- "Selective Receiver ERP" on page 312
- "Sender ERP" on page 316
- "Sense Codes that IMS Receives" on page 319
- "Sense Codes that IMS Sends" on page 319
- "SIGNAL Protocol" on page 320
- "Symmetrical Session Shutdown for LU 6.1 (SBI and BIS)" on page 320

### BID Protocol

IMS does not send the SNA BID command, but receives the command when acting as a secondary half session.

## BID Protocol

When receiving the BID command in a between-bracket state, IMS responds DR1 and enters a receive state for the pending input. If the BID command is received while IMS is in-brackets, IMS rejects the BID with an exception DR1 response (X'08130000').

## Bracket and Half-Duplex Protocol

IMS uses SNA bracket protocol to resolve contention, and uses the change-direction indicator of the half-duplex protocol to control normal flow send/receive mode while within bracket state.

IMS uses the following options when in session with another ISC logical unit:

- Bracket reset state can be selected at data traffic active state. This state can be set to between-brackets (BETB) or in-brackets (INB).
- Half-duplex send or receive state can be selected at data traffic active state.
- Either half session can send end-bracket (EB). IMS must always be bound to allow EB.
- Half-duplex flip-flop is used for normal flow send/receive states.
- Bracket termination rule 1 (conditional rule) is used.

### Related Reading:

- For more information on session bind parameters, see “Appendix B. Bind Parameters for SLU P and LU 6.1” on page 471.
- For information on bracket termination rules, see *Systems Network Architecture Format and Protocol Reference: Architectural Logic*.

### **Bracket Protocol for IMS Input**

One or more input messages can be received within a bracket as described below:

- Any number of nonresponse, nonconversation, noncommand, or nontest mode input messages can be sent within the bracket. The input, the subsequent asynchronous output, or an LUSTATUS can end the bracket. Further, a single nonresponse, nonconversation, noncommand, or nontest mode input message with ATTACH EB can be sent.
- One response mode or command message, except /TEST, can optionally occur after any of the above within the same bracket. The response mode or command message output ends the bracket.
- Any number of conversation or test mode inputs followed by conversation or test mode outputs can optionally begin after the input messages listed in the first bullet above within the same bracket. The bracket ends on the last conversation output message (or command complete message from /END for test mode). The bracket can also be ended by the other half session's sending an input LUSTATUS or CHASE with EB, but not by input data sent with EB.
- The bracket can be ended by an EB occurring on an FMH7 or LUSTATUS resulting from an error.

Table 24 on page 297 summarizes the bracket and send/receive indicators acceptable for the various message types input to IMS.



## Bracket and Half-Duplex Protocol

Table 24. Bracket and Send/Receive Indicators for Messages Sent to IMS

VTAM Indicators <sup>1</sup> Sent with Message						
X = IMS supported	BB	EB	BB/EB	BB/CD	CD	-BB, -EB, -CD
<b>Input Message Type Using ATTACH SCHEDULER:</b> MFS-autopaged input						
First page (of multiple page input)	X					X
Nonfirst, nonlast page						X
Last page (see tran. types below)						
IMS transaction: Nonresponse						
Nonconvers. mode	Note 2	Note 3	Notes 2, 3	Note 2	X	X
Response mode (Including Fast Path)	Note 2			Notes 2, 4	Note 4	X
Conversational	Note 2			Notes 2, 4	Note 4	X
IMS message switch	X	X	X	X	X	X
IMS command	X	Note 5	Note 5	Note 4	Note 4	X
Input while in test mode <sup>6</sup>					Note 4	X
<b>Input Message Type Using ATTACH (no SCHEDULER):</b> MFS-autopaged input						
First page (of multipage input)	X					X
Nonfirst, nonlast page						X
Last page (see tran. types below)						
IMS transaction: Nonresponse, Nonconvers. mode						
Response mode (Including Fast Path) <sup>7</sup>	Note 2			Notes 2, 4	Note 4	X

## Bracket and Half-Duplex Protocol

Table 24. Bracket and Send/Receive Indicators for Messages Sent to IMS (continued)

VTAM Indicators <sup>1</sup> Sent with Message						
X = IMS supported	BB	EB	BB/EB	BB/CD	CD	-BB,-EB, -CD
Conversational	Note 2			Notes 2, 4	Note 4	X
IMS message switch		X	X			
ATTACH SYSMMSG		X	X			
IMS command	X	Note 5	Note 5	Note 4	Note 4	X
Input while in test mode <sup>6</sup>					Note 4	X
<b>Other Input Message Types:</b>						
VTAM normal flow command/ indicator <sup>8</sup> .	X				X	X
FMH7 (ERP) messages		Note 9	X	X	X	
IMS MFS paging control request (QMODEL FMHs)				X	X	
RAP FMH					X	

Table 24. Bracket and Send/Receive Indicators for Messages Sent to IMS (continued)

VTAM Indicators <sup>1</sup> Sent with Message						
X = IMS supported	BB	EB	BB/EB	BB/CD	CD	-BB, -EB, -CD
<b>Notes:</b>						
<ol style="list-style-type: none"> <li>Symbols: <ul style="list-style-type: none"> <li>BB: Begin-Bracket</li> <li>EB: End-Bracket</li> <li>CD: Change-Direction. Allowed on Last- or Only-in-Chain.</li> <li>BB/EB: Begin- and End-Bracket. Allowed on First- or Only-in-Chain.</li> </ul> </li> <li>Not valid for the last page of MFS-autopaged input.</li> <li>Not valid for the first page of MFS-autopaged input.</li> <li>Denotes the optimal end-of-message indicators to prevent IMS from soliciting change-direction using SNA SIGNAL.</li> <li>Valid only for /DIS, /RDIS, and /FOR commands. Also optimal for these commands to keep IMS from immediately forcing a between-brackets state by sending LUSTATUS - NO-OP (X'0006') with EB. The output from these commands is always queued and sent asynchronously.</li> <li>Applies to test "echo" mode only. Not applicable to /TEST MFS.</li> <li>For ISC, response mode operation is forced for attached transactions not indicating EB (regardless of their definition on the TRANSACT macro statement) when the TERMINAL macro statement or ETO user descriptor indicates either TRANRESP or FORCRESP. The input is rejected if the TERMINAL macro statement or ETO user descriptor indicates NORESP.</li> <li>LUSTATUS, CHASE, and CANCEL can indicate EB or CD. For information on these command protocols, see "Data Flow Control Protocol Reference" on page 295. Other VTAM normal flow commands or indicators must not indicate either EB or CD.  When IMS is waiting for conversational input, LUSTATUS and CHASE cannot be sent to IMS unless they also indicate end-bracket. EB causes the conversational output message to be dequeued, the conversation mode to be terminated, and the Conversational Abnormal Termination exit routine to be invoked. See the description of LUSTATUS and "Handling IMS Response Mode or Conversational Output Errors" on page 285. The session is terminated if an LUSTATUS or CHASE is received by IMS with any other protocol.</li> <li>FMH7 messages and LUSTATUS NO-OP indicating EB force end of conversation, response mode, and test mode and cause the associated IMS output message to be dequeued. Further, the Conversational Abnormal Termination exit routine is invoked. See the description of LUSTATUS and "Handling IMS Response Mode or Conversational Output Errors" on page 285.</li> </ol>						

### Bracket Protocol for IMS Output

The output bracket and send/receive protocol used by IMS and the number of output messages sent per bracket depend upon a combination of one or more of the following:

- The IMS message type
- Whether the message is to be MFS paged
- Whether the other half session supports the SCHEDULER model
- The output component specified during IMS system definition
- The bracket and ATTACH protocol associated with the originating input transaction

Within IMS, the protocols used for nonlast chains (pages) of MFS-paged output and the last (MFS-paged) or only chain of response mode, conversation mode, test mode, or command replies are predefined regardless of whether the originating input transaction and resulting output replies are synchronous (ATTACH) or asynchronous (SCHEDULER). This is also true for output resulting from input sent

## Bracket and Half-Duplex Protocol

with ATTACH EB and for asynchronous output that must be sent ATTACH because the other half session lacks SCHEDULER support.

CD is used for all nonlast chains of MFS demand-paged messages, the last (MFS-paged) or only chain of test mode, and nonlast conversational mode messages. CD allows the paging operation or synchronous event between IMS and the other half session to continue.

EB ensures that the end of the synchronous event occurs by placing the session in reset state for the last (MFS-paged) or only chain of response mode replies, last conversational mode replies, and command replies. This is also true for replies resulting from input sent with ATTACH EB and for asynchronous output that must be sent ATTACH because the other half session lacks SCHEDULER support. All nonlast chains (pages) of MFS-autopaged messages are sent without either CD or EB.

Using ATTACH SCHEDULER, your installation must define to IMS the protocols to be used for the last (MFS-paged) or only chain of other asynchronous output. IMS allows you to specify single or multiple messages for each component defined for a session and whether IMS should send more than one asynchronous message before indicating change-direction or end-bracket. The possible component definitions are:

### **SINGLE1**

Asynchronous output for this component is sent one message per bracket. Each message begins a bracket (if necessary) and always ends a bracket.

### **SINGLE2**

Asynchronous output for this component is sent one message at a time with the VTAM begin-bracket (if necessary) and change-direction indicators to allow the receiving subsystem to optionally send its communication traffic.

### **MULT1**

All asynchronous messages for a given LTERM are sent before the bracket is ended. Traffic is sent BB (if necessary), message1, message2,...messageN, EB. EB occurs after the last message for the LTERM is acknowledged and dequeued.

### **MULT2**

All asynchronous messages for a given LTERM are sent before change-direction is sent. Traffic is sent BB (if necessary), message1, message2...messageN, CD. CD occurs after the last message for the LTERM is acknowledged and dequeued.

Consider the definition (on the IMS system definition TERMINAL macro statement or an ETO logon descriptor), and use of protocols for ISC components to be used for transmission of asynchronous messages. This is particularly true in an IMS-to-IMS environment, because most messages between the two subsystems are asynchronous. Incorrectly defining or using these ISC protocols can:

- Require extra transmissions to occur in order to acquire the flow or to end the bracket
- Cause unnecessary bracket contention error recovery operations
- Produce output protocols unacceptable to the receiving subsystem.

A further explanation follows:

**SINGLE1**

Use of SINGLE1 results in EB on each message and might cause bracket contention if the other subsystem has data to send.

EB on each message can cause process errors within the other subsystem if the transaction must reply synchronously within the initiating bracket. An error of this type can occur in the receiving subsystem when a SINGLE1 component is used to send response mode, conversation, or test mode messages, or IMS commands from one IMS subsystem to another.

**SINGLE2 or MULT2**

Change-direction sent on messages from a component defined as SINGLE2 might result in unnecessary transmissions to end the bracket when either no output is available or no reply is available within the other subsystem.

**MULT1 or MULT2**

Messages sent on components defined as MULT1 OR MULT2 indicate change-direction or end-bracket only after the last message on a queue has been dequeued. Therefore, extra transmissions might result if the other subsystem must signal for the flow to return synchronous replies.

For components defined as MULT1 and MULT2, IMS suppresses the queue rotation that normally occurs between output messages. This suppression allows all messages from a queue selected for output to be sent before IMS initiates output on other queues. Selection of the next queue with available output occurs when the previous queue has emptied or when input changes the active output queue. The queues are not rotated if an LUSTATUS - queue empty or another input that indicates change-direction occurs for the same queue.

Table 25 summarizes the bracket and send/receive indicators that are acceptable for IMS output messages.

Table 25. Bracket and Send/Receive Indicators for Output Messages

Output Message Type Sent Using Attach Scheduler:	VTAM Indicators <sup>1</sup> Sent with Message		
	SINGLE1 msg.init INB BETB	SINGLE2 msg.init INB BETB	MULT1/2 <sup>2</sup> msg.init INB BETB
MFS demand-paged output:			
First SNA chain			
(ATTACH SCHEDULER) <sup>3</sup>	EB BB/EB	EB BB/EB	EB BB/EB
Nonlast page			
last page if OLP <sup>4</sup>	CD	CD	CD
Last page without OLP (See appropriate message type)			
MFS autopaged output:			
First page	N BB	N BB	N BB
Nonfirst, nonlast page	N	N	N
Last page (See appropriate message type)			
Response mode output (Including Fast Path)	EB	EB	EB
Conversational: Nonlast conversational			

## Bracket and Half-Duplex Protocol

Table 25. Bracket and Send/Receive Indicators for Output Messages (continued)

VTAM Indicators <sup>1</sup> Sent with Message			
Output Message Type Sent Using Attach Scheduler:	SINGLE1 msg.init INB BETB	SINGLE2 msg.init INB BETB	MULT1/2 <sup>2</sup> msg.init INB BETB
output message	CD	CD	CD
Last conversational			
output message	EB	EB	EB
Message switch output	EB BB/EB	CD BB/CD	N BB
Command output			
/FOR, /DIS, /RDIS	EB BB/EB	CD BB/CD	N BB
/TEST <sup>5</sup>	CD	CD	CD
Other	EB	EB	EB
Test mode output <sup>5</sup>	CD	CD	CD
None of the above message types - asynchronous	EB BB/EB	CD BB/CD	N BB
<b>Output Message Type Sent Using ATTACH (no SCHEDULER):</b>			
MFS demand-paged output:			
First SNA chain (ATTACH)	CD	CD	CD
Nonlast page, last page if OLP <sup>4</sup>	CD	CD	CD
Last page without OLP (See appropriate message type)			
MFS autopaged output:			
First page	N	N	N
Nonfirst, nonlast page	N	N	N
Last page (See appropriate message type)			
Response mode output (Including Fast Path) <sup>6</sup>	EB	EB	EB
Conversational:			
Nonlast conversational output message	CD	CD	CD
Last conversational output message	EB	EB	EB
Nonresponse, nonconversation output			
(No SCHEDULER model defined for other half session)	EB BB/EB	EB BB/EB	EB BB/EB
ATTACH SYMSMSG	BB/EB	BB/CD	BB
Command output			
/TEST <sup>5</sup>	CD	CD	CD
Other (Except /FOR, /DIS, /RDIS) <sup>7</sup>	EB	EB	EB
Test mode output <sup>5</sup>	CD	CD	CD
<b>Other Output Message Types:</b>			

## Bracket and Half-Duplex Protocol

Table 25. Bracket and Send/Receive Indicators for Output Messages (continued)

VTAM Indicators <sup>1</sup> Sent with Message			
Output Message Type Sent Using Attach Scheduler:	SINGLE1 msg.init	SINGLE2 msg.init	MULT1/2 <sup>2</sup> msg.init
	INB BETB	INB BETB	INB BETB
VTAM command/indicator <sup>8</sup>			
FMH7 (ERP) messages:			
During conversation	CD	CD	CD
During test mode	CD	CD	CD
Other <sup>9</sup>	EB BB/EB	CD BB/CD	Note 10
QSTATUS (QMODEL)	Note 11	Note 11	Note 11

**Notes:**

1. Symbols:
  - INB** Indicates that synchronous or asynchronous output occurs within the same bracket as the previous input
  - BETB** Indicates that IMS initiates output while in a between-brackets state
  - BB** Begin-bracket
  - EB** End-bracket
  - CD** Change-direction (allowed on last- or only-in-chain)
  - N** No bracket or send/receive indicators
2. LUSTATUS - queue empty is used at the end of a queue to send EB for MULT1 and CD for MULT2. EB is sent when an ERP backout has reset the DFC and ATTACH states to between-brackets and no component 1 (COMPT1) is defined during IMS system definition. Also, the BB indicator in the BETB column occurs only for the first message or MFS page that must initiate an output bracket asynchronously.
3. Although IMS is between-brackets after having sent stand-alone ATTACH SCHEDULER for response mode, Fast Path, or conversational demand-paged output, IMS does not accept input until the preceding output is successfully transmitted and dequeued using the appropriate paging requests.
4. MFS operator logical paging (OLP) is in effect if the MOD specifies PAGE=YES and autopaged output is not indicated in the MFS system control area (SCA).
5. Applies to test "echo" mode only. Not applicable to /TEST MFS.
6. Response mode operation always occurs for input transactions that are directly attached.
7. See "LUSTATUS Protocol" on page 308 for more information on /DISPLAY, /RDISPLAY, and /FORMAT command output.
8. BIS is sent without CD or EB. CHASE is always sent CD. See "LUSTATUS Protocol" on page 308 and "CANCEL Protocol" on page 304 for the DFC bracket, send/receive, and response requirements for LUSTATUS and CANCEL, respectively.
9. An FMH7 requests the protocol associated with the resulting between-brackets reset state of the last committed input component after backout of the DFC and ATTACH states due to the ERP operation.
10. FMH7 is sent with EB for MULT1, CD for MULT2.
11. CD/RQE1 is sent on a QSTATUS that results from an invalid cursor in an MFS DPM demand-page request for output. CD/RQE2 or EB/RQD2 is sent on QSTATUS, which results from the receipt of QPURGE during demand-paged output. CD/RQE2 is sent: 1) when in IMS conversation or test mode or 2) when the input is sent with ATTACH SCHEDULER and the associated output component was SINGLE2 or MULT2. All other cases result in EB/RQD2.

**Related Reading:** For examples of nonpaged and MFS bracket and half-duplex protocol, see "Appendix H. ISC Data Flow Control Examples" on page 539.

## CANCEL Protocol

### CANCEL Protocol

IMS sends and receives the SNA CANCEL command. CANCEL allows the sender to terminate an output SNA chain without having to send all of the chain following a sender- or receiver-detected error.

The CANCEL command sent by IMS because of either a sender- or receiver-detected error is sent with the following VTAM indicators:

Output (Determined in Following Order):	VTAM Indicators with CANCEL		
	-CD -EB	CD	EB
After /DEQ (LUSTATUS ABORT is next)	X		
When primary half session and FIC/OIC was between bracket (conditional BB)	X		
After receiving selective receiver ERP sense code		X	
After receiving SIGNAL		X	
When conversational or response mode output is still available	X		
FIC/OIC was EB RQD* (conditional EB)			X
While in test (echo) mode		X	
Non-MFS demand-paged output for SINGLE1 component (ref. FIC/OIC either between bracket or EB above)			
MFS demand-paged output for SINGLE1 component			X
Output for SINGLE2 component		X	
Output for MULT1 or MULT2 component	X		

The CANCEL command received by IMS because of either a sender- or receiver-detected error can indicate whatever protocol (consistent with current DFC states) is appropriate for the message sender.

## Chaining Protocol

Whether operating as the primary or secondary half session, IMS sends both single and multiple RU chains. Half sessions can operate with either single or multiple RU chains as specified in the bind parameters.

IMS messages are usually sent and received as single SNA chains where only-in-chain (OIC) or first-in-chain (FIC) indicates beginning-of-message, and OIC or last-in-chain (LIC) indicates end-of-message. However, each page of an MFS-paged input or output message is sent or received as a single SNA chain. The beginning-of-message occurs when the first page is sent or received, and the end-of-message occurs at sync point (RQD2 or RQE2, CD) requested at the end of the last input or output page (except for MFS operator logical paging).

Optionally, end-of-message can be requested using LUSTATUS - commit immediately following the last page of MFS DPM-autopaged input.

Use the CANCEL command when errors are detected on multiple RU input chains.



**Related Reading:**

- For more information on LUSTATUS - commit, see “LUSTATUS Protocol” on page 308.
- For more information on multichain purging, see “ERP Purging”.

**CHASE Protocol**

IMS sends and receives SNA CHASE. IMS sends CHASE requesting RQE1/CD to cause a synchronizing event after the receipt of an exception response when it is between pages (chains) of MFS-autopaged output and is not bound with Definite Response Chains Only.

IMS responds with DR1, as necessary, when receiving CHASE. IMS can receive CHASE between any two messages within a bracket. IMS can also receive CHASE as a synchronizing event after IMS has sent an exception response to MFS-autopaged input. (The sender of the MFS-autopaged input receives the exception response while between output pages or after a chain indicating RQE1.)

Only CHASE carrying EB is accepted while IMS is waiting for conversational input. EB causes the conversational output message to be dequeued, conversation mode to be terminated, and the Conversational Abnormal Termination exit routine to be invoked, just as if an /EXIT command had been received. The session is terminated if CHASE is received with any protocol other than EB.

**ERP Purging**

After sending an exception response and before continuing to send or receive, the exception response sender might need to enter error recovery process (ERP) PURGE mode until the DFC state managers of both half sessions are synchronized. Additionally, ERP PURGE must be complete prior to sending the FMH7 ERP message. ERP purging can occur for either single or multiple SNA chains. Single chain purge occurs when DFC states are synchronized within the same chain receiving the exception response. Multichain purge occurs when more than a single SNA chain must be purged before the DFC state managers are synchronized. (The CANCEL and CHASE commands are logically considered part of the same chain when they occur immediately after the SNA chain that resulted in the exception response.)

Single- and multiple-chain purging occur within the data flow control support layer and are independent of SNA presentation layer functions such as MFS.

ERP PURGE must be entered when a half session sends an exception response to an RU (OIC, MIC, or LIC) that does not result in ending a bracket or in both half-session DFC states being synchronized. That is, ERP PURGE begins when a half session sends an exception response to the following RUs:

- FIC RQE\* (not applicable for EB)
- MIC RQE\* (not applicable for CD or EB)
- LIC RQE\* -EB and -CD
- OIC RQE\* -EB and -CD

ERP PURGE ends when the purging half session receives an RU that either ends the bracket or causes both half-session DFC states to again be synchronized. That is, ERP PURGE terminates when the purging half session receives the following RUs:

## ERP Purging

**OIC RQD\*** This includes the CANCEL and CHASE commands, and is not applicable for CD or EB.

**OIC RQE\* CD or EB**  
This includes the CANCEL command.

**LIC RQD\*** Not applicable for CD or EB.

**LIC RQE\* CD or EB**

ERP PURGE can span several SNA chains before terminating. When more than one chain is purged, an additional exception response is necessary at the point of bracket termination or DFC state synchronization (except for CANCEL or CHASE commands) to prevent an ambiguous or incorrect understanding as to the disposition of the message by the sending half session (response receiver). This sender ERP exception response (X'0867') indicates that the chain was purged because of an error on a previous chain and that the message sender should resend the message at the next possible opportunity. The X'0867' exception response must be sent when a half session ends ERP PURGE after receiving the following RUs in a chain subsequent to the one that resulted in the original exception response:

**OIC RQD\*** This does not include CANCEL or CHASE, and is not applicable for CD or EB.

**OIC RQE\* CD or EB** Except for CANCEL

**LIC RQD\*** Not applicable for CD or EB.

**LIC RQE\* CD or EB**

The DFC protocol that can be received with the FMH7 ERP message must be either change-direction or end-bracket and can include begin-bracket, as appropriate. The DFC protocol (CD or EB) that is sent by IMS with the FMH7 ERP message is determined by the ATTDSP value resulting after ERP backout. When backout results in a between-brackets state, the ATTDSP value is component 1. When backout results in a state other than between-brackets, the ATTDSP value is the last committed input component. The protocols are set for the resulting component. End-bracket is sent when an ERP backout has reset the DFC and ATTACH states to between-brackets and no component 1 (COMPT1) was defined during IMS system definition.

**Related Reading:** For information on the protocols that are set for the resulting component, see "Bracket Protocol for IMS Output" on page 299.

### Resulting DFC State after Sender ERP Purge

Table 26 reflects all valid bracket and send/receive states that result after both half sessions reach a sync point after an exception response that indicates a sender ERP sense code other than selective receiver ERP.

Table 26. Resulting DFC States after Sender ERP Purge

Type of Chain in Error	PHS Sends Data, SHS Sends Exception: Half-Session States		SHS Sends Data, PHS Sends Exception: Half-Session States	
	PHS	SHS	PHS	SHS
BB/— FIC				
RQ** CD OIC,LIC	1	1	INB.SEND	INB.RCV

Table 26. Resulting DFC States after Sender ERP Purge (continued)

Type of Chain in Error	PHS Sends Data, SHS Sends Exception: Half-Session States		SHS Sends Data, PHS Sends Exception: Half-Session States	
	PHS	SHS	PHS	SHS
RQD* OIC,LIC	BETB	BETB	INB.RCV	INB.SEND
RQD* CANCEL	BETB	BETB	INB.RCV	INB.SEND
RQ** CD CANCEL	1	1	INB.SEND	INB.RCV
RQD* EB CANCEL	1	1	BETB	BETB
BB/EB FIC				
RQE* OIC,LIC	BETB	BETB	BETB	BETB
RQD* OIC,LIC	BETB	BETB	BETB	BETB
RQD* CANCEL	BETB	BETB	BETB	BETB
RQ** CD CANCEL <sup>1</sup>				
RQD* EB CANCEL <sup>1</sup>				
--/ FIC				
RQE* OIC,LIC	INB.SEND	INB.RCV	INB.RCV	INB.SEND
RQ** CD OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* OIC,LIC	INB.SEND	INB.RCV	INB.RCV	INB.SEND
RQD* CANCEL	INB.SEND	INB.RCV	INB.RCV	INB.SEND
RQ** CD CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* EB CANCEL	BETB	BETB	BETB	BETB
--/EB FIC				
RQ** CD OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* OIC,LIC	INB.SEND	INB.RCV	INB.RCV	INB.SEND
RQD* CANCEL	INB.SEND	INB.RCV	INB.RCV	INB.SEND
RQ** CD CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* EB CANCEL	BETB	BETB	BETB	BETB
<b>Where:</b>				
BETB: between-brackets				
INB : in-brackets				
PHS : primary half session				
SHS : secondary half session				
<sup>1</sup> CD and EB might not be sent while in DFC BETB state.				

### Resulting DFC State after Selective Receiver ERP Purge

Table 27 reflects all valid bracket and send/receive states that result after both half sessions reach a sync point after an exception response that indicates the SNA selective receiver ERP sense code, and before the FMH7 is sent by the half session detecting the error.

Table 27. Resulting DFC States after Selective Receiver ERP Purge

Type of Chain in Error	PHS Sends Data, SHS Sends Exception: Half-Session States		SHS Sends Data, PHS Sends Exception: Half-Session States	
	PHS	SHS	PHS	SHS
BB/-- FIC				

## ERP Purging

Table 27. Resulting DFC States after Selective Receiver ERP Purge (continued)

Type of Chain in Error	PHS Sends Data, SHS Sends Exception: Half-Session States		SHS Sends Data, PHS Sends Exception: Half-Session States	
RQ** CD OIC,LIC	1	1	INB.SEND	INB.RCV
RQD* OIC,LIC	ERP.RCV	ERP.SEND	INB.SEND	INB.RCV
RQD* CANCEL	ERP.RCV	ERP.SEND	INB.SEND	INB.RCV
RQ** CD CANCEL	1	1	INB.SEND	INB.RCV
RQD* EB CANCEL	1	1	ERP.SEND	ERP.RCV
BB/EB FIC				
RQE* OIC,LIC	ERP.RCV	ERP.SEND	ERP.SEND	ERP.SEND
RQD* OIC,LIC	ERP.RCV	ERP.SEND	ERP.SEND	ERP.SEND
RQD* CANCEL	ERP.RCV	ERP.SEND	ERP.SEND	ERP.SEND
RQ** CD CANCEL <sup>1</sup>				
RQD* EB CANCEL <sup>1</sup>				
--/ FIC				
RQE* OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQ** CD OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQ** CD CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* EB CANCEL	ERP.RCV	ERP.SEND	ERP.SEND	ERP.RCV
--/EB FIC				
RQ** CD OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* OIC,LIC	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQ** CD CANCEL	INB.RCV	INB.SEND	INB.SEND	INB.RCV
RQD* EB CANCEL	ERP.RCV	ERP.SEND	ERP.SEND	ERP.RCV

**Note:** <sup>1</sup> CD and EB cannot be sent while in between-brackets state

## LUSTATUS Protocol

IMS sends and receives LUSTATUS as summarized in Table 28. The DFC bracket, send/receive, and response requirements illustrated in this figure are subject to the considerations detailed in the balance of this section. Any LUSTATUS sense code not listed in this figure causes the ISC session to be terminated.

Table 28. VTAM Indicators Sent with LUSTATUS

LUSTATUS-	VTAM Indicators with LUSTATUS								
	RQE1		RQD1			RQE2	RQD2		
	EB	CD	--	CD	EB	CD	--	CD	EB
X = IMS Supported S = Suggested									
Sense Codes Received <sup>1</sup>									
Commit-X'0006'						S	S	X	S

Table 28. VTAM Indicators Sent with LUSTATUS (continued)

LUSTATUS-	VTAM Indicators with LUSTATUS								
	RQE1		RQD1			RQE2		RQD2	
	EB	CD	--	CD	EB	CD	--	CD	EB
NO-OP-X'0006'	X	S		X	S				
Queue Empty-X'0007'		S		X	S				
Function Abort									
-X'0864'		S	S	X	S				
-X'0865'		S	S	X	S				
-X'0866'		X	X	X	X				
Sense Codes Sent									
Commit-X'0006'									
NO-OP-X'0006'		X			X				
Queue Empty-X'0007'		X			X				
Function Abort									
-X'0865'		X	X		X				

**Notes:**

1. While IMS is waiting for conversational input, it accepts LUSTATUS carrying EB only. EB causes the conversational output message to be dequeued, conversation mode to be terminated, and the Conversational Abnormal Termination exit routine to be scheduled. The session is terminated if an LUSTATUS carrying any other protocol is received.

IMS responds DR1 or DR2 when receiving LUSTATUS RQD1 or RQD2. IMS requests DR1, DR2, or exception DR1 or DR2 as indicated in Table 28 on page 308. The session is terminated if an exception occurs to LUSTATUS.

- LUSTATUS - Queue Empty.

IMS sends and receives LUSTATUS - queue empty.

IMS sends LUSTATUS - queue empty:

- After receiving an asynchronous (ATTACH SCHEDULER) input message and having no output immediately available.
- After sending all available output on a given queue.

RQD1 and end-bracket are indicated on the LUSTATUS sent when IMS is not in conversation or response mode and has been left in-brackets/SEND, and no output is available to be sent from a component defined as SINGLE1 or MULT1. RQE1 and change-direction are indicated on the LUSTATUS sent when IMS is not in conversation or response mode and has been left in-brackets/SEND, and no output is available to be sent from a component defined as SINGLE2 or MULT2.

- LUSTATUS - Function Abort. The supported sense codes are:

**X'0864'**

Loop occurs upon re-execution. Sender should not resend the same data.

**X'0865'**

Data sender is responsible for detecting and preventing loop.

**X'0866'**

Data receiver is responsible for detecting and preventing loop.

## LUSTATUS Protocol

IMS sends LUSTATUS - function abort X'0865' only under the following conditions:

- When an IMS /DEQUEUE command has been issued by an authorized IMS terminal operator before the last page of an MFS-demand or autopaged output message has been sent by IMS.

The DFC bracket and send/receive protocol indicated on the LUSTATUS is the same as that which would occur with the last chain (page) had the message been successfully sent.

The following occurs when IMS receives LUSTATUS - function abort:

- Except during conversational mode, any function abort received after an output message (including the last page of MFS demand-paged or autopaged output) sent RQE/CD causes the output message to be committed and does not cause session termination. LUSTATUS that is processed as a normal flow input before it is processed as an LUSTATUS command also causes the message to be committed.
- Any function abort causes an incomplete input MFS-autopaged message to be discarded.
- Function abort X'0864', received before the last page of an output MFS demand-paged message is sent, causes the message to be dequeued prior to continuing normal input/output operations if the output is conversational and demand-paged. The Conversational Abnormal Termination exit routine is invoked, just as if an /EXIT command had been received on the session.
- Function abort X'0865', received before the last page of an active output MFS demand-paged message is sent, causes the message to be returned to the output message queue and the session to be terminated.
- Function abort X'0866', received before the last page of an active output MFS demand-paged message, causes the message to be returned to the message queue and made available for retransmission before normal input/output operations continue.
- Any function abort resets the ATTACH states to those in effect at the last sync point.

LUSTATUS - function abort might never be sent to IMS in reply to any RU indicating RQD2 or RQE2, or the session is terminated.

- LUSTATUS - Commit

IMS does not send LUSTATUS - commit. IMS receives LUSTATUS - commit as an end-of-message indication immediately following the last page of autopaged input and with EB while awaiting conversational input. This creates a “normal” end of the conversation by invoking the Conversational Abnormal Termination exit routine with a new input vector of X'28'. This allows the exit routine to schedule a transaction to commit pending resources reflected in the scratch pad area (SPA).

**Related Reading:** For more information on normal and abnormal termination of conversations, see “Handling IMS Response Mode or Conversational Output Errors” on page 285.

- LUSTATUS - NO-OP

IMS sends and receives LUSTATUS - NO-OP.

IMS sends LUSTATUS - NO-OP:

- To allow RQE1 and change-direction (CD) to flow after SIGNAL RCD.

- To end the bracket after receiving a synchronous input message that generates no output. In IMS, this only occurs for the `FORMAT`, `/RDISPLAY`, and `/DISPLAY` commands when attached synchronously. The `LUSTATUS - NO-OP` is sent indicating `RQD1` and `EB`.
- To end the bracket at those times when bound in-brackets/`SEND` and no synchronous output or restart is possible. That is, IMS is not in conversation or response mode after session restart. The `LUSTATUS` is sent, indicating `RQD1` and `EB`.
- To allow required input when bound in-brackets/`SEND` at session restart and a pending conversational message is dequeued using `STSN` protocol. `LUSTATUS` is sent indicating `RQE1` and `CD`.
- To end the bracket after receiving exception or `ERP FMH7` (without `EB`) sense code `X'0864'`, following response mode or the last output message of an IMS conversation.

`LUSTATUS - NO-OP` can be received during asynchronous or synchronous input processing and might indicate either `CD` or `EB`, as necessary. Receipt of `LUSTATUS EB` during an IMS conversation or response mode or while in test mode causes termination of the IMS conversation, response, or test mode and causes the associated output message to be dequeued. For conversation, the Conversational Abnormal Termination exit routine is invoked just as if an `/EXIT` command had been received.

When IMS is the secondary half session, it receives a special case of `LUSTATUS BB/EB RQ*1` after responding with `DR1` to a `BID` command sent by the primary half session. This `LUSTATUS` removes IMS (secondary) from a `DFC` in-brackets/pending state that was set after the aforementioned `DR1` was sent and the primary half session has no other message available to be sent. This condition might occur when, for example, the application within the primary half session that caused the `BID` to be sent might have been abnormally terminated during the time between sending the `BID` and receiving the `BID` response.

`LUSTATUS-CD` should not be followed by another `LUSTATUS-CD`. Doing so creates a back-and-forth effect between the half sessions. IMS always responds `LUSTATUS-EB` when receiving `LUSTATUS-CD` and no output is available to be sent.

## Paged Messages ERP

For sender-detected errors (`/DEQ` command) detected after transmission of an IMS multichain demand or autopaged output message has been initiated, IMS must send the `CANCEL` command (as necessary) in conjunction with `LUSTATUS - abort`. The `CANCEL` command is used to terminate a single multi-RU chain, or page, while the `LUSTATUS` is used to terminate the process receiving the multichain, or paged, message. If IMS receives a `CANCEL` during an input autopaged message without receiving the subsequent `LUSTATUS - abort` on the next input RU, IMS returns an exception response indicating selective receiver `ERP` followed by an `FMH7` with the abort sense code `X'0865'` and an appropriate `ERP` message. Only the `LUSTATUS - abort` need be sent or received when the error is detected while between chains.

Errors detected on the first page of an autopaged input message result in IMS either sending a contention sense code or a selective receiver `ERP` sense code followed by an `FMH7` indicating one of the nonfunction abort sense codes and an appropriate error message. Errors detected on nonfirst pages of an autopaged input message or for input page request received during an IMS demand-paged output message result in IMS sending the selective receiver `ERP` sense code followed by

## Paged Messages ERP

an FMH7 indicating the X'08650000' function abort sense code and an appropriate error message. The function abort sense code is used in these cases to cause the sending process (of autopaged input or demand-paged requests), rather than just a single chain or page, to be terminated.

## Ready-to-Receive Protocol

IMS does not send the SNA ready-to-receive (RTR) command, but receives the command when acting as a primary half session.

When receiving the RTR command, IMS responds either with a DR1, followed by an available output message, or with an exception DR1 response (X'08190000'), if output is either not available or cannot be sent.

## RSHUT Protocol

IMS does not send, but does receive, the RSHUT command. When receiving RSHUT, IMS sends a DR1 response and schedules termination of the session at the next convenient point. The session is normally terminated at the end of the current input or output chain.

## Selective Receiver ERP

Selective receiver error recovery procedure (ERP) is initiated using the X'08460000' sender ERP sense code. The procedure has the following characteristics:

- The procedure is symmetrical to both half sessions.
- ERP messages can be sent by the sender of the exception response indicating selective receiver ERP. An ERP message is an ERP FMH7 header followed by an IMS error message. Each selective receiver ERP message is preceded by an ERP FMH7 function management header.
- Contention for ERP messages does not occur for the exception response sender.
- ERP messages can consist of a full chain of arbitrary length, but must immediately follow the exception response indicating selective receiver ERP.
- The ERP message is recognized by the receiver for special processing.
- The response sender has an opportunity to reset the function management level bracket and send/receive states.

The ERP messages sent by IMS have associated default IMS message output descriptors (MODs) and display output formats (DOFs). The error message is formatted using SNA character string (SCS) controls even when sent to a component defined with MFS.

**Related Reading:** For more information on SNA character sets, see “Appendix E. SNA Character String Controls” on page 487.

If necessary, the sender of the exception response enters into an ERP PURGE mode until both half sessions' send/receive states are again synchronized. The receiver of the exception response must cause the resynchronizing event.

**Related Reading:**

- For information on the ERP FMH7 function management header, see “Error Recovery Procedure FM Header” on page 324.
- For information on the ERP PURGE mode, see “ERP Purging” on page 305.



- For information on the resynchronizing event that the receiver of the exception response must start, see “Sender ERP” on page 316.

**Selective Receiver ERP Sense Codes Supported**

During output, IMS recognizes receipt of the SNA-defined selective receiver ERP system sense code when assuming either half-session role, returns the message to the queue, ensures that a synchronizing event occurs to end the other half session’s ERP purge cycle, and enters a receive state that can be satisfied only by input (ERP message) from the other half session.

When receiving this ERP message, IMS takes one of three actions based on the FMH7 sense code and the DFC protocols associated with the received ERP message:

- The ERP message (DFS2083) is routed to the IMS master terminal operator as a warning that an error condition was recovered on the ISC session. The ISC output message in error is either dequeued or retransmitted, and normal input or output operations continue. The message is retransmitted when an FMH7 is received with sense code X'0866' and without EB. The message is dequeued when an FMH7 is received with EB during test, response, or conversational mode output.
- The ERP message (DFS2073) is routed to the IMS LTERM associated with the IMS terminal operator who was the source (using an IMS message switch) of the ISC message switch output in error. The ISC output message in error is dequeued and normal input or output operations continue.
- The ERP message (DFS2073) is routed to the IMS master terminal operator and the ISC session is terminated. The ISC output message in error remains on the IMS message queue.

All ERP messages must carry an EB or CD or the session is terminated. Further, EB received with the ERP message forces end of IMS conversation, response, or test mode. When conversation mode is terminated, the Conversation Abnormal Termination exit routine is also invoked as would occur when an IMS /EXIT command is received. IMS processes FMH7 ERP messages received based on the FMH7 sense code and the DFC protocols associated with the message as in the following sections.

**X'0864xxxx'—Function Abort**

The loop occurs upon retransmission. The sender should not resend the data. Table 29 shows how IMS processes the FMH7 ERP messages with sense code X'0864xxxx'.

Table 29. IMS Processing for Sense Code X'0864xxxx'

	MTO DFS2083	MTO DFS2073	TERMINAL DFS2073
Response mode			
FMH7 W/EB	X		
FMH7 W/CD <sup>1</sup>	X		
Non-last conversation			
FMH7 W/EB <sup>2</sup>	X		
FMH7 W/CD		X	
Last conversation			
FMH7 W/EB <sup>2</sup>	X		
FMH7 W/CD <sup>1, 2</sup>	X		
Message switch			X

## Selective Receiver ERP

Table 29. IMS Processing for Sense Code X'0864xxxx' (continued)

	MTO DFS2083	MTO DFS2073	TERMINAL DFS2073
Test mode			
FMH7 W/EB	X		
FMH7 W/CD		X	
Other Message Types			
FMH7 W/EB	X		
FMH7 W/CD <sup>1</sup>	X		

**Notes:**

1. FMH7 received with CD schedules LUSTATUS-NO-OP (X'0006') with EB to be returned next on the session if the output in error was a response mode reply, the last conversational output, or an ATTACH without SCHEDULER.
2. Conversation Abnormal Termination exit routine is invoked as would occur for /EXIT.

### X'0865xxxx'—Function Abort

The sender is responsible for detecting the loop. Table 30 shows how IMS processes the FMH7 ERP messages with sense code X'0865xxxx'.

Table 30. IMS Processing for Sense Code X'0865xxxx'

	MTO DFS2083	MTO DFS2073	TERMINAL DFS2073
Response mode			
FMH7 W/EB	X		
FMH7 W/CD		X	
Non-last conversation			
FMH7 W/EB <sup>1</sup>	X		
FMH7 W/CD		X	
Last conversation			
FMH7 W/EB <sup>1</sup>	X		
FMH7 W/CD		X	
Message switch			X
Test mode			
FMH7 W/EB	X		
FMH7 W/CD		X	
Other Message Types			
FMH7 W/EB		X	
FMH7 W/CD		X	

**Notes:**

1. Conversation Abnormal Termination exit routine is invoked as would occur for /EXIT.

### X'0866xxxx'—Function Abort

The receiver is responsible for detecting the loop. Table 31 shows how IMS processes the FMH7 ERP messages with sense code X'0866xxxx'.

Table 31. IMS Processing for Sense Code X'0866xxxx'

	MTO DFS2083	MTO DFS2073	TERMINAL DFS2073
Response mode			
FMH7 W/EB	X		
FMH7 W/CD	X		
Non-last conversation			

Table 31. IMS Processing for Sense Code X'0866xxxx' (continued)

	MTO DFS2083	MTO DFS2073	TERMINAL DFS2073
FMH7 W/EB <sup>2</sup>	X		
FMH7 W/CD	X		
Last conversation			
FMH7 W/EB <sup>2</sup>	X		
FMH7 W/CD	X		
Message switch	X		
Test mode			
FMH7 W/EB	X		
FMH7 W/CD		X	
Other Message Types			
FMH7 W/EB	X		
FMH7 W/CD <sup>1</sup>	X		

**Notes:**

1. FMH7 received with CD schedules LUSTATUS-NO-OP (X'0006') with EB to be returned next on the session if the output in error was a response mode reply, the last conversational output, or an ATTACH without SCHEDULER.
2. Conversation Abnormal Termination exit routine is invoked as would occur for /EXIT.

During input, IMS can detect many types of input errors and internal processing conditions, such as undefined transaction codes, incorrect transaction formats, security violations, SNA protocol and data structure errors, and sync-point request errors. These errors can occur on almost any SNA request element of the message and might result in IMS sending an exception response indicating a selective receiver ERP sense code; purging the input message until a synchronizing CANCEL command, RQD1 or RQD2, change-direction indicator, or CHASE command is received; and then sending a selective receiver ERP message.

IMS provides the following sense information in the ERP FMH7 sent with the ERP message:

**X'08260000'** Used for input and internal processing errors, except those below, detected where a normal IMS ERP message is sent.

**X'10030000'** Sent when the ATTACH ATTPRN value is not known to IMS.

**X'08650000'** A function abort sense code for which the data sender is responsible for detecting and preventing loops. Function abort X'08650000' is sent by IMS after receiving an unsolicited CANCEL command (sender-detected error) during an MFS-autopaged input message that does not result in a DFC between-brackets state. Function abort X'08650000' is also sent by IMS for errors detected on input page requests received during an IMS demand-page output message and for errors detected on nonfirst pages of an autopaged input message.

**Related Reading:** For more information on X'08650000', see "Paged Messages ERP" on page 311.

**X'084B0000'** Sent when the attach ATTDPN value is not available for the component entering the data. That is, the input DPN value was not ISCEDT, the optional ISC edit alias, or basic edit; or MFS was not

## Selective Receiver ERP

available for the component. However, if MFS was available but the DPN value was not a valid MID, sense code X'08260000' occurs rather than X'084B0000'.

**X'080F0000'** Sent when the ATTACH data stream profile (DSP) value does not define a valid or defined component number for the half-session name within IMS.

**X'1008xxxx'** The user field (xxxx) is defined by SNA as follows:  
X'6001'—Invalid ATTACH FM header ATTDDBA value  
X'1204'—Invalid version ID on FMH4

Other types of errors detected during input or output operations that cause immediate session termination are:

- Bracket protocol violations
- Send/receive protocol violations
- Unsupported response requests
- Unsupported or invalid FMH types and formats
- Other major VTAM-detected errors or conditions such as LOSTERM, buffer pool or copy RPL space exhausted, or input RU truncation.

### Receiver-Detected Errors during Data Flow Reset State

The following sense codes can be sent and received when the session flow is not in data flow active state, as, for example, during bind, resynchronization (STSN), or start data traffic (SDT):

**X'0845xxxx'** Bind rejected because of invalid user data.

**X'0847xxxx'** Restart mode mismatch detected by secondary half session at receipt of BIND1 STSN or SDT.

**X'08210000'** Bind rejected because of invalid session parameters.

**X'080Dxxxx'** Bind rejected because of bind race.

**Related Reading:** For more information on bind race winner determination, see "Resolving a Bind Race" on page 269.

## Sender ERP

**Related Reading:** For more information on sender error recovery procedure (ERP), see *Systems Network Architecture Format and Protocol Reference: Architectural Logic*.

The SNA CANCEL command can be used during sender error recovery processing as necessary to terminate the chain in error and to provide synchronization between half sessions. CANCEL can be either solicited (by the sender of the exception response) or unsolicited (because of a receiver-detected error, which results in an exception response).

If necessary during chain RU or autopaged output, the sender of the exception response enters into an ERP PURGE mode until both half sessions' send/receive states are again synchronized. The receiver of the exception response must cause the resynchronizing event. This event can be one of the following:

- An RQD1 or RQD2
- Receipt of change-direction

- A CANCEL command (caused by either RQD1 or RQE1/CD), if the exception response occurs during an output chain
- A CHASE command (because of RQD1 on CHASE), if the exception response is received while IMS is between RQE1 output chains (pages) of an autopaged output message

### **Sender ERP Sense Codes**

The only sender ERP sense codes sent between two IMS subsystems are X'08130000' and X'0846xxxx'.

The valid contention sense codes received by IMS and additional operations performed are:

- X'08130000'—Bracket reject; no RTR sent  
No ready-to-receive (RTR) condition is set. When the pseudo-wait is satisfied, the message is retransmitted from the beginning. This sense code can also be used to reject the SNA BIS command sent by IMS.
- X'08140000'—Bracket reject; RTR sent  
The message is returned to the queue, and an RTR pending state is entered. No output is sent while IMS is between-brackets until an RTR is received. However, messages flow, subject to the bracket and send/receive protocol defined for the message type or component, when IMS is again left in an in-brackets/SEND state following receipt of change-direction.

After either of these sense codes, IMS attempts to send the output message.

IMS recognizes receipt of contention system sense code X'08130000' when assuming a primary half-session role, returns the output message to the message queue, and enters a pseudo-receive state that can be satisfied by:

- Receiving input from the secondary half session
- Being posted for output as a result of an IMS master terminal operator command, a message switch from another logical unit, or additional messages inserted by an IMS application program

After either of these actions, IMS attempts to send the output message.

IMS only indicates contention when assuming a secondary half-session role and receiving either input data or normal flow commands (BID and BIS) after having already initiated a bracket to the primary half session. IMS sends the X'08130000' sense code to indicate contention in these cases. The X'08140000' contention sense code is not sent by IMS.

Additional exception response sender ERP sense codes sent and received are:

- X'08190000'—No output available  
This code is sent by IMS when assuming the primary half-session role and no output is immediately available following receipt of a ready-to-receive (RTR) indicator. IMS does not receive this sense code because IMS does not send RTR.
- X'0846xxxx'—Selective Receiver ERP  
IMS sends and receives the SNA selective receiver ERP sense code. IMS sends this sense code when a response mode, conversational transaction, or application abnormally terminates. xxxx is a user sense field that is ignored by IMS when it is received. However, IMS does include this sense code in the message that is sent to the master terminal or to the terminal operator that was the source of the message. During output, it is normally set to the binary value of

## Sender ERP

the IMS error message number that is sent as the ERP message when sending the response. However, the user sense code can be set to X'0000' as, for example, when the error, such as an application abend, occurs for a response mode or conversational transaction.

### **Related Reading:**

- For information on how IMS sends and receives the SNA selective receiver ERP sense code, see “Selective Receiver ERP” on page 312.
- For more information on this sense code, see *IMS/ESA Messages and Codes*.

- X'0864xxxx'—Function abort. Loop occurs upon retransmission. Sender should not resend data.

IMS receives, but does not send, this function abort sense code. When receiving this code, IMS dequeues the associated output message (if still active) and then continues with normal input or output operations. xxxx is a user sense field ignored by IMS. An LUSTATUS - abort is sent by IMS if input is received prior to the end of an IMS MFS-paged output message.

The function abort sense code X'0864xxxx' cannot be sent to IMS during nonlast IMS conversational output; otherwise, the output message is returned to the queue. The master terminal operator is notified and the session is terminated. When the function abort sense code X'0864xxxx' is sent following the last conversational output message, the message is dequeued and the Conversational Abnormal Termination exit routine is invoked just as occurs for /EXIT.

- X'0865xxxx'—Function abort. Sender responsible to detect loop.

IMS receives, but does not send, the sender ERP abort sense code. If received, this code causes IMS to return a message to the queue and close the session. IMS retransmits the message from the beginning at the next opportunity after session restart. xxxx is a user sense field ignored by IMS.

- X'0866xxxx'—Function abort. Receiver responsible to detect loop.

This sense code causes IMS to retransmit the message from the beginning at the next opportunity. The session is not terminated. xxxx is a user sense field ignored by IMS.

- X'08670000'—Multichain ERP purge.

IMS sends the multichain purge sense code at the end of ERP purging, as defined under “ERP Purging” on page 305. IMS receives the multichain ERP purge sense code only after receiving an exception response to a nonfirst or nonlast page of an MFS output autopaged message. Non-MFS autopaged output and the first and last page of an MFS output autopaged message are sent requesting definite response. Exception to these SNA chains can result only in single chain purge.

Other sender ERP sense codes cause IMS to notify the master terminal operator and terminate the session.

### **Sender-Detected Errors on Nonpaged Messages**

When an error is detected after initiating transmission of a single SNA nonpaged, multi-RU message, the sender can send the SNA unsolicited CANCEL command. IMS sends unsolicited CANCEL only if a current output message is terminated by an operator /DEQUEUE command entered prior to the last segment of the message (single SNA chain) being transmitted.

If IMS receives an unsolicited CANCEL while data is being received by IMS, the message is discarded or backed out.

## Sense Codes that IMS Receives

**Definition:** A *sense code* is a 2-byte field containing the category and modifier defined for the particular exception condition that has occurred. The 2 bytes following the sense code can contain optional user data, and are not supported for all sense codes. Table 32 shows the input sense codes that IMS receives.

Table 32. Sense Codes that IMS Receives

Input Sense Codes:		Sender ERP	FMH7	LUSTATUS
Commit/NO-OP	- X'00060000'			X
Queue empty	- X'00070000'			X
Bind race	- X'080D0000'	X		
Contention	- X'08130000'	X		
Contention	- X'08140000'	X		
Invalid parms	- X'08210000'	X		
Sel Rcvr ERP	- X'08460000'	X		
Reject restart	- X'08470000'	X		
Abort	- X'08640000'	X	X	X
Abort	- X'08650000'	X	X	X
Abort	- X'08660000'	X	X	X
ERP purge	- X'0867'	X		
Other		1	1	1

**Note:** <sup>1</sup> Other sense codes cause IMS to notify the master terminal operator and terminate the session.

**Related Reading:** For more information on sense codes, see *Systems Network Architecture Format and Protocol Reference: Architectural Logic*.

## Sense Codes that IMS Sends

Table 33 shows the output sense codes that IMS sends.

Table 33. Sense Codes that IMS Sends

Output Sense Codes:		Sender ERP	FMH7	LUSTATUS
NO-OP	- X'00060000'			X
Queue empty	- X'00070000'			X
Bind race	- X'080D0000'	X		
ATTDSP	- X'080F0000'		X	
Contention	- X'08130000'	X		
No output	- X'08190000'	X		
Invalid parms	- X'08210000'	X		
Not supported	- X'08260000'		X	
Sel Rcvr ERP	- X'08460000'	X		
Reject restart	- X'08470000'	X		
ATTPDN	- X'084B0000'		X	
Abort	- X'08650000'	X		X

## Sense Codes that IMS Sends

Table 33. Sense Codes that IMS Sends (continued)

Output Sense Codes:		Sender ERP	FMH7	LUSTATUS
ERP purge	- X'0867'	X		
ATTPRN	- X'10030000'		X	
FMH4 Version ID	- X'10081204'		X	
ATTDBA	- X'10086001'		X	

## SIGNAL Protocol

IMS sends the SIGNAL command to request change-direction (SIGNAL RCD - X'00010000') at the end of an input message chain that does not indicate change-direction and that produces output that must be sent prior to subsequent input. For example, IMS replies to input commands, response mode transactions, conversational transactions, or input while in test mode. SIGNAL RCD is sent before IMS sends any required DR1 or DR2 response for the input message.

Based on the IMS response protocol and the point at which IMS sends SIGNAL RCD, IMS requires that the next input following SIGNAL RCD be either LUSTATUS or CHASE indicating CD or EB. If neither of these is received by IMS, the session is terminated. Change-direction allows IMS to send the pending output message; end-bracket causes the message to be dequeued and the conversation mode, response mode, or test mode to be terminated. No other input can be processed until the pending output message has been successfully transmitted or dequeued.

After receiving SIGNAL RCD while in bracket state, IMS sends change-direction either at the end of the current output message or immediately on receipt of an LUSTATUS. A SIGNAL RCD received while between brackets prevents IMS from sending further output messages until normal flow input (data or SNA command) occurs.

**Related Reading:** For an example of using SIGNAL RCD, see “Signal Protocol Example” on page 546.

## Symmetrical Session Shutdown for LU 6.1 (SBI and BIS)

Two data flow control commands — stop bracket initiation (SBI) and bracket initiation stopped (BIS)—allow a symmetrical and orderly termination for peer level LU 6.1 half sessions. SBI is a VTAM-expedited flow command, and BIS is a normal flow command.

These commands control only the normal flow requests that initiate new brackets. They do not preclude replies that can flow within existing brackets. Either half session can send SBI at any time to request the receiving half session to suppress future initiation of a bracket. After the receiver of the SBI reaches a point between brackets that is acceptable for shutdown, the receiver returns the BIS command to the SBI initiator. The BIS sender then enters a “no begin-bracket” or “NOBB” state, during which that half session can receive bracket initiation requests and may send requests or replies in the normal flow when in-brackets, but cannot initiate a bracket on its own. The primary half session is responsible for detecting and resolving SBI race conditions by maintaining indications of which half session is in NOBB state and terminating the session if both half sessions reach this state.



## Symmetrical Session Shutdown for LU 6-1 (SBI and BIS)

A successful completion of a shutdown sequence refers to successful entry into the NOBB state by both half sessions— that is, normal session termination.

IMS sends the SBI command either when the IMS /QUIESCE command is entered or when the quiesce option is specified on the /CHECKPOINT command. When BIS is returned from the other subsystem, IMS continues with the QUIESCE or CHECKPOINT command process. If the /QUIESCE command was entered and a CHECKPOINT QUIESCE is not in progress, IMS sends BIS at the next transition to a between-brackets state. If a CHECKPOINT QUIESCE is in progress when IMS receives BIS, IMS continues to send output according to other /CHECKPOINT parameters (FREEZE, PURGE, and DUMPQ) and then sends BIS when the shutdown process is complete.

After receiving an SBI or BIS, IMS sends BIS at the next between-brackets state when a CHECKPOINT QUIESCE is not in progress, or at the end of the shutdown process when a CHECKPOINT QUIESCE is in progress.

When IMS is the primary half session, IMS checks after either sending or receiving BIS to ensure that both half sessions are in NOBB state. If both are, IMS resets all message counts and terminates the session. Otherwise, IMS continues with normal processing. When IMS is the secondary half session, IMS waits for the primary half session to terminate the session. If an LTERM subpool is allocated to the session, it is deallocated (and associated message counts are reset) prior to the session termination when both half sessions are in NOBB state. This deallocation/termination process allows a subsequent cold start of the session and frees any allocated subpools for use by other sessions. This process also ensures that no resynchronization or recovery is required when the session is restarted.

Because initiating a bracket is not allowed after the NOBB state is entered, a problem occurs when a single-bracket-chain (begin- and end-bracket) input message requires an immediate reply (such as a system or error message) that initiates a bracket. Rather than preclude end-bracket on these input messages while in NOBB state, IMS accepts and continues to process all input. When a reply occurs that would require IMS to initiate a bracket while in NOBB state, the session is terminated.

While IMS is in response mode, in conversational mode, or waiting for a Fast Path response, subpool deallocation cannot occur nor can BIS be sent, because no EB has flowed either on output or input (that is, IMS is not between-brackets).

**Related Reading:** For examples of SBI/BIS protocol, see “SBI/BIS Examples” on page 545.

---

## Function Management Headers

In SNA, function management (FM) headers are an optional part of the request unit sent over a link. This section describes the FM headers supported by IMS on ISC sessions. The headers addressed in this section include:

- The ATTACH function management header. This type 5 header is used to attach a process so it can receive session input. It carries the name of the process to be attached synchronously, as well as other parameters to be used by the attached process. All messages have an explicit or implicit ATTACH FM header. The header is implicit when not actually sent with the message. An implicit ATTACH FM header assumes reset parameter values or values of a previous ATTACH as defined later in this section.

## Function Management Headers

- The error recovery procedure (ERP) function management header. This type 7 header is sent after an error condition requiring an error message to be sent is detected.
- The reset attached process (RAP) function management header. This type 5 header is input to IMS to cause the attached process and the associated data flow control states to be reset.

The following header types are subordinate to the ATTACH FM header and cannot be used unless the ATTACH FM header has previously been sent within that bracket to attach the associated process:

- The SCHEDULER function management header. This type 6 header is sent after the SCHEDULER process has been attached. It carries the name of the process to be asynchronously scheduled within IMS, as well as other parameters required by the SCHEDULER process.
- The SYSMMSG function management headers. The type 6 system message headers precede system messages sent by or received by IMS.
- The data descriptor function management header. This type 4 header is used in conjunction with MFS to send a data structure name, version ID, or a current output field tab separator.
- The QMODEL function management headers. These type 6 headers are used for demand-paged messages sent by IMS Message Format Service.

All function management headers must be on a first-in-chain or only-in-chain.

Table 34 summarizes the header types that IMS sends and receives. These header types are described in the sections that follow.

*Table 34. Function Management Header Types*

<b>Function Management Header Type:</b> <b>Y = Supported</b>	<b>Sent by IMS</b>	<b>Received by IMS</b>
FMH4 - Data Descriptor	Y	Y
FMH5		
ATTACH	Y	Y
Reset Attach Process (RAP)		Y
FMH6 - QMODEL		
QGET		Y
QGETN		Y
QPURGE		Y
QSTAT	Y	
QXFR	Y	
FMH6 - SCHEDULER	Y	Y
FMH6 - SYSMMSG		
SYSSTAT (default)		Y
SYSSTAT	Y	Y
SYSERROR	Y	Y
FMH7 - ERP	Y	Y

When receiving a header, IMS ignores any coded parameters beyond the last parameter to which it is sensitive (that is, that IMS supports). The other LU 6.1 subsystem should do the same when receiving headers that IMS sends.

## Using FM Headers to Invoke ISC Edit

In the following situations, IMS can use ISC edit (ISCEDT) to edit transactions, commands, and message switches between LTERMs for an IMS-ISC session:

- The attach manager is in the reset state (between-brackets or following a RAP), and either no ATTACH FM header or an ATTACH FM header that does not specify a destination process name (ATTPDN) is received.
- The destination process name (ATTPDN/SCDDPN) within the ATTACH or SCHEDULER FM header equals ISCEDT or indicates the user-defined alias for ISC edit as defined during system definition on the COMM macro statement.
- The attach manager is not in the reset state, the active process is ISC edit, and either no ATTACH FM header or a header that does not specify an ATTPDN is received.

Input that does not specify the ATTPRN/SCDPRN parameter is edited only for the transaction code and password at the beginning of the message. The optional IMS password and leading characters less than X'41' are deleted during editing. The format and requirements for the IMS transaction code and password are the same as for basic edit. No editing of any kind occurs for the remainder of the message.

Input parameters passed from the attach manager or SCHEDULER (if supplied) are return destination process name (ATTRDPN/SCDRDPN), return primary resource name (ATTRPRN/SCDRPRN), and primary resource name (ATTPRN/SCDPRN).

The input ATTPRN/SCDPRN parameter is passed from the attach manager or SCHEDULER as an IMS transaction code or as the LTERM name on a message switch, as defined under "Initiating a Process: ATTACH FM Header". When the ATTPRN/SCDPRN parameter is supplied, no editing of any kind occurs for the input. ATTPRN/SCDPRN defines an IMS destination, but does not provide an input password to IMS transaction security (if this security is defined for the ISC node).

### **Related Reading:**

- For examples of using the ATTACH parameter for ISC edit, see "Appendix F. Examples Using ISC Edit ATTACH Parameters" on page 489.
- For information on the format and requirements for the IMS transaction code and password for basic edit, see *IMS/ESA Administration Guide: System*.

## Initiating a Process: ATTACH FM Header

The ATTACH FM header is used to attach a process to receive session input. It carries the name of the process to be attached synchronously, as well as other parameters used by the attached process. Because ATTACH is defined for synchronous execution, you must understand the relationships between the process to be attached, the IMS message types and execution modes, and the SCHEDULER process.

**Related Reading:** For more information on ISC and IMS execution modes, see "Relationship of ISC and IMS Execution Modes" on page 238.

IMS sends the ATTACH FM header without a SCHEDULER FM header:

- On the conversational, response mode, or IMS command output message replies resulting from an input ATTACH.
- On asynchronous replies that result from a message received on the same session with ATTACH EB.

## Initiating a Process: ATTACH FM Header

- When the other half session is bound without SCHEDULER model support.
- On system messages sent by IMS via SYSMSG.

An IMS message can consist of either single or multiple SNA chains as indicated within the ATTACH ATTIU parameter. The ATTACH FM header can be present only once for each input or output IMS message. For output MFS demand-paged messages, the ATTACH FM header is present only on the first output SNA chain. The first input paging request to the demand-paged output must contain an ATTACH (for DPN=QMODEL); subsequent input page requests for the same demand-paged output message can optionally contain an ATTACH (which must also indicate DPN=QMODEL).

**Related Reading:** For information on the format of the ATTACH FM header, see “ATTACH FM Header Format” on page 339.

## Error Recovery Procedure FM Header

IMS sends the FMH7 ERP header and its associated message following a selective receiver ERP exception response (X'0846'). An ERP exception response need not be received by IMS prior to receiving a FMH7 ERP header and associated message. FMH7 messages sent by IMS indicate either RQE1/CD or RQD1/EB, as appropriate. ERP messages sent or received by IMS are limited to a single SNA chain of SCS characters and must not contain VLVB format records. FMH7 messages received by IMS must indicate either CD or EB. RQE1 or RQD1 is allowed on either.

**Related Reading:** For more information on the error recovery procedure (ERP) and its associated ERP message header, see “Selective Receiver ERP” on page 312.

Single-segment error messages created within IMS are 79 characters or less in length. Multisegment messages created within IMS can be greater than 79 characters in length, but each segment has a 79-character maximum length.

Error messages created by input to the ISC error message process for output to the master terminal (or source operator terminal for a message switch) create single- and multisegment output messages (DFS2073 or DFS2083).

IMS inserts error messages into IMS message queues using a default MFS MOD name. These messages are formatted as defined by the MOD during output to a component defined with MFS DPM. These MODs and associated DOFs are contained in the MFS format library.

Each selective receiver ERP message is preceded by an ERP message header that includes the specific system and user error sense codes.

**Related Reading:** For information on the ERP FM header format, see “Error Recovery Procedure FM Header” on page 349.

## Resetting the Active Process: RAP FM Header

The reset attached process (RAP) FM header is used to reset the receiving half session's active process and all session states, except bracket and send/receive states, to the equivalent of a between-brackets state.

## Resetting the Active Process: RAP FM Header

IMS receives the RAP FM header, but does not send it. After receiving the RAP request, IMS responds with an ATTACH SCHEDULER, or with LUSTATUS - queue empty if no output is available to be sent. If the RAP request is received during an IMS demand-paged output message, the message is dequeued (committed) before a check for more output is made. The IMS operation performed for the RAP request is the same as for the NEXTMSG operator control request provided with MFS input or the PA2 request from a 3270 device.

The RAP request must be sent without data and must indicate change-direction, RQD2, or RQE2.

**Example:** The following example illustrates the use of the RAP FM header:

```

      ←----- RAP CD RQE2 -----→
> dequeue any demand-page output in progress,
  then, if more output is available..
      ----- ATTACH DPN= ... -----→
      .
      .
> or, if no output exists..
      ----- LUSTATUS - queue empty CD/EB -----→
```

**Related Reading:** For more information on the RAP FM header, see “Reset Attached Process (RAP) FM Header Format” on page 355.

## Requesting Asynchronous Transaction Processing: SCHEDULER FM Header

ATTACH is defined for synchronous scheduling and execution; SCHEDULER is defined for asynchronous scheduling and execution.

The SCHEDULER process can be attached to send messages to be processed asynchronously. That is, IMS perceives the messages as being processed without regard to scheduling or execution timing, whether output will result at all, and ignoring any synchronous relationship between the input and any resulting reply. Each half session between the logical units might indicate, by using the session bind parameters, whether the SCHEDULER process is supported and available to receive asynchronous messages. The IMS half session always indicates the SCHEDULER process is available. The other half session might indicate that the SCHEDULER process is or is not available to receive asynchronous input.

When the bind indicates that the other half session does not support the SCHEDULER process, IMS defaults to sending each output with the ATTACH FM header.

The SCHEDULER process is attached by indicating the SNA name (X'02') as the ATTACH DPN parameter and by concatenating the SCHEDULER FM header to the ATTACH FM header. During the time that the SCHEDULER is attached, each subsequent message to be scheduled asynchronously must be sent with at least the SCHEDULER FM header, but does not require another ATTACH FM header. Variable-length ATTACH parameters passed to the SCHEDULER are ignored on input and are not sent with output by IMS. The SCHEDULER FM header carries the name of the process to be scheduled (SCDDPN), as well as information necessary for scheduling and return-reply routing. SCDDPN is a required parameter for the

## Requesting Asynchronous Transaction Processing: SCHEDULER FM Header

SCHEDULER and must be supplied on IMS SCHEDULER input. SCDDPN on messages sent by IMS are set from a previous input SCDRDPN or optionally by MFS using the output message format descriptor. When no other value is available, SCDDPN is set to "ISCEDT" as a default.

The SCHEDULER process is always available for IMS input. IMS sends an ATTACH with the ATTDPN parameter set to X'02' (SCHEDULER) concatenated with a SCHEDULER FM header under each of the following conditions:

- When the other half session supports the SCHEDULER model (as defined in the bind parameters).
- When sending an asynchronous reply resulting from input entered previously on the session and scheduled using the SCHEDULER process.
- When sending unsolicited asynchronous messages. Examples of such messages are an IMS message switch or an output reply resulting from an application not scheduled directly as a result of input on the session.

Within IMS, the SCHEDULER supports both single- and multiple-chain input and output messages using the ATTACH ATTIIU parameter. IMS does not permit multiple input or output messages to be scheduled using the ATTACH multiple-chain indicator.

The SCHEDULER FM header can be present only once for each input or output IMS message. For output MFS demand-paged messages, the SCHEDULER FM header is present only on the first output SNA chain (first output MFS page). IMS does not send or receive the SCHEDULER SCHEDSTAT, PURGE, or PURGSTAT FM headers.

### Related Reading:

- For information on the input and output requirements for bracket, send/receive, and response protocols for the SCHEDULER, see the data flow control (DFC) protocols defined in "Data Flow Control Protocol Reference" on page 295.
- For more information on the SCHEDULER FM header, see "SCHEDULER FM Header Format" on page 355.

## System Message Process (SYSMSG) and Related FM Headers

The system message process is indicated by a process name in the ATTACH FM header.

Input system messages are routed to the master terminal operator when no ATTPRN is supplied in the ATTACH or SYSERROR FM header for a SYSMSG. This routing is accomplished by converting the incoming SYSMSG into IMS system message DFS2072. If an ATTPRN value is supplied, it becomes the IMS input message destination transaction code or LTERM name. The system message process cannot be used to access the IMS command processor.

A reply can result from an input SYSMSG if the ATTPRN value supplied is for an IMS transaction. This reply is returned through ISC edit or MFS, depending upon the output component definition.

IMS inserts system messages into IMS message queues using a default MFS MOD name. These messages are formatted as defined by the MOD during output to a component defined with MFS DPM. These MODs and associated DOFs are contained in the MFS format library.

## System Message Process (SYSMSG) and Related FM Headers

IMS sends SYSMSG only for:

- IMS broadcast messages
- System messages that IMS sends after sending a response to input. These system messages can occur instead of the application sending a reply (for example, when an abnormal termination occurs). IMS system messages directly solicited by an IMS command are sent as normal replies. Other system-generated messages can result from error conditions. These messages are converted to exception responses to the input transaction and are sent as error messages using the error recovery process (ERP).

MFS might edit an output SYSMSG, depending on the output component definition and whether the reply was inserted with an MFS MOD name.

Messages sent to or received by SYSMSG can include either the SYSSTAT or SYSERROR FM header. The SYSSTAT FM header is assumed by the receiver if no header is supplied by the sender. IMS can receive either header, and attaches a prefix to each header with IMS message number DFS2072 (if the message is to be sent to a master terminal operator). IMS sends broadcast output using the SYSSTAT header. All other IMS system messages sent as SYSMSG use the SYSERROR header. If the ATTACH ATTRDPN or ATTRPRN parameters are supplied on an input message that results in SYSMSG output, these parameters are included in the output SYSERROR FM header to be used by the receiver's SYSMSG process. The ATTACH ATTRDPN and ATTRPRN parameters are not sent for an IMS output SYSMSG.

Single-segment system messages created within IMS are 79 characters or less in length. Multisegment messages created within IMS can be greater than 79 characters in length, but each segment has a 79-character maximum length.

System messages created by input to the ISC SYSMSG process for output to the master terminal create single- and multisegment output messages. System messages created by input to the ISC SYSMSG process in which ATTRPRN and ATTRDPN are specified must have a MID capable of handling multisegment messages with a minimum of five segments.

**Related Reading:** For more information on the system message process, see “SYSMSG Process Headers” on page 356.

## System Message Process (SYSMSG) and Related FM Headers



---

## Chapter 15. Using MFS with ISC

Message formatting for MFS DPM is specified on the IMS system definition TERMINAL macro statement via the parameter DPM-Xn, where X might be A or B. For ISC nodes, the form DPM-Bn is always used. This section addresses the following:

- How MFS DPM-Bn can be invoked to format IMS input and output
- The way in which the ATTACH, SCHEDULER, and reset attached process (RAP) function management headers are used by MFS
- The MFS support for the SNA-defined QMODEL and data descriptor function management headers (which control MFS processing)

**Related Reading:** For information on the MFS facilities available to the ISC session using the ATTACH, SCHEDULER, RAP, QMODEL, and data descriptor protocols, see *IMS/ESA Application Programming: Transaction Manager*.

### **In this Chapter:**

- “Activating MFS Input Formatting”
- “Activating MFS Output Formatting” on page 330
- “MFS Distributed Presentation Management (DPM) Messages” on page 330
- “MFS Page Delete Function” on page 331
- “MFS Online Error Detection” on page 331
- “The ATTACH and SCHEDULER FM Headers under MFS” on page 333
- “Data Descriptor FM Headers” on page 334
- “Controlling Demand-Paged Messages: QMODEL FM Headers” on page 335
- “Request (Input) QMODEL FM Headers” on page 336
- “Reply (Output) QMODEL FM Headers” on page 337
- “The RAP FM Header under MFS” on page 338

Some input MFS format errors (such as invalid cursor, incorrect output formats, or no output formats) are not detectable within IMS before IMS sends the requested response to the input message. In these cases, IMS sends the error message by using ATTACH ATTDPN=SYSMSG. This ATTACH results even if the input message indicated change-direction and exception response and is an implicit acknowledgment to the message.

An ATTACH SYSMSG received by IMS during demand-paged output is treated as an invalid MFS paging request and causes session termination. The received SYSMSG is discarded.

---

## Activating MFS Input Formatting

When MFS is used, input messages can be processed by the message and format descriptors. When IMS receives an input message from an ISC logical unit, basic edit or ISC edit is performed unless MFS is defined for the input component (ATTDSP) and a MID name accompanies the message. The MID name can be supplied by including it either as the ATTDPN parameter on the ATTACH function management header or as the SCDDPN parameter on the SCHEDULER FM header.

## Activating MFS Input Formatting

The MFS escape characters (//) are not supported for ISC. When the MID name is present, MFS edits the message using the specified MID and its associated device input format (DIF).

---

## Activating MFS Output Formatting

MFS output formatting occurs when an output message has an associated message output descriptor (MOD) supplied in one of the following ways:

- The application program supplies a MOD name with the output message.
- The input message is processed by a message input descriptor (MID) whose definition specifies a MOD name for output formatting.
- The output message is a message switch that is created by a MID whose definition specifies a MOD name for output editing.

If no MOD is associated with the output message, no MFS editing occurs. Values for the ATTACH and SCHEDULER parameters are the default values that are defined earlier under these respective header types.

You can use the MOD to specify the next MID name to be used to format input as the result of output. In an ISC session, IMS cannot control the relationship between formatted IMS output messages and subsequent input messages. Therefore, involvement is required by the remote subsystem or user application program to ensure that the proper MID name is used for input MFS formatting. The ATTACH and SCHEDULER RDPN/DPN parameters perform the ISC function of externalizing the next MID on output and invoking MFS formatting on subsequent input.

### Related Reading:

- For information on MFS MID/MOD chaining, see “MFS Application Design” in *IMS/ESA Application Programming: Transaction Manager*.
- For information on the ATTACH and SCHEDULER RDPN/DPN parameters, see “ATTPDN” on page 343.

---

## MFS Distributed Presentation Management (DPM) Messages

An IMS MFS DPM message can consist of either single or multiple SNA chains as indicated within the ATTACH parameters. For MFS DPM input or output using a device format defined with the paging OPTIONS=MSG, IMS sends or receives the entire message as a single chain of one or more related transmissions. For MFS DPM output using a device format defined with paging OPTIONS=DPAGE or PPAGE, IMS sends each logical or presentation page as a single chain of one or more related transmissions.

MFS output DPM format definitions also allow options for paged output (OPTIONS=PPAGE or DPAGE) to be sent as demand-paged or autopaged. Demand-paged output requires paging requests to the other half session for each output page. Autopaged output is sent as a series of consecutive pages (chains) with no paging requests to the other half session.

MFS DPM input using a device format defined with the paging OPTIONS=DPAGE allows IMS to optionally receive each logical page as a single chain of one or more related transmissions. All chains of autopaged input (multiple logical pages as multiple related input chains) are received consecutively without paging requests by IMS.

## MFS Distributed Presentation Management (DPM) Messages

FM headers can occur as part of the first or only transmission of each input or output chain of the message.

---

### MFS Page Delete Function

The MFS page delete function is not available for use by an ISC session. `OPTIONS=NPGDEL` is forced on the IMS system definition `TERMINAL` macro statement or an ETO user descriptor, because paging requests other than QMODEL-architected paging requests are not supported. This includes MFS operator control table requests and other forms of input data that might occur during output demand-paging. An error (exception response and appropriate ERP message) occurs if data other than a valid QMODEL paging request is received during demand-paged output. The output message is returned to the queue as a result of the error condition and retransmitted at the next opportunity. If the output demand-paged message is conversational or response mode output, it is immediately resent and input is not allowed until the output message is successfully transmitted or dequeued by a valid QMODEL paging request or appropriate ERP action.

---

### MFS Online Error Detection

This section describes how IMS detects input errors, output errors, and paging errors for MFS.

#### Output Errors

MFS output errors are detected after IMS has already sent the response to any preceding input. If an error is detected during MFS MOD or DOF block selection, an error message is sent as `ATTACH SYSMSG`, and the IMS message is returned to the message queue for retransmission. If the MFS test mode is in effect, it is reset.

Invalid page requests cause error messages to be sent.

#### Input Errors

An error message is sent and the input message is rejected if one of the following occurs:

- An error is detected during MFS MID or DIF block selection.
- A nonzero version ID is received in the data descriptor FM header and it does not match the version ID in the MFS descriptor.
- An error is detected during DPAGE selection (that is, no condition is satisfied on matching the DPAGE label with the DSN in the DD FM header or the `COND=` with the data).

If one of the following errors is detected during multiple DPAGE input, an error message is sent to the other subsystem, and the input message is rejected:

- Multiple transmission chains. More data is present in the chain than defined for the DPAGE selected.
- Any mapped-input LPAGE contains no data segments (as a result of segment routines cancelling all segments, for example).

If one of the following errors is detected during a single DPAGE input (that is, multiple DPAGE input is not requested in MFS definitions), an error message is sent and the input message is rejected:

## MFS Online Error Detection

- A single transmission chain is received and contains more data than defined for the DPAGE selected.
- Multiple transmission chains are transmitted.
- The mapped-input message contains no data segments (as a result of segment routines cancelling all segments, for example).

If the input message is cancelled by the User Segment Edit exit routine, or if a User Segment Edit exit routine failure is detected, an error message is sent to the other subsystem. In the latter case, the input message is rejected.

## Paging Errors

An error message is sent to the other subsystem when an invalid paging request (QMODEL FM header) is detected:

- The QNAME parameter on the QGETN or QGET function management header does not match the ATTDQN/SCDDQN parameter on the ATTACH or SCHEDULER FM header sent by IMS. An attached demand-paged output message is returned to the message queue for retransmission. A scheduled demand-paged output message continues to wait for a QGETN with the proper QNAME specified.
- The QORG parameter on the QGETN or QGET FM header is invalid. The output message is returned to the message queue for retransmission.
- The QGETN FM header is received and no output message is in progress.
- A paging request other than QGETN is received as the first input following scheduled demand-paged output (ATTACH SCHEDULER), or a QSTATUS FM header is sent by IMS because of an invalid cursor on a paging request.
- The QGETN FM header is received for an OLP demand-paged output and current cursor position is at the last page.

**Related Reading:** For more information on invalid cursor conditions, see “QSTATUS FM Header” on page 337.

The QGET FM header causes a QSTATUS or an error message to be sent in the following cases:

- Between messages.
- Non-OLP demand-paged output message is in progress. An error message is sent and the output message is placed on the queue for retransmission.
- Cursor value does not contain a valid 2-byte binary number. An error message is sent, and the output message is placed on the message queue for retransmission.
- Cursor value is outside the range for the output message.

**Related Reading:**

- For more information on invalid cursor conditions, see “QSTATUS FM Header” on page 337.
- For more information on paging errors, see “Error Recovery Procedure FM Header” on page 324.

---

## The ATTACH and SCHEDULER FM Headers under MFS

All input and output messages include an implicit or explicit ATTACH FM header and, optionally, a SCHEDULER FM header. MFS DPM provides format definition options to permit the insertion of input routing parameters (ATTRDPN/SCDRDPN, ATTRPRN/SCDRPRN, and ATTPRN/SCDPRN) into the input data stream. The input ATTDPN/SCDDPN is the MFS MID used to format the input message. MFS DPM provides output format definition options to permit the insertion of user-defined information into these headers.

The ATTACH FM header and SCHEDULER FM header, if required, are sent without data as the only element of the first transmission chain of demand-paged output. The first input paging request to the demand-paged output must contain an ATTACH (for DPN=QMODEL); subsequent input page requests for the same demand-paged output message can optionally contain an ATTACH (and must also indicate DPN=QMODEL). For all other MFS input and output messages the ATTACH (and SCHEDULER, if required) FM headers are sent with data as the first or only element in the first or only transmission chain of the message.

MFS uses the following ATTACH and SCHEDULER FM header parameters:

- ATTDP and ATTIU. These ATTACH parameters indicate whether the message is demand-paged or whether it is single or multiple chain.
- ATTDDBA. This ATTACH parameter indicates the output data blocking algorithm or input data deblocking algorithm to be used by IMS. The data entity deblocked on input becomes the input data record to MFS. MFS then produces a standard IMS message consisting of pages and segments. On output, the MFS record becomes the data entity for output VLVB blocking. An exception is MFS DPM stream mode, which is sent as chain (ATTDDBA) output.
- ATTDPN/SCDDPN and ATTPRN/SCDPRN. The ATTDPN/SCDDPN parameter is used on input to activate MFS input formatting. The ATTPRN/SCDPRN can be inserted into the input data stream and, therefore, made available to the application. On output message replies from IMS, the ATTDPN/SCDDPN or ATTPRN/SCDPRN can be inserted into the ATTACH or SCHEDULER FM headers of the output message processed by MFS as specified by MFS format descriptions. These override the ATTRDPN/SCDRDPN and ATTRPRN/SCDRPRN that might have been wrapped from the source input message ATTACH or SCHEDULER FM headers.
- ATTRDPN/SCDRDPN. The ATTRDPN/SCDRDPN can optionally be inserted into the input data stream and, therefore, made available to the application for input messages processed by MFS DPM. The return destination process name can be inserted into the output ATTACH or SCHEDULER FM header of the output message processed by MFS. The MFS-suggested return destination process to be associated with a reply to the output message is the MID name specified in the NXT= operand of the MOD.
- ATTRPRN/SCDRPRN. The return primary resource name (RPRN) can be inserted into the ATTACH or SCHEDULER FM header of the output message processed by MFS. It can be specified using MFS definition. If returned to IMS with a reply to the message on which it was sent, it determines the destination — transaction or LTERM — that is to receive the input reply.
- ATTDQN/SCDDQN. The destination queue name is sent by IMS on demand-paged output as a message identifier and should be returned on all requests associated with demand-paged messages.

## The ATTACH and SCHEDULER FM Headers under MFS

When the first chain (ATTACH SCHEDULER) of an asynchronous demand-paged output message is successfully sent, the output queue (IMS LTERM) is automatically locked against input or output. This function is also automatic following a QSTATUS FM header sent by IMS because of a cursor error on a demand-paged request. The queue lock state is reset when the page request (ATTACH QMODEL, QGETN with QNAME) is received and the first page of data is sent without resending the ATTACH or SCHEDULER FM header. The queue lock condition is also reset across session failure and subsequent restart. During the time that IMS waits for this page request (QGETN), provided conversation or response mode is not used, IMS can send or receive messages for other LTERMs. During the time that the output LTERM is locked, no input is accepted from the ISC session that produces output to the locked queue because of the input/output component relationships defined for the locked queue using the IMS system definition NAME macro statement or an ETO logon descriptor. Input is accepted that produces output for other LTERMs.

---

## Data Descriptor FM Headers

The input and output data descriptor FM headers are used to receive and send a data structure name and version ID or a field tab separator character.

### Input Data Descriptor FM Header

On input to IMS, this header can follow the ATTACH or SCHEDULER FM header and is used to receive a data structure name (if DPAGE selection is by DPAGE name) and version ID or the input field tab separator character. The input data descriptor FM header should be sent to IMS if input DPAGE selection is to be performed on the DPAGE name; that is, if you specify OPTIONS=DNM on the DIV statement (TYPE=INPUT).

If you set OPTIONS=DNM, and specify no DPAGE name or the wrong DPAGE name, an error occurs.

The version ID should be sent to IMS in the first or only FM header of the input message if MFS is to verify that the correct definition is used to map the data. If the version ID is sent and is X'0000', no verification occurs. The field tab separator is not required on input, because MFS provides for up to eight separators with the FTAB function. If received on input, the field tab separator is used instead of the MFS FTAB specification (if any) for the current transmission.

**Related Reading:** For the format of the input data descriptor FM header, see “Input Data Descriptor FM Header Format” on page 347.

### Output Data Descriptor FM Header

On output from IMS, the data descriptor FM header is used to send a data structure name and version ID or the output field tab separator character if OFTAB= parameter of the DIV or the DPAGE statement is specified.

It is sent in the only transmission chain of a nonpaged output message or in each transmission chain of a paged output message. For a nondemand-paged message (OPTIONS=MSG), this header follows the ATTACH or SCHEDULER FM header. For an autopaged message, this header follows the ATTACH or SCHEDULER FM header and precedes the data for the first logical or presentation page. The data descriptor FM header precedes the data for each additional page of output and is the only FM header in the transmission chain. For demand-paged output, this

header follows the QFXR header. The version ID is sent only once for each message in the first or only FM header for the output message. Additionally, if OFTAB is specified and OPTIONS=DNM is requested, the output field tab separator character used for the current transmission is included in the FM header. If OPTIONS=NODNM is specified, a data descriptor FM header will not be sent, regardless of whether OFTAB has been defined or not.

**Related Reading:** For the format of the output data descriptor FM header, see “Output Data Descriptor FM Header Format” on page 348.

## Controlling Demand-Paged Messages: QMODEL FM Headers

Queue model (QMODEL) headers are sent and received to control demand-paged messages. All demand-paging requests (except RAP) must be made using the QMODEL-defined FM headers.

**Restriction:** MFS DPM-Bn does not support operator control table requests or input messages while sending demand-paged output.

The QMODEL headers that IMS supports are listed in Table 35.

Table 35. IMS-Supported QMODEL Headers

	Sent by IMS <sup>1</sup>	Received by IMS <sup>2</sup>
QXFR	X	
QGETN		X
QGET		X
QPURGE		X
QSTATUS	X	

**Notes:**

1. QMODEL headers are not sent between two IMS subsystems.
2. A QMODEL FM header must precede a QMODEL page request for the first page. These FM headers can be preceded by the QMODEL ATTACH FM header with the DPN parameter containing the value X'03'.

QXFR is a QMODEL reply. All other headers are QMODEL requests. Each half session between the logical units might indicate, by using the session bind parameters, whether the QMODEL process is available to receive QMODEL requests. The bind parameters have no effect on QMODEL replies. IMS always indicates “QMODEL available” and is prepared to receive QMODEL (paging) requests to IMS demand-paged output. The other half session might or might not indicate “QMODEL available.” IMS ignores the other half-session bind indication for QMODEL requests, because only QMODEL replies can be sent by IMS.

Queue model uses LU 6.1 protocols to handle messages as follows:

- Although SNA allows multiple messages to be active, IMS messages are serial. That is, only one message is active at any given time. When active message processing is completed, another message can be processed. Synchronous (ATTACH without SCHEDULER) output MFS DPM demand-paged messages are considered active immediately when the first chain (ATTACH) is sent. Demand-paged output sent with ATTACH SCHEDULER does not become active until the first page request is returned to IMS.
- MFS demand-paged output provides for two types of message organization: linear (OPTIONS=DPAGE) and hierarchic (OPTIONS=PPAGE). For both types,

## Controlling Demand-Paged Messages: QMODEL FM Headers

pages can be retrieved sequentially (get next) or linearly (get by cursor). However, linear retrieval is permitted only if operator logical paging (OLP) is defined. Hierarchic retrieval is not supported.

**Related Reading:** For more information on the required bracket, send/receive, and response protocols required for QMODEL FM headers, see “Data Flow Control Protocol Reference” on page 295.

---

## Request (Input) QMODEL FM Headers

This section describes the QGETN, QGET, and QPURGE FM headers.

### QGETN FM Header

A QGETN FM header is received by IMS to cause sequential transfer of a single logical or presentation page of MFS DPM demand-paged output. It is the only QMODEL FM header in the chain and is never followed by function management data.

The QGETN FM header is not sent by IMS.

The QGETN FM header can be sent to IMS in the following cases:

- To retrieve first page of demand-paged output. A QGETN must always be the first request following the receipt of a scheduled demand-paged output. It must be preceded by an ATTACH QMODEL FM header.
- Following a QSTATUS FM header that indicates an invalid cursor in scheduled demand-paged output.
- When a non-OLP demand-paged output message is in progress.
- When an OLP demand-paged output message is in progress and the current cursor position is not at the last page of the message.

Each of the above cases causes a QXFR FM header to be returned in reply.

The QGETN FM header cannot be sent to IMS in the following cases:

- Between messages. An error message is returned.
- While an OLP demand-paged output message is in progress and the current cursor position is at the last page.

**Related Reading:**

- For more information on invalid cursor conditions, see “QSTATUS FM Header” on page 337.
- For information on the format of the QGETN FM header, see “QGETN FM Header Format” on page 350.

### QGET FM Header

Use the QGET FM header as input to IMS to transfer to the next, the last, or any logical page, according to cursor.

A QGET FM header is only valid for demand-paged output with operator logical paging defined. It is the only FM header in the input chain and is never followed by function management data.

The QGET FM header is not sent by IMS.



## Request (Input) QMODEL FM Headers

The QGET FM header cannot be sent to IMS in the following cases:

- Between messages. An error message is returned.
- After receiving the ATTACH for a scheduled demand-paged output message. The first paging request for this type of output must be QGETN. QGET can be used after requesting the first output page via QGETN.
- While a non-OLP demand-paged output message is in progress. An error message is returned. The output message is placed on the queue for retransmission.
- When the cursor value does not contain a valid 2-byte binary number. An error message is returned, and the output message is placed on the message queue for retransmission.
- When the cursor value is outside the range for the output message.

### **Related Reading:**

- For information on invalid cursor conditions, see “QSTATUS FM Header”.
- For information on format of the QGET FM header, see “QGET FM Header Format” on page 349.

## QPURGE FM Header

When IMS receives a QPURGE header, it halts processing on the demand-paged output message in progress. QPURGE can be received by IMS only while an MFS demand-paged output message is active. The QPURGE FM header is the only FM header in the chain and is never followed by function management data. IMS replies to a QPURGE FM header with a QSTATUS. If IMS receives a QPURGE header to delete the demand-paged output message, the message is deleted only after IMS receives a sync-point response in reply to QSTATUS.

The QPURGE FM header is not sent by IMS.

**Related Reading:** For information on the format of the QPURGE FM header, see “QPURGE FM Header Format” on page 351.

---

## Reply (Output) QMODEL FM Headers

This section describes the QXFR and QSTATUS FM headers.

### QXFR FM Header

The QXFR header is sent in each output transmission chain for a demand-paged output message in reply to a valid paging request by a QMODEL header. A QXFR FM header can be followed by the data descriptor (DD) FM header (based upon MFS format definitions) and carries logical or presentation page data. This header must not be sent to IMS.

**Related Reading:** For information on the format of the QXFR FM header, see “QXFR FM Header Format” on page 353.

### QSTATUS FM Header

A QSTATUS FM header is sent by IMS in reply to an input QPURGE FM header or because an invalid cursor was detected on a page request for a demand-paged output.

If a QSTATUS results from a QPURGE during demand-paged output, it is sent to request a sync point. The message is dequeued when the sync-point response is

## Reply (Output) QMODEL FM Headers

received. The message is returned to the queue and made available for retransmission if an exception sync-point response is received.

A QSTATUS resulting from an invalid cursor during scheduled demand-paged output is sent RQE1/CD and allows a subsequent QGETN to again request the first page of the message or a QPURGE to cause the message to be dequeued.

The QSTATUS FM header is the only FM header in the chain and is not followed by function management data. The QSTATUS FM header must not be sent to IMS.

**Related Reading:** For information on the format of the QSTATUS FM header, see “QSTATUS FM Header Format” on page 352.

---

## The RAP FM Header under MFS

When used by MFS, the reset attached process (RAP) FM header can be sent to IMS to delete the demand-paged output message in progress (prevent further processing). This header is equivalent to the MFS operator control function of NEXTMSG. This header can also be sent to IMS to delete the operator logically paged message in progress. A RAP FM header cannot be followed by function management data. The RAP FM header is not sent by IMS.

**Related Reading:** For information on the format of the RAP FM header, see “Reset Attached Process (RAP) FM Header Format” on page 355.

---

## Chapter 16. FM Header Format Reference

Each header's length is defined by a 1-byte length field. The value in this field includes the length field itself. A concatenation flag indicates whether an additional header follows. Within the header formats, variable- and fixed-length parameters are positional by command code. A 1-byte length field precedes each variable-length positional parameter. The value contained in this length field can be X'00' through X'08' and does not include the length field itself. If the length field contains X'00', the variable parameter is omitted and the next positional variable-length parameter length field occurs followed by its variable-length parameter field. Trailing positional parameter length fields of X'00's at the end of the header can be eliminated for input to IMS and are not sent on output from IMS. IMS also does not send trailing blanks for any of the above names.

When receiving a header, IMS ignores any coded parameters beyond the last parameter to which it is sensitive (that is, that IMS supports). The other LU 6.1 subsystem should do the same when receiving headers that IMS sends.

### In this Chapter:

- "ATTACH FM Header Format"
- "Data Descriptor FM Header" on page 347
- "Error Recovery Procedure FM Header" on page 349
- "QMODEL FM Headers" on page 349
- "Reset Attached Process (RAP) FM Header Format" on page 355
- "SCHEDULER FM Header Format" on page 355
- "SYSMSG Process Headers" on page 356

---

## ATTACH FM Header Format

The format of the ATTACH FM header is defined in Table 36.

Table 36. Attach FM Header Format

Byte	Bits	Name	Content
0		FMHL	
1	0	FMHC	
	1-7	FMHT	B'0000101'
2-3		FMH5CMD	X'0202'
4	0-3	FMH5MOD	
	4		Reserved
	5	ATTDP	Session Local Flag
			(B'0') Not demand-paging
			(B'1') Demand-paging
	6-7	ATTIU	Interchange Unit code
			(B'00')= multiple chain
			(B'01')= single chain
			(B'10')= reserved
			(B'11')= reserved
5		FMH5FXCT	Length of fixed-length parameters must be X'02'

## ATTACH FM Header Format

Table 36. Attach FM Header Format (continued)

Byte	Bits	Name	Content
6		ATTDSP	One-byte hexadecimal input value into a data stream profile. (IMS requires a value from X'00' to X'03' to select components 1-4 respectively. Value X'00', or component 1, is assumed if no ATTACH FM header is received while the Attach manager is in reset state.)
7		ATTDBA	Application data handling algorithm X'00': Undefined (Input only to IMS. IMS also assumes this as the default value if no ATTACH FM header is received while the attach manager is in reset state.) X'01': Variable length, variable blocked (input/output for IMS) X'02': Document subset of SCS (not supported by IMS) X'03': Card subset of SCS (not supported by IMS) X'04': A chain of RUs (input to/output from IMS) X'05': An RU (input only to IMS) X'06'-X'FF' Reserved
<i>Resource Names</i>			
8-m		ATTDPN <sup>1</sup>	Name of a process to be initiated. For an SNA-defined process, the DPN is a 1-byte, nongraphic hexadecimal character. Supported SNA processes are: X'01' - SYSMMSG X'02' - SCHEDULER X'03' - QMODEL  The IMS processes are described later under the ATTDPN parameter.
m+1-n		ATTPRN <sup>1</sup>	Name of primary resource for the process being initiated
n+1-p		ATTRDPN <sup>1</sup>	Name of suggested return process name
p+1-q		ATTRPRN <sup>1</sup>	Name of suggested primary resource for the return process
q+1-r		ATTDQN <sup>1</sup>	Name of queue to be associated with the attached process
r+1-s		ATTACC	Access code—Ignored by IMS.

**Notes:**

1. If the ATTDPN is set to SCHEDULER (X'02'), the remaining ATTACH parameters must not be specified. In this case, an equivalent for each of the asterisked parameters might occur in the concatenated SCHEDULER FM header. These SCHEDULER parameters are prefixed by "SCD" rather than by "ATT", but follow the same definition and rules as those for ATTACH.

**Related Reading:** For examples of using the ATTACH parameters, see "Appendix H. ISC Data Flow Control Examples" on page 539 and "Appendix F. Examples Using ISC Edit ATTACH Parameters" on page 489.

The following sections describe the parameters in the ATTACH FM header that are supported by IMS:

### ATTIU

The interchange unit code (ATTIU) indicates whether a single input or output IMS message consists of one or more SNA chain. All input messages that are not MFS-autopaged messages might indicate either single or multiple chain, but are restricted to one actual transmission chain indicating end-of-message. MFS-autopaged input messages might indicate multiple chain regardless of whether the message is actually one or more than one SNA chain. MFS also offers an option that permits single chain messages to produce multiple DPAGEs. Non-MFS and MFS output that is not autopaged is always sent as a single transmission chain with ATTIU indicating single chain. MFS-autopaged output messages are always sent with ATTIU indicating multiple chain.

**Restriction:** IMS does not support any relationship between consecutive input or output messages. Therefore, the interchange unit code indicating multiple chain cannot be used to send IMS batched input messages or consecutive messages to be scheduled using the SCHEDULER process.

IMS requires attached (synchronous) input message switches to be one chain with at least EB specified.

**Restriction:** IMS does not support MFS-autopaged synchronous input message switches of more than one SNA chain.

The ATTIU parameter is not automatically retained from one input or output message to another. This parameter must be explicitly provided with each input message, or by MFS if the output message is multichain and is formatted by MFS. The input reset state for ATTIU is “multiple chain.”

### ATTDSP

The data stream profile (ATTDSP) parameter is ignored when attaching QMODEL (ATTDPN=X'03'). During other input, the ATTDSP field indicates a specific IMS input component. The input component is used to identify an input IMS LTERM from a set of LTERMs (LTERM subpool) allocated to the session. The input DSP value must indicate a valid input component (an IMS LTERM defined with the input component value), or the input is rejected.

Several internal IMS functions are associated with the LTERM defined for the input component. For example, the LTERM defines a particular origin (input) and destination (output) path within IMS. This output destination can be the same as, or different from, the input origin LTERM and represents a stable application and operator reference point within IMS.

For statically defined terminals, input terminal security can be defined for the input LTERM. This security authorization can be unique for each LTERM or can represent a security level, or group, within IMS. Input terminal security authorizes access to IMS terminals, transactions, and commands.

The input ATTACH DSP parameter remains in effect until it is:

- Changed by another ATTACH
- Reset by a system failure
- Backed out to its value at the last sync point by the ERP

## ATTACH FM Header Format

- Reset at between-brackets state
- Reset by the RAP FM header

The input reset state for the ATTDSP parameter is X'00' or "component 1."

Because IMS does not remember the active input ATTDSP across IMS system failures, the ATTDSP must be provided using an ATTACH FM header to restore the receiving process after a bind or negotiable bind response indicating "in-brackets" (possible restart).

**Related Reading:** For more information on the restart process, see "Coordinating the Restart Process" on page 275.

The output LTERM is determined by either the LTERM selected for input or by the message sender, as for message switches and broadcast messages. IMS also allows the receiving message processing program (MPP) to change the destination of resulting output by issuing a CHNG call and inserting the output to a modifiable alternate PCB. For all system messages resulting directly from input (for example, commands), the output LTERM selected is the normal destination LTERM defined for the LTERM from which the input originated.

During output, the DSP field is set to the output component value associated with the output LTERM.

IMS Message Format Service can be defined as available on a component-by-component basis. The output component also specifies the bracket and send/receive protocol to be used for asynchronous output sent using the ATTACH having a DPN parameter indicating "SCHEDULER."

## ATTDBA

The deblocking algorithm (ATTDBA) determines the data blocking and deblocking algorithm to be used. This algorithm, along with an indication of whether MFS is being used, determines the amount of data presented to or from a process for a single get or put operation.

When IMS is sending messages and MFS is not being used, each segment of the message on the message queue becomes a data entity within the RU (or chain of RUs). If a segment of the message spans message queue records (LRECLs), then each spanned portion of the segment becomes a data entity. If MFS is used, the segments are blocked and sent as a data entity according to the definition of the MFS record.

When IMS receives messages, each data entity of the input message is inserted in the message queue as a segment of the message. Because ISC does not support spanned queue segments on input, the largest message queue LRECL must be large enough to handle the largest data entity or MFS data record received. If this LRECL definition is exceeded, an error is detected and an appropriate error message is produced (DFS074). Also, the actual space available in the message queue is reduced by any variable prefix items on the message.

**Related Reading:** For more information on prefix segments, see *IMS/ESA Installation Volume 1: Installation and Verification*.

IMS implements four of the algorithms defined by SNA:

**UNDEFINED**

Same as RU, below, for IMS.

**RU** Input RU is the entity presented to the attached process. IMS receives this DBA value but does not send it.

**VARIABLE LENGTH, VARIABLE BLOCKED (VLVB)**

Each data entity sent or received is preceded by a 2-byte length field that allows it independence from RU size or boundaries. Several data entities can be blocked into a single RU, or a data entity can span RUs. The length includes the 2 bytes for the length field itself.

**CHAIN OF RUs**

Each data entity is sent or received as a single SNA chain.

Each message received by IMS is deblocked based on the ATTDDBA field in the ATTACH FM header. The input ATTDDBA field value must indicate a supported algorithm, or the input is rejected.

The input ATTACH DBA parameter remains in effect until it is:

- Changed by another ATTACH
- Backed out to its value at the last sync point by the ERP
- Reset at between-brackets state
- Reset by the RAP FM header
- Reset by a session failure

The input reset state for the ATTDDBA parameter is “UNDEFINED” (equivalent to “RU”).

Because IMS does not remember the active input ATTDDBA across IMS system failures, the ATTDDBA must be provided using an ATTACH FM header to restore the receiving process after a bind or negotiable bind response indicating “in-brackets” (possible restart).

**Related Reading:** For more information on the restart process, see “Coordinating the Restart Process” on page 275.

With the exception of output using MFS stream mode, output from IMS is automatically sent indicating the variable-length, variable blocked (VLVB) algorithm in the ATTACH or SCHEDULER FM header. This output is sent indicating chain assembly.

**ATTDPN**

The destination process name (ATTDPN) parameter identifies, explicitly or implicitly, the input process to be attached to the half session. One responsibility of the attached process is to determine, for the receiving subsystem, the destination of the input message within that subsystem. In some cases, the named process might be the message destination. Within IMS, the receiving process (except basic edit) uses the input primary resource name (ATTPRN), if provided, as the message destination transaction queue or LTERM. If an ATTPRN is not provided, the standard IMS algorithm for determining message destination is used.

**Related Reading:**

- For more information on the ATTPRN parameter, see “ATTPRN” on page 345.

## ATTACH FM Header Format

- For more information on the IMS algorithm for determining message destination, see *IMS/ESA Administration Guide: System* under “IMS Messages and Their Scheduling.”

All attached processes execute synchronously with the session. However, some processes can schedule additional work to be done asynchronously within IMS. The process attached remains attached until it is:

- Changed by another ATTACH
- Backed out to its value at the last sync point by the ERP
- Reset by the end-bracket indicator
- Reset by the RAP FM header
- Reset by a system failure

The input reset state for the ATTDPN parameter is “ISCEDT”.

Because IMS does not remember the active input ATTDPN across IMS system failures, the ATTDPN must be provided using an ATTACH FM header to restore the receiving process after a bind or negotiable bind response indicating “in-brackets” (possible restart).

The ATTDPN parameter might indicate that the process to be attached is one of the following:

- ISC edit ('ISCEDT' or user-named alias)
- IMS Basic Edit ('BASICEDT')
- MFS formatting (MFS MID name)
- System Message (SYSMSG), X'01'
- SCHEDULER model, X'02'
- Queue model (QMODEL), X'03'

With the exception of MFS and QMODEL, these processes are always available. MFS is only available if defined for the input component (ATTDSP) during IMS system definition. If the ATTDPN is an MFS MID name and MFS is not available, the input message is rejected. QMODEL is available only following an MFS demand-paged output message.

ISCEDT (as well as the alias defined by the user for ISCEDT during IMS system definition) and BASICEDT are reserved names, and cannot be used as MFS MID or MOD names. Use of these names results in the use of the named process rather than an MFS process.

ISC edit is selected if no ATTACH FM header (or no ATTDPN) is supplied when the Attach manager is in reset state. The “active” process is used if no ATTACH FM header (or no ATTDPN) is supplied when the attach manager is not in the reset state.

IMS sets the output ATTDPN on a return reply to the value contained in the ATTRDPN parameter optionally provided within the ATTACH of the previous input request. When sending system messages, IMS automatically inserts the SYSMSG ATTDPN where necessary. MFS DPM provides a way to optionally specify, override, or delete the output ATTDPN by using the output message format descriptor. When the ATTDPN parameter is not available for output, a 1-byte field containing X'00' is included within the output ATTACH FM header.



**Related Reading:** For information on the ATTRDPN and ATTRPRN parameters, see “ATTRDPN and ATTRPRN” on page 346.

## ATTPRN

The ATTACH primary resource name (ATTPRN) parameter represents the destination for an input message in the receiving subsystem. This parameter is sent on a return reply from a remotely executed message that results from the returning reply message of the ATTACH ATTRPRN parameter that was sent on a message request. However, in situations where the message request contains no input ATTRPRN, the output ATTPRN parameter can be created using MFS, based on user-defined information.

IMS does not permit specification of IMS command verbs as PRNs.

This destination normally represents a terminal (an LTERM), rather than an application program. If the request ATTRPRN/reply ATTPRN represents an application program, consider the synchronous versus asynchronous relationship between the source application program and the remote transaction executions.

Further, you should consider input transaction security requirements for processing and routing the reply. If supplied on input to IMS, the ATTPRN is used as an IMS transaction code or as the LTERM name for a message switch. The data stream is not edited for destination and security information. If the ATTPRN represents a transaction, and if password security (which applies to statically defined terminals only) was defined for the node, an input security error results, because no available source exists for the input password.

**Related Reading:** For more information on password security, see *IMS/ESA Installation Volume 1: Installation and Verification*.

The ATTPRN is rejected during IMS conversation mode when an application has previously inserted the transaction code into the SPA. The ATTPRN only applies to a single message instance and overrides, but does not destroy, any “preset” destination established using an IMS /SET command. Subsequent messages with no ATTACH FM header or with an ATTACH FM header where the ATTPRN parameter is not supplied, can use any preset destination previously established by the IMS /SET command or by an IMS transaction code, LTERM name, or command verb supplied in the data or through MFS formatting.

IMS sets the output ATTPRN on a return reply to the value contained in the ATTRPRN parameter optionally provided within the ATTACH of the previous input request.

**Related Reading:** For more information on ATTRDPN and ATTRPRN, see “ATTRDPN and ATTRPRN” on page 346.

MFS DPM provides a means to optionally specify, override, or delete the output ATTPRN using an output message format descriptor. When this parameter is not present, a 1-byte field containing X'00' is sent with the output ATTACH FM header.

The ATTPRN is not automatically retained from one input or output message to another. This parameter must be explicitly provided with each input message, or through the output MFS descriptors if the output message is processed by MFS.

## ATTACH FM Header Format

### ATTRDPN and ATTRPRN

The return destination process name (ATTRDPN) and return primary resource name (ATTRPRN) parameters define the reply process and the return primary resource within the source session and should be returned to the source session on resulting replies to facilitate return-reply routing within the source session.

If provided on input to IMS and not changed or deleted by an output MFS format description, these parameters are returned to the source session unmodified in the reply ATTACH FM header as the output ATTDPN and ATTPRN parameters respectively. These parameters are associated with the message to be processed and are recovered with the message across session and subsystem failures.

The input ATTRDPN and ATTRPRN are not automatically retained from one input or output message to another. The ATTRDPN and ATTRPRN must be explicitly provided with each input message or through the MFS output format descriptors when MFS DPM is used for output. The ATTRDPN results from MFS if a next MID was specified in the MOD.

A different procedure is followed when the reply is to be returned on a session different from the session on which the input message originated. If the output ATTRPRN is not set by MFS, and the source of the output message is not associated with the same session, IMS does not wrap the input ATTRDPN and ATTRPRN as the output ATTDPN and ATTPRN respectively. In this case, IMS also automatically inserts the source LTERM name of the terminal entering the input message switch or transaction (for alternate PCB output) as the output ATTRPRN parameter.

When the ATTRDPN parameter is not available for output, a 1-byte length field containing X'00' is included in the output ATTACH FM header.

Table 37 and Table 38 summarize IMS actions relative to the DPN, PRN, RDPN, and RPRN fields in the ATTACH and SCHEDULER headers sent with the message.

Table 37. IMS Interpretations for the DPN, PRN, RDPN, and RPRN Fields

Input FMH	IMS Interpretation
DPN	MFS MID name or input message editor name (ISC edit or basic edit). Defaults to ISCEDT if not supplied.
PRN	Input transaction code or LTERM name, overriding data stream and preset mode except during conversation.
RDPN	Saved as default reply DPN field.
RPRN	Saved as default reply PRN field.

Table 38. IMS Actions for the DPN, PRN, RDPN, and RPRN Fields

Output FMH Field	IMS ACTION Reply Message Sent on Same Session as Input	IMS ACTION Reply Message or Asynchronous Output Sent on another Session
ATTDPN	Wrapped input ATTRDPN if provided; optionally overridden by MFS <sup>1</sup>	Provided only using MFS <sup>1</sup>
ATTPRN	Wrapped input ATTRPRN; optionally overridden by MFS	Provided only using MFS
ATTRDPN	Provided only using MFS	Provided only using MFS

Table 38. IMS Actions for the DPN, PRN, RDPN, and RPRN Fields (continued)

Output FMH Field	IMS ACTION Reply Message Sent on Same Session as Input	IMS ACTION Reply Message or Asynchronous Output Sent on another Session
ATTRPRN	Provided only using MFS	Automatically defaulted to source LTERM name; optionally overridden by MFS

**Notes:**

1. For ATTACH-only output, if no parameter is available, the DPN is not sent. For SCHEDULER output, the default is ISCEDT.

## ATTDQN and ATTDTP

The destination queue name (ATTDQN) parameter names a specific message instance. This parameter is valid only for output MFS demand-paged messages. For MFS demand-paged output, ATTDQN is sent by IMS as an output message identifier. This name can then be used by the other half session as the QNAME parameter within paging requests to access the IMS demand-paged output message. Within IMS, only one message can be active for a given session at any one time. This means that after output paging begins on a message and until the paging operation is terminated, all input must be paging requests indicating the same QNAME (from ATTDQN) value. The session-local flag (ATTDTP) in the ATTACH FM header is set to 1. ATTDQN and ATTDTP are not sent or received under other conditions.

## ATTACC

IMS does not send the access code (ATTACC) parameter during output, and ignores this parameter on input.

## Data Descriptor FM Header

The tables below show the formats of the input and output data descriptor FM headers.

### Input Data Descriptor FM Header Format

Byte	Bits	Name	Contents
0	0-7	FMHL	Length
1	0	FMHC	
	1-7	FMHT	B'0000100'
2	0-7	FMH4FXCT	Fixed-length parm (X'03')
3		FMH4DTYP	
	0-7		
		FMH4UNDF	X'00' Reserved
		FMH4FIX	X'40' Field formatted record (FFR) - Fixed
		FMH4FXSP	X'41' FFR - All fields terminated by Separator (Note 1)
		FMH4MXSP	X'42' FFR - Fields terminated by Separator or Length defined by map X'43' - X'FF' Reserved (Note 1)

## Data Descriptor FM Header

Byte	Bits	Name	Contents
4		FMH4SEP	Separator character (Note 2)
5	0-7	FMH4PCTL	Presentation control byte (Note 3)
6-m		FMH4DSN	Data structure name (Note 4)
m+1-n		FMH4BDT	Block data type (Note 3)
n+1-p		FMH4VERS	Version ID (Note 5)

### Notes:

1. From DIV or DPAGE statement.
2. Not required, because it is received on MFS format description.
3. Ignored by IMS if sent.
4. DPAGE name if DPAGE selection is to be performed on the data structure name. FMT name if OPTIONS=MSG. Required field.
5. Version ID to be used by MFS to verify that the correct definition is used to map the data. This field should be present only once for each input message if multiple transmission chains are used to create the input message. If present once for each message, it should be sent in the first transmission chain of the message. If the version ID is not sent or it is X'0000', MFS descriptor verification is not performed.

## Output Data Descriptor FM Header Format

Byte	Bits	Name	Contents
0	0-7	FMHL	Length
1	0	FMHC	
	1-7	FMHT	B'0000100'
2	0-7	FMH4FXCT	Fixed-length parm (X'03')
3		FMH4DTYP	
	0-7		
		FMH4UNDF	X'00' Reserved
		FMH4FIX	X'40' Field formatted record (FFR) - Fixed
		FMH4FXSP	X'41' FFR - All fields terminated by separator
		FMH4MXSP	X'42' FFR - Fields terminated by separator or length defined by map X'43' - X'FF' Reserved
4		FMH4SEP	Separator character
5	0-7	FMH4PCTL	Presentation control byte (Note 1)
6-m		FMH4DSN	Data structure name (Note 2)
m+1-n		FMH4BDT	Block data type (Note 3)
n+1-p		FMH4VERS	Version ID (Note 4)

Byte	Bits	Name	Contents
<b>Notes:</b>			
1. Set to 0 on output.			
2. FMH4DSN is:			
<ul style="list-style-type: none"> <li>• The FMT name if OPTIONS=MSG</li> <li>• The DPAGE name if OPTIONS=DPAGE</li> <li>• The PPAGE name if OPTIONS=PPAGE</li> </ul>			
3. Omitted if version ID not present. Set to 0 if version ID present.			
4. Version ID specified in FMT definition or calculated by MFS if the default is used. This field is present only in the first or only data descriptor FM header for the message.			

## Error Recovery Procedure FM Header

The format of the ERP FM header is as follows:

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHTYPE	X'07'
2-5		ERPSENSE	SNA sense code that would appear on error response
6-7		ERPSEQ	Sequence number of chain for which error was detected

## QMODEL FM Headers

The beginning of each header is fixed-length. This fixed-length area is followed by a variable number of fixed-length subfields, followed by a variable number of variable-length subfields.

## QGET FM Header Format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command Code: X'0A10'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte Length Fields
	1-6		Reserved
	7	QGETLAST	B'1' Coded request, get last <sup>1</sup>
5		FMH6FXCT	Length of Fixed-Length Parameters (=X'01')

## QMODEL FM Headers

Byte	Bits	Name	Contents
6		QORG	Type of paging request X'00' Queue organization not specified. Valid on input to IMS. X'01' Sequential. Invalid. X'02' Linear. Valid on input to IMS if it matches the QORG in the QXFER sent by IMS. <sup>2</sup> X'03' Hierarchic. Not supported by IMS. X'04' - X'FF' Reserved
7-m		QNAME	Q from which records are retrieved. IMS message ID (ATTDQN/SCDDQN) <sup>3</sup>
m+1-n		QCURSOR	Cursor vector in target queue <sup>4</sup>
n+1-p		QTRNSZ	Size of maximum record to be returned (binary). Ignored by IMS if sent. All fixed-length parameters must be present when you use this header.

### Notes:

- The QGETLAST selects a logical page whose value is equal to the last logical page in the message.  
The QGET FM header is valid and causes a QXFR reply only if the OLP demand-paged output message is in progress and the cursor value requested is within the range of the output message.
- If QORG is invalid, an error message is sent and the output message is returned to the message queue for retransmission.
- The QNAME (message identifier) is required on each page for scheduled demand-paged output and is optional for synchronous demand-paged output. If the QNAME parameter is present, it must match the ATTDQN/SCDDQN parameter in the ATTACH FM header sent by IMS. If QNAME is not specified for synchronous demand-paged output, IMS assumes it to be the same as the ATTDQN parameter name. If the QNAME parameter does not match the ATTDQN/SCDDQN parameter, an error message is sent and the output message is returned to the message queue for retransmission.
- The cursor vector consists of one 2-byte binary number representing the logical page number. IMS does not support hierarchic retrieval; therefore, the 2-level cursor (logical page number followed by the presentation page number) is invalid and results in an error message being issued. The length value of the QCURSOR is 2. The next 2 bytes contain the logical page number (absolute) request.

## QGETN FM Header Format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command Code: X'0A10'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte Length Fields
	1-7		Reserved
5		FMH6FXCT	Length of Fixed-Length Parameters (=X'01')

Byte	Bits	Name	Contents
6		QORG	Type of paging request X'00' Queue organization not specified. This form is valid for both OLP and non-OLP output. X'01' Sequential. <sup>1</sup> X'02' Linear. <sup>1</sup> X'03' Hierarchic. Not supported by IMS. X'04' - X'FF' Reserved
7-m1		QNAME	Q from which pages are retrieved. Specifies a message ID (ATTDQN/SCDDQN) <sup>2</sup>
m+1-n		QTRNSZ	Size of maximum record to be returned (binary). Ignored by IMS if sent.

**Notes:**

1. If QORG is specified, the QORG specified on the resulting QXFR FM header sent by IMS must match it. Otherwise, an error message is sent and the output message is returned to the message queue for retransmission.
2. The QNAME (message identifier) is required on each page request for scheduled demand-paged output and is optional for synchronous demand-paged output. If the QNAME parameter is present, it must match the ATTDQN/SCDDQN parameter in the ATTACH FM header sent by IMS. If QNAME is not specified for synchronous demand-paged output, IMS assumes it to be the same as the ATTDQN parameter name. If the QNAME parameter does not match the ATTDQN/SCDDQN parameter, an error message is sent and the output message is returned to the message queue for retransmission.

## QPURGE FM Header Format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command Code: X'0A06'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte Length Fields
	1-7		Reserved
5		FMH6FXCT	Length of fixed-length parameters (=X'01')
6		QORG	Type of queue purge function (if multiple DPAGE input) or paging request (if demand-paged output) <sup>1</sup>
7-m		QNAME	Name of IMS message ID (ATTDQN/SCDDQN) <sup>2</sup>

## QMODEL FM Headers

Byte	Bits	Name	Contents
<b>Notes:</b>			
1. If IMS receives QPURGE during demand-paged output, the QORG, if specified (that is, QORG is not equal to 0), must match the QORG specified in the QFXR FM header sent by IMS.			
2. The QNAME (message identifier) is required on each page request for scheduled demand-paged output and is optional for synchronous demand-paged output. If IMS receives QPURGE during demand-paged output, the QNAME, if specified, must match the ATTDQN/SCDDQN parameter (message ID) in the ATTACH FM header sent by IMS. If QNAME is not specified for synchronous demand-paged output, IMS assumes the same name as the ATTDQN parameter name. If the QNAME parameter does not match the ATTDQN/SCDDQN parameter, an error message is sent and the output message is returned to the message queue for retransmission.			

## QSTATUS FM Header Format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command code: X'0A0A'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte length fields
	1-7		Reserved
5		FMH6XCT	Length of fixed-length parameters (=X'02')
6		QORG	Type of queue add/retrieval requests. 1
7	0-4		Not supported by IMS. Bit values should be: B'0'.
	5	QINVCUR	B'0' if message is a reply to QPURGE. B'1' Invalid Cursor Indicates invalid logical page request received for OLP output.
	6-7		Not supported by IMS.
8			Reserved
9-12		QCURSOR	Current cursor value is 0 <sup>2</sup>
13-16		QSENSE	Sense data <sup>3</sup>
17-n		QNAME	Message ID (ATTDQN/SCDDQN)

### Notes:

1. If QSTATUS is sent by IMS in reply to the QPURGE, QORG is not specified (that is, QORG contains the value of zero). If QSTATUS is sent by IMS in reply to QGET by cursor request, QORG is set to linear (that is, QORG contains the value of two).
2. This parameter is present only if the QINVCUR value is set to B'1'. The cursor value is set to 0. A QGETN request retrieves the first page of the message for which this error (invalid cursor) was detected.
3. This parameter is present only if the QINVCUR value is set to B'1'. It consists of the DFS223 error message as a binary number.



## QXFR FM Header Format

Byte	Bits	Name	Contents
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	Type: B'0000110'
2-3		FMH6CMD	Command Code: X'0A08'
4		FMH6MOD	Modifier
	0	FMH6LNSZ	B'0' 1-Byte Length Fields
	1-7		Reserved
5		FMH6FXCT	Length of fixed-length fields (=X'02')
6		QORG	Types of paging requests valid for this message. X'01' Sequential retrieval of pages. Operator logical paging (OLP) is not defined. Message is dequeued after successful transmission of last page. X'02' Linear retrieval of pages. Operator logical paging (OLP) or browsing is allowed. Message is dequeued only by explicit action from the other subsystem (RAP or QPURGE FM header is received). X'03' Hierarchic retrieval of pages. Not supported by IMS. X'00', X'04'-X'FF' Reserved
7	0-4	Reserved	
	5	QDISP	B'0' Disposition is save.
	6		Reserved.
	7	QEMSG	B'0' Not end of message B'1' End of message Set to B'1' if last logical page (OPTIONS=DPAGE) or last presentation page (OPTIONS=PPAGE) of a message is transmitted.
8-n		QCURSOR <sup>1</sup>	Cursor vector for current message. For OPTIONS=PPAGE, this field includes a 1-byte length field of value 4 followed by the 2-byte current logical page number and the 2-byte current presentation page number. For OPTIONS=DPAGE, this field includes a length field of value 2, followed by the current logical page number. <sup>2</sup>

## QMODEL FM Headers

Byte	Bits	Name	Contents
n+1-n		QCOUNT <sup>1</sup>	Number of occurrences of pages at lowest level of cursor. If present, this field includes a length byte of value 2, followed by the defined number of pages in the current logical page. The QCOUNT field is present only if the defined number of presentation pages in the current logical page is greater than 1.
n+1-p		QRECLNG	Length of record before truncation. If 0, then either record was not truncated or QTRNSZ on QGET(N) was ignored. This field is not supported by IMS and will not be included in the FM header.

### Notes:

1. If operator logical paging (OLP) is allowed, the QCURSOR and QCOUNT fields contain information the receiver can use for formulating subsequent QGET by cursor requests.
2. Assume the following output message with OPTIONS=PPAGE:
 

Logical Page 1	3	Presentation Pages Defined
Logical Page 2	1	Presentation Page Defined
Logical Page 3	2	Presentation Pages Defined

The following illustrates the values placed in the QCURSOR and QCOUNT fields with sequential retrieval requests:

Transmission	Count/LP	No./PP No.	Qcursor Count/PP No.	Qcount Comment
1	4 / 1	1	2 / 3	Logical Page 1
2	4 / 1	2	2 / 3	Logical Page 1
3	4 / 1	3	2 / 3	Logical Page 1
4	4 / 2	1	(omitted)	Logical Page 2
5	4 / 3	1	2 / 2	Logical Page 3
6	4 / 3	2	2 / 2	Logical Page 3

### Note:

Output Message with OPTIONS=DPAGE:

Logical Page 1  
Logical Page 2  
Logical Page 3

The following illustrates the values placed in the QCURSOR field with sequential retrieval requests:

Transmission	Qcursor Count/LP	Qcount (Not Present)	Comment
1	2 / 1		Logical Page 1
2	2 / 2		Logical Page 2
3	2 / 3		Logical Page 3

---

**Reset Attached Process (RAP) FM Header Format**

Byte	Bits	Name	Contents
0		FMHL	
1	0	FMHC	
	1-7	FMHT	B'0000101'
2-3		FMH5CMD	X'0204'
4		FMH5MOD	
	0	FMH5LNSZ	B'0': 1-byte parameter length field
	1-7		Length of parameter length fields B'0': 1-byte parameter length field B'1': (Reserved)
5		FMH5FXCT	Length of fixed-length parameters (X'00')

---

**SCHEDULER FM Header Format**

Byte	Bits	Name	Contents
0		FMHL	
1	0	FMHC	
	1-7	FMHT	B'0000110'
2-3		FMH6CMD	X'0802'
4		FMH6MOD	
	0	FMH6LNSZ	B'0' 1-byte parameter length field
	1	FMH6RPLY	B'0' No reply; B'1' Reply <sup>1</sup>
	2	FMH6PROT	B'0' No protection; <sup>2</sup> B'1' Protection
	3	FMH6DELY	B'0' Timer optional; B'1' Timer required; <sup>1</sup>
	4-7		Reserved
5		FMH6FXCT	Length of fixed-length parameters - X'01'
6			Schedule Request control
	0	SCDTIME	Initiation delay specification B'0' Interval B'1' Time Not supported by IMS
	1-7		Reserved
7-m		SCDDPN <sup>3</sup>	Name of process to be initiated. (Required field)
m+1-n		SCDPRN <sup>3</sup>	Name of the primary resource for the process to be initiated
n+1-p		SCDRDPN <sup>3</sup>	Suggested name of the return process
p+1-q		SCDRPRN <sup>3</sup>	Suggested name of the primary resource for return process name
q+1-r		SCDDQN <sup>3</sup>	Name of queue associated with process
r+1-s		SCDREQN	Name of this process request instance (Not supported by IMS)

## SCHEDULER FM Header Format

Byte	Bits	Name	Contents
s+1-t		SCDDELY	Time interval to be decremented prior to initiation of destination process. (Not supported by IMS)

### Notes:

1. Must be 0 on both input and output.
2. On input, this field is ignored. On output, this field is set to 0 for nonrecoverable output and to 1 for recoverable output.
3. See sections 343 through "ATTDQN and ATTDTP" on page 347 for a more detailed description of these parameters. In the ATTACH section, these parameters are prefixed by "ATT" rather than "SCD". The rules for operation of these SCHEDULER parameters are the same as the rules defined for the ATTACH parameters.

## SYMSMSG Process Headers

### SYSEERROR FM Header Format

Byte	Bits	Name	Contents
0		FMHL	
1	0	FMHC	
	1-7	FMHT	B'0000110'
2-3		FMH6CMD	X'0404'
4		FMH6MOD	
	0	FMH6LNSZ	B'0': 1-byte parameter length field
	1-7		Reserved
5		FMH6FXCT	Length of fixed-length parameters (X'00')
Optional Variable Fields			
6-m		ATTDPN	RDPN field supplied with an input message resulting in this SYMSMSG
m+1-n		ATTPRN	RPRN field supplied with an input message resulting in this SYMSMSG

### SYSSTAT FM Header

Byte	Bits	Name	Content
0		FMHL	Length
1	0	FMHC	
	1-7	FMHT	B'0000110'
2-3		FMH6CMD	X'0402'
4		FMH6MOD	
	0	FMH6LNSZ	B'0' 1-Byte length fields
	1-7	Reserved	
5		FMH6FXCT	Length of fixed-length parameters

---

## Part 5. Administering IMS in a Client/Server Environment

<b>Chapter 17. CPI Communications</b> . . . . .	359
CPI-C Driven Application Programs . . . . .	359
SAA Resource Recovery Commit Processing . . . . .	359
Normal Termination . . . . .	360
Backout Processing. . . . .	360
Abnormal Termination . . . . .	360
Session Failure . . . . .	361
Return Codes . . . . .	361
System Restart/Resolve-in-Doubt Processing . . . . .	361
CPI-C Application Program Recovery . . . . .	362
Programming Requirements . . . . .	362
Pseudonym Files . . . . .	362
Distributing SYNC-Point/Protected Conversations. . . . .	362
Introducing RRS/MVS . . . . .	363
The Two-Phase Commit Protocol. . . . .	363
Local-Resource Recovery Versus Distributed-Resource Recovery. . . . .	364
IMS as a Resource Manager . . . . .	365
Activating Protected Conversations . . . . .	365
APPC as the Communications Manager . . . . .	366
Using OTMA with Protected Conversations . . . . .	366
XRF and Protected Conversations . . . . .	366
RSR and Protected Conversations . . . . .	366
<b>Chapter 18. Administering APPC/IMS and LU 6.2 Devices</b> . . . . .	367
Introducing APPC/IMS. . . . .	367
APPC/IMS Application Program Interface . . . . .	368
Implicit API . . . . .	368
Explicit API . . . . .	369
APPC/IMS Application Programs . . . . .	369
Standard IMS Application Programs. . . . .	369
MSC and Standard IMS Application Programs . . . . .	370
Modified IMS Application Programs . . . . .	371
MSC and Modified IMS Application Programs . . . . .	371
CPI Communications Driven Application Programs . . . . .	372
Using the MOD Name and LTERM Interface . . . . .	373
Establishing APPC/IMS . . . . .	374
TP_Profile . . . . .	374
MVS System Data File Manager Utility (ATBSDFMU) Example . . . . .	376
Side Information — Outbound . . . . .	377
PARMLIB Member . . . . .	378
APPC/MVS Timeout Service . . . . .	379
APPC/MVS Error Extract Service. . . . .	380
Initializing LU 6.2 Descriptors . . . . .	380
Using MSC in an APPC/IMS Environment . . . . .	381
Recovering APPC Transactions in an MSC Environment . . . . .	382
Recoverable versus Irrecoverable Transactions . . . . .	382
Local APPC Transaction Discardability versus Nondiscardability . . . . .	382
OTMA Transaction Discardability versus Nondiscardability . . . . .	383
Transaction Processing Point of Failure . . . . .	383
Recovering Transactions after an LU 6.2 Session Failure . . . . .	383
Recovering Transactions after an MSC Link Failure . . . . .	384
Recovering Transactions after a Local IMS Failure . . . . .	384
Recovering Transactions after a Remote IMS Failure . . . . .	385

Recovering Transactions after an Intermediate IMS Failure . . . . .	385
Recovering Transactions after an Application Program Failure . . . . .	385
Recoverability Flows of LU 6.2 Transactions . . . . .	386
Planning for XRF . . . . .	388
Transaction Retry Characteristics. . . . .	389
Qualifying Network LU Names. . . . .	389
DFSAPPC System Service . . . . .	390
Message Switching . . . . .	390
Asynchronous Output Delivery. . . . .	393
APPC Transaction Security . . . . .	394

---

## Chapter 17. CPI Communications

This chapter introduces CPI Communications driven application programs and distributed SYNC-POINT protected conversations.

### In this Chapter:

- “CPI-C Driven Application Programs”
- “Distributing SYNC-Point/Protected Conversations” on page 362

---

### CPI-C Driven Application Programs

A CPI Communications driven application program can use IMS-managed resources in two ways:

- Using the SQL calls to access DB2 through the IMS External Subsystem (ESS) Attach Facility. If the DB2 resource translation table (RTT) is not used, the initial DB2 plan name is the application program name. After the APSB call, the DB2 plan name is the PSB name specified in the APSB call.
- Using the APSB call to allocate IMS resources.

You can use the following SAA resource recovery calls when you want an application program to commit or back out changes to IMS or DB2 resources:

- Use the Commit call (SRRCMIT) to commit changes.
- Use the Backout call (SRRBACK) to back out changes.

CPI Communications driven application programs:

- “SAA Resource Recovery Commit Processing”
- “Normal Termination” on page 360
- “Backout Processing” on page 360
- “Abnormal Termination” on page 360
- “Session Failure” on page 361
- “Return Codes” on page 361
- “System Restart/Resolve-in-Doubt Processing” on page 361
- “CPI-C Application Program Recovery” on page 362
- “Programming Requirements” on page 362

### SAA Resource Recovery Commit Processing

By issuing the SRRCMIT call, an application program tells IMS to commit changes to database resources:

- Issue the SRRCMIT call when the application program updates any IMS resources or accesses DB2 resources.
- Reissue the SRRCMIT call after making any subsequent changes to an IMS or DB2 resource.
- Issue the SRRCMIT call before terminating your application program.

If the application program terminates with any uncommitted changes to IMS resources, IMS attempts to commit these changes. Dangling conversations are deallocated abnormally.

## SAA Resource Recovery Commit Processing

When you issue the SRRCMIT call, IMS gets control and generates an internal sync-point call (if the conversation was not allocated with SYNCLVL=SYNCPT). All database changes are committed. All messages inserted to alternate PCBs (program control blocks) are sent to their final destination.

## Normal Termination

In IMS, normal termination occurs when an application program terminates without abending. For a CPI Communications driven application program, an implicit commit occurs.

**Definition:** An *implicit commit* occurs when the application program does not issue an explicit call to commit the current transaction, but one of the following occurs:

- The application program terminates normally.
- For a standard or modified DL/I application program, the application program issues a GU call to the message queue.

## Backout Processing

Transaction updates are backed out when any of the following occurs:

- Backout is issued by the application program.
- The application program terminates abnormally.
- The application program terminates normally, but the implicit commit fails.
- IMS resources are inflight during IMS system restart.

The backout consists of the following actions:

- All database updates are backed out.
- All messages inserted to non-express alternate PCBs are discarded.
- All messages inserted to express PCBs that have not been enqueued are discarded.
- The APPC/MVS verb ATBCMTP TYPE=ABEND is issued.<sup>31</sup>

The application program tells IMS that a backout is required by issuing SRRBACK or by terminating abnormally.

## Abnormal Termination

IMS considers an application program to have terminated abnormally if either of the following occurs:

- The implicit commit fails.
- The application program abends.

When your application program abends, IMS backs out to the last IMS sync point.

---

31. Issuing the verb ATBCMTP causes all LU 6.2 conversations associated with this TPI to terminate with CM\_DEALLOCATE\_ABEND.



## Session Failure

If any LU 6.2 session fails during the conversation, you can choose to end the application program or continue processing. IMS TM is not involved and places no restrictions on your choice of committing or backing out updates. The application programmer makes this decision.

Because IMS TM is not informed of the session failure, it takes no action. The normal processing rules for commit and backout apply.

## Return Codes

Your application program receives return codes from IMS on the SAA resource recovery SRRCMIT and SRRBACK calls.

Your application program can receive the following return codes:

### RR\_OK

The backout or commit operation completed successfully. All protected resources if backed out have been returned to their previous sync point; if they have been committed, they have advanced to a new sync point, and all changes made during the logical unit of work have been made permanent.

### RR\_PROGRAM\_STATE\_CHECK

A non-CPI Communications driven IMS application program issued an SAA resource recovery Commit call. No commit or backout has been performed.

### RR\_BACKED\_OUT

A resource manager voted “no” during sync point processing. The sync point was initiated by the SAA resource recovery Commit call. The resource state is backed out for all resources.

## System Restart/Resolve-in-Doubt Processing

**Definition:** After a system failure, a key part of restart processing is known as *resolve-in-doubt* processing. If the system fails, IMS determines whether to perform resolve-in-doubt processing for IMS-protected resources. Examples of IMS-protected resources are:

- IMS DB databases
- DB2 databases
- IMS TM message-queue messages

If the IMS system fails before the transaction completes phase one of the two-phase commit process (sync point), IMS backs out during IMS restart. Backout includes transactions that were processing at the time of failure.

If the transaction completes phase one of the commit process, resolve-in-doubt processing can take place during IMS restart. If only IMS resources are affected, commit processing occurs. If DB2 resources are involved, resolve-in-doubt processing occurs between IMS and DB2.

No transactions using explicit CPI Communications driven application programs are preserved across an IMS restart.

## CPI-C Application Program Recovery

### CPI-C Application Program Recovery

No recovery processing exists for application programs using the explicit CPI Communications driven interface. IMS discards all CPI Communications driven transactions at restart regardless of their state at the time of failure. Application program designers should be aware of the SAA resource recovery resynchronization functions and consider the impact on their application program designs.

The application program should provide full integrity by issuing a SAA resource recovery Commit or Backout call for session failures. Application programs that require recovery assistance must be standard DL/I application programs.

**Related Reading:** For more information on IMS DL/I calls, see *IMS/ESA Application Programming: Database Manager*.

### Programming Requirements

The CPI Communications driven application program activates IMS sync point processing by issuing the SRRCMIT and SRRBACK calls. The calls that initiate implicit sync point (DL/I GU to the message queue, CHKP, and SYNC) are invalid for CPI Communications driven application programs, and receive status AD (function parameter invalid).

If you allocate a conversation with SYNCLVL=NONE or SYNCLVL=COMMIT, include module DFSCPIR0 with your application program in the link-edit step. Including this module allows your application program to resolve the external references for SRRCMIT and SRRBACK.

No language-unique programming concerns exist in IMS for the SAA resource recovery interface.

### Pseudonym Files

APPC/IMS uses APPC/MVS services to provide SAA resource recovery support. APPC/MVS does not provide SAA resource recovery pseudonym files. However, you can create your own pseudonym files.

**Related Reading:** For sample pseudonym files, see *SAA CPI Resource Recovery Reference*. These sample pseudonym files include examples on how to define working storage in the different languages.

Briefly, programmers of different languages need to define the following:

- IMS TM C programmers need to define MVS as their operating system.
- IMS TM COBOL programmers must define their buffers in working storage.
- IMS TM FORTRAN programmers must define EXTERNAL statements for SRRCMIT and SRRBACK.

---

### Distributing SYNC-Point/Protected Conversations

- “Introducing RRS/MVS” on page 363
- “Activating Protected Conversations” on page 365

## Introducing RRS/MVS

With OS/390, Release 3, a new system resource recovery platform is used. Resource Recovery Services/MVS (RRS/MVS) is the sync-point manager, coordinating the update and recovery of multiple protected resources. RRS/MVS controls how and when protected resources are committed by coordinating with the resource managers, such as IMS, that have registered with RRS/MVS.

RRS/MVS supports the Common Programming Interface for Resource Recovery (CPI-RR), an element of the SAA CPI that specifies resource recovery and provides coordination for recovering local and distributed resources.

### Definitions:

- A *protected resource* is a set of local or distributed data that is updated in a synchronized and controlled manner. In the APPC environment, a protected resource is a resource that is updated in an allocated conversation in which SYNCLVL=SYNCPT has been specified.
- A *resource manager* is a product, such as IMS, that owns protected data resources that are updated in an APPC conversational environment in which SYNCLVL=SYNCPT has been specified. IMS acts as a resource manager for DL/I data, Fast Path data, and the message queues.

The three participants in resource recovery include:

- RRS/MVS (sync-point manager)
- Resource manager (such as IMS or DB2)
- Application program

Figure 51 shows the three participants in the resource recovery process, and their interaction.

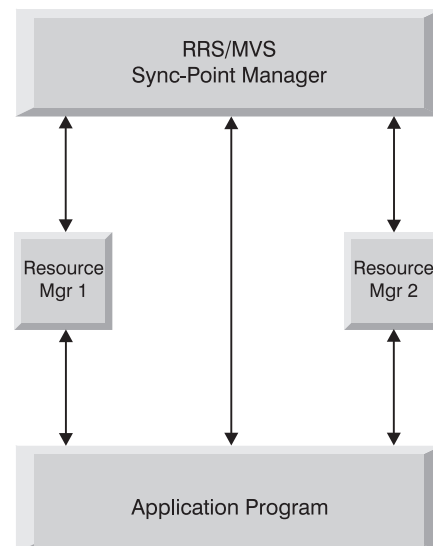


Figure 51. Participants in Resource Recovery

### The Two-Phase Commit Protocol

**Definition:** The *two-phase commit protocol* is a process involving RRS/MVS and the resource manager that ensures that updates made to a set of resources by an application program are either all made or none made. The application program decides whether to commit its changes to the resources; this commit is made to

## Introducing RRS/MVS

RRS/MVS, which polls all of the resource managers as to the feasibility of the commit call. Each resource manager votes whether to commit the updates. This is called *phase 1* of the two-phase commit protocol.

After RRS/MVS has gathered the votes, *phase 2* begins. If all votes are to commit the updates, then the phase 2 action is to commit; otherwise, phase 2 results in a backout of the updates. System failures, communication failures, resource manager failures, or application program failures are not barriers to completing the two-phase commit protocol.

### Definitions:

- A *unit of recovery* is a unit of work that spans one commit (synchronization) point to the next commit point.
- Units of recovery are termed *inflight* between the time they are created (or the previous sync point) until the resource manager votes to commit the updates. If the resource manager fails while units of recovery are inflight, the resource manager backs out all of the database updates on the subsequent start.
- Units of recovery are termed *indoubt* between the time when the resource manager votes to commit the updates and the time when the sync-point manager calls the resource manager to do the commit. If IMS fails while units of recovery are indoubt, IMS holds the database updates until they are resolved.

### Local-Resource Recovery Versus Distributed-Resource Recovery Definitions:

- In a *local-resource recovery environment*, the recovery participants reside on the same MVS system.
- In a *distributed-resource recovery environment*, the recovery participants and the updated resources are scattered across multiple systems. In a distributed-resource recovery environment, the APPC/PC (APPC/protected conversation) resource manager is used to provide the communications for the sync-point calls to remote systems.

Figure 52 illustrates how a distributed recovery environment operates.

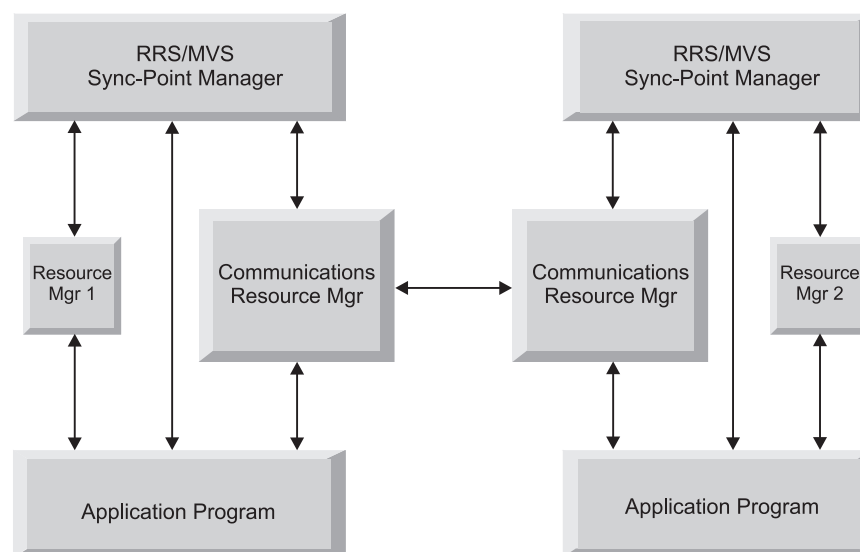


Figure 52. Distributed Resource Recovery

### IMS as a Resource Manager

In this environment, a resource manager controls a protected resource. In general, a resource manager does the following:

- Provides an application programming interface (API) to allow application programs to access its resources
- Logs changes to data before making the changes permanent
- Logs unit of work status
- Participates in the commit or backout actions for updated resources
- Contains recovery mechanisms to restore data to a previous state

For its own resources, IMS is both the sync-point manager and the resource manager.

Within the two-phase commit protocol, IMS must do each of the following:

- Register with RRS/MVS as a resource manager
- Participate in the sync-point process
- Express interest in the unit of work
- Recover its unit of work status after an outage

**Registration:** A component of RRS/MVS provides registration services so that IMS can identify itself as a resource manager. By registering, IMS is provided a set of services to aid in maintaining resource consistency associated with the protected conversation.

IMS registers each time a control region is started for a DB/DC active system on an MVS system with the recovery platform support. In an XRF environment, the active system registers during its restart. The alternate system registers at the time of takeover. No registration or participation occurs from an RSR tracking system.

**Expressing Interest:** In addition to registering and supplying resource manager exit routines for specific stages of the two-phase commit protocol, IMS must also express interest in participating in the two-phase commit process for a particular unit of recovery.

**Resolution of Incomplete Interests:** In the event of an IMS or MVS outage, during the IMS restart, the incomplete UR expressions of interest must be resolved.

RRS/MVS maintains unit of recovery information (such as identifier, state, and resource manager private data), which RRS/MVS presents to the restarting resource managers that previously expressed interest.

**Sync-Point Participation:** After IMS successfully registers and restarts, it supplies addresses of its exit routines to RRS/MVS. Several exit routines (such as prepare, commit, and backout) represent specific points in the two-phase commit protocol, which IMS can call to participate in the process.

## Activating Protected Conversations

MVS uses a new construct with RRS/MVS called a context.

**Definition:** A *context* is the entity for which resource managers perform services, to which they allocate resources and lock ownership, and in which they can express interest in participating in the protocol to ensure that the resource is updated in an orderly manner.

## Activating Protected Conversations

The type of context that the resource manager creates, owns, and manipulates is called the *private context*. A resource manager can create a context on behalf of another resource manager. RRS/MVS uses the private context to identify an application program's unit of work to maintain information for the resource manager concerning which of their resources are associated with the unit of work.

### APPC as the Communications Manager

When APPC is the communications manager, RRS/MVS support is activated when a conversation is allocated with SYNCLVL=SYNCPT. This type of conversation is a protected conversation.

When SYNCLVL=SYNCPT is specified, APPC acquires a private context on behalf of IMS. IMS provides its resource manager name to APPC in its identity call. APPC provides the private context to IMS as the message header. IMS, using this context, then assumes the role of a participant in the two-phase commit process with the sync-point manager, RRS/MVS.

In addition to the SYNCLVL=SYNCPT, the keyword ATNLOSS=ALL must be specified in the VTAM definition file for whichever LUS the user wishes to enable for protected conversations.

### Using OTMA with Protected Conversations

In an OTMA environment, OTMA is not a resource manager registered with RRS/MVS. The process remains an inter-process protocol between a server (IMS) and a number of clients (application programs). Therefore, OTMA cannot obtain a private context token to pass to IMS, as APPC does. The client-adaptor code that uses OTMA is responsible for obtaining and owning a private context, and for providing the context ID. In messages passed between the partners, the context-ID field contains the context token (if it is a protected conversation).

When IMS finds the context-ID in the message, IMS assumes the role of a participant in the two-phase commit process, as it does in the APPC environment.

**Related Reading:** For more information on these OTMA topics, see *IMS/ESA Open Transaction Manager Access Guide*.

### XRF and Protected Conversations

Running protected conversations (using RRS/MVS with either APPC/PC or OTMA) in an IMS-XRF environment does not guarantee that the alternate system can resume and resolve any unfinished work started by the active system. A failed resource manager must re-register with its original RRS/MVS system if the RRS/MVS system is still available when the resource manager restarts. Only if the RRS/MVS on the active system is not available can an XRF alternate system register with another RRS/MVS in the Sysplex and obtain the incomplete unit of recovery data of the failing active system. Because IMS retains indoubt units of recovery until they are resolved, it is recommended that a switch back to the active system be done as soon as possible to obtain the unit of recovery information and to resolve and complete all the work of the resource managers.

### RSR and Protected Conversations

Active systems tracked in an RSR environment by a remote system can process protected conversations with APPC/PC or OTMA, although it will be necessary to resolve in-doubt units of recovery using commands after a takeover. Most likely, the remote site is not part of the active Sysplex, and no means exists for the new IMS system to acquire unfinished unit of recovery information from RRS/MVS. IMS provides commands to investigate protected conversation work and to resolve it if necessary.

---

## Chapter 18. Administering APPC/IMS and LU 6.2 Devices

This chapter introduces APPC/IMS, and describes how to administer APPC/IMS and LU 6.2 devices.

### **In this Chapter:**

- “Introducing APPC/IMS”
- “APPC/IMS Application Program Interface” on page 368
- “APPC/IMS Application Programs” on page 369
- “Establishing APPC/IMS” on page 374
- “Initializing LU 6.2 Descriptors” on page 380
- “Using MSC in an APPC/IMS Environment” on page 381
- “Recovering APPC Transactions in an MSC Environment” on page 382
- “Planning for XRF” on page 388
- “Transaction Retry Characteristics” on page 389
- “Qualifying Network LU Names” on page 389
- “DFSAPPC System Service” on page 390
- “APPC Transaction Security” on page 394

---

## Introducing APPC/IMS

APPC/IMS, a part of IMS TM, lets you use the CPI Communications interface to build CPI application programs. APPC/IMS allows distributed and cooperative processing between IMS and systems that have implemented APPC. APPC/IMS delivers support for APPC with facilities provided with APPC/MVS.<sup>32</sup> APPC/IMS supports the CPI resource recovery Commit (SRRCMIT) and Backout (SRRBACK) calls for IMS-managed local resources. These protected resources include:

- IMS TM message-queue messages
- IMS DB databases
- DB2 databases

APPC/IMS also supports the existing IMS DL/I application programming interface (API) enabling application programs to use LU 6.2 communications without the function of the CPI Communications interface. This allows most existing applications to continue to function with the APPC/IMS support of LU 6.2.

### **Definitions:**

- In this book, “transaction programs”, “applications”, “application programs”, and “programs” are synonymous.
- In this chapter, “APPC application programs” are synonymous with “LU 6.2 application programs”.
- “LU 6.2 transactions” are those that originate from an LU 6.2 application program.

---

32. The APPC/IMS interface is provided by APPC/MVS and supports the CPI Communications interface. IMS/ESA TM supports the CPI resource recovery interface.

## Introducing APPC/IMS

Figure 53 illustrates the protocol boundaries and the IMS functional entities that provide LU 6.2 support.

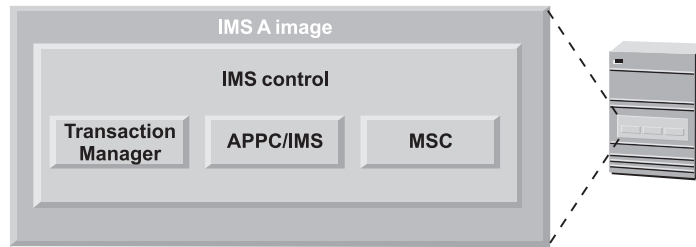


Figure 53. Functional Support Boundaries

**Recommendations:** For APPC/IMS, do the following:

- Schedule your IMS standard or modified application programs entered from LU 6.2 remote systems using MSC. Be aware that CPI-C driven application programs cannot have transactions that execute on remote systems.
- Define your APPC/IMS LUs for use by APPC/MVS, as well as by any APPC application program.
- Use the LTERM and the MOD name in the first segment of your message. Use the LTERM to change the destination for your output to a non-LU 6.2 device. Use the MOD name to format error messages.
- Use a network-qualified LU name. You do not need to have unique names for the LUs on different systems.

IMS dependent regions are automatically defined to APPC as subordinate address spaces of the IMS Scheduler. An IMS BMP cannot be defined as an ASCH controlled application. It may use explicit conversation services through the IMS base LU.

IMS manages the APPC/IMS buffers automatically; no definition is necessary. No special considerations are needed for EMH.

---

## APPC/IMS Application Program Interface

APPC/IMS IMS TM has two distinct application program interfaces (APIs): the implicit and explicit interfaces. The same application program can use both APIs.

### Implicit API

The *implicit API* is an extension of the IMS standard DL/I API (call xxxTDLI). It allows IMS application programs to communicate with LU 6.2 application programs without being sensitive to LU 6.2 protocols and without requiring the programmer to have any knowledge of LU 6.2. APPC/IMS provides functions not normally available to an LU 6.2 application program: message queuing, and automatic asynchronous message delivery and recovery. Existing IMS transactions use the implicit API to communicate with APPC.

Implicit API messages are placed on the IMS message queues or the Fast Path expedited message handling (EMH) buffers for Fast Path transactions. The originating IMS system determines whether to mark the input messages as discardable or nondiscardable.



## APPC/IMS Application Program Interface

When the implicit API is used, IMS issues all required CPI Communications calls. The application program interacts strictly with the IMS message queues or the Fast Path EMH buffers.

### Explicit API

The *explicit API* is the CPI Communications API and is available to any IMS application program.<sup>33</sup> The application program makes calls to APPC using the CPI Communications interface without using IMS. These CPI calls are handled directly by APPC/MVS. Messages sent or received by the CPI Communications interface are not stored on the IMS message queues or in the EMH buffers, and these messages are not available for transaction restart. No IMS-provided functions are involved for these messages.

---

## APPC/IMS Application Programs

APPC/IMS has three different types of application programs:

### Standard

No explicit use of CPI Communications facilities.

### Modified

Uses the I/O PCB to communicate with the original input terminal. Uses CPI Communications calls to allocate new conversations and to send and receive data.

### CPI Communications driven

Uses CPI Communications calls to receive the incoming message and to send a reply on the same conversation. Uses the DL/I APSB call to allocate a PSB to access IMS databases and alternate PCBs.

You can schedule your standard and modified application programs locally and remotely using MSC. The logic for local application programs differs from the logic for remote application programs. In the following sections, the differences are described.

## Standard IMS Application Programs

Standard IMS application programs use the existing IMS call interface. Application programs that use the IMS standard API can take advantage of the LU 6.2 protocols. Standard IMS application programs use a DL/I GU call to trigger a sync point and to get the incoming transaction. These standard IMS application programs also use DL/I ISRT calls to generate output messages to the same or different terminals, which can be LU 6.2 terminals.<sup>34</sup> The identical program can work correctly for both LU 6.2 and non-LU 6.2 terminal types. IMS generates the appropriate calls to APPC/MVS services.

IMS provides the following services for standard IMS application programs:

- Receives incoming transaction from an LU 6.2 application program
- Calls the Input Message Routing exit routine
- Schedules transactions into local and remote IMS dependent regions
- Provides necessary transaction recoverability
- Provides necessary transaction rollback and retry

---

33. You can also use MVS ATBxxx call services. See *MVS/ESA Programming: Writing Transaction Programs for APPC/MVS* for information on using these calls.

34. A non-message-driven BMP is considered a standard IMS application program when it does not use the explicit API.

## APPC/IMS Application Programs

- Provides integration of IMS-controlled conversation flows with database updates during sync point for all APPC Sync\_Level options (NONE, CONFIRM, SYNCPT)
- Provides all needed LU 6.2 calls to APPC/MVS services
- Sends either synchronous or asynchronous output to an LU 6.2 application program
- Keeps asynchronous output on IMS message queue until successful transmission
- Allocates new LU 6.2 conversations for messages inserted to alternate PCBs using the DL/I ISRT call

Existing application programs that are sensitive to a terminal's hardware characteristics, such as cursor position or MFS format names, might need to be changed to communicate with LU 6.2 application programs.

### **Restrictions:** LU 6.2 Synchronous Conversations with Implicit Transactions.

1. If a LU 6.2 synchronous conversation implicit transaction initializes other transactions (program-to-program switch), an express PCB can not be used to do the ISRT. An express PCB will cause race conditions to occur and the output may randomly return to the inputting terminal on a new asynchronous conversation with TPNAME DFSASync. The original conversation may not be deallocated.
2. If a transaction initializes more than one child transaction, which in turn may initialize other transactions, and one of the child transactions provides the response, the result is unpredictable.

Depending on the execution sequence of these transactions the LU can receive a DFS2082 message with the response sent to the default TP name DFSASync or the LU receives the response and no DFS2082 message is issued.

## MSC and Standard IMS Application Programs

When an APPC application program enters an IMS transaction that executes on a remote IMS, an LU 6.2 conversation is established between the APPC application program and the local IMS system. The local IMS system is considered the partner LU of the LU 6.2 conversation. The transaction is then queued on the remote transaction queue of the local IMS. From this point on, the transaction goes through normal MSC processing. After the remote IMS system executes the transaction, the output is returned to the local IMS system, and is then delivered to the originating LU 6.2 application program.

The originating (local) IMS system provides the following services:

- Receives incoming transaction from an LU 6.2 application program
- Calls the Input Message Routing exit routine
- Queues the transaction to its remote transaction queue
- Sends the transaction across the MSC link
- Receives the transaction response
- Sends either synchronous or asynchronous output to an LU 6.2 application program

The remote IMS system provides the following services for the remote standard application program:

- Receives the incoming transaction from the partner IMS (originating or intermediate IMS) over the MSC link

## APPC/IMS Application Programs

- Schedules transactions into dependent regions
- Commits database changes at sync point
- Provides necessary transaction recoverability
- Provides necessary transaction rollback and retry
- Keeps transaction output on the IMS message queue until the transmission is successful
- Returns the transaction output to the local IMS system over the MSC link

**Restriction:** MSC is not supported if the originating LU 6.2 conversation is allocated with SYNCLVL=SYNCPT.

## Modified IMS Application Programs

Modified IMS application programs use a DL/I GU call to retrieve the incoming transaction and to trigger a sync point. These modified IMS application programs also use DL/I ISRT calls to generate output messages to the same or different terminals, which can be LU 6.2 terminals.<sup>35</sup> Unlike standard IMS application programs, modified IMS application programs use CPI Communications calls to allocate new conversations, and to send and receive data. IMS has no direct control of these CPI Communications conversations.

Modified IMS transactions are indistinguishable from standard IMS transactions until program execution. In fact, the same application program can be a “standard IMS” application on one execution, and a “modified IMS” application on a different execution. The distinction is simply whether the application program has used CPI Communications resources.

IMS provides the following services for modified IMS application programs:

- Receives incoming transactions from LU 6.2 application programs
- Schedules transactions into local and remote dependent IMS regions
- Appropriate transaction recoverability before transaction scheduling
- Provides integration of IMS-controlled conversation flows with database updates during sync point for APPC Sync\_Level options (NONE, CONFIRM, SYNCPT)
- Provides all necessary LU 6.2 calls to APPC/MVS services for IMS-controlled LU 6.2 conversations
- Sends either synchronous or asynchronous output to LU 6.2 application programs
- Keeps asynchronous output on the IMS message queue until successful send occurs
- Allocates new LU 6.2 conversations for any messages inserted to alternate PCBs using the DL/I ISRT calls

IMS does not provide any services for conversations explicitly allocated by the application program. Explicitly allocated conversations need to be deallocated if a program abend occurs.

## MSC and Modified IMS Application Programs

When an APPC program enters an IMS transaction that executes on a remote IMS system, an LU 6.2 conversation is established between the APPC program and the local IMS system. The local IMS is considered the partner LU of the LU 6.2 conversation. The transaction is then queued on the local IMS system’s remote

---

35. A non-message-driven BMP is considered a modified standard IMS application program when it uses the explicit API.

## APPC/IMS Application Programs

transaction queue. From this point on, the transaction goes through normal MSC processing. After the remote IMS system executes the transaction, the output is returned to the local IMS system, and then delivered to the originating LU 6.2 program.

The originating (local) IMS system provides the following services:

- Receives incoming transaction from an LU 6.2 application program
- Calls the Input Message Routing exit routine
- Queues the transaction to its remote transaction queue
- Sends the transaction across the MSC link
- Receives the transaction response
- Sends either synchronous or asynchronous output to an LU 6.2 application program

The remote IMS system provides the following services for the remote modified application program:

- Receives the incoming transaction from the partner IMS (originating or intermediate system) over the MSC link
- Schedules transactions into dependent regions
- Appropriate transaction recoverability before transaction scheduling
- Commits database changes at sync point
- Provides necessary transaction recoverability
- Provides necessary transaction rollback and retry
- Keeps transaction output on the IMS message queue until the transmission is successful
- Returns the transaction output to the local IMS system over the MSC link

**Restriction:** MSC is not supported if the originating LU 6.2 conversation is allocated with SYNCLVL=SYNCPT.

## CPI Communications Driven Application Programs

CPI Communications driven application programs are defined only in the APPC/MVS TP\_Profile data set; they are not defined to IMS. Their definition is dynamically built by IMS when a transaction is presented for scheduling by APPC/MVS based on the APPC/MVS TP\_Profile definition after IMS restart. The definition is keyed by TP name. APPC/MVS manages the TP\_Profile information.

When a CPI Communications driven transaction program requests a PSB, the PSB must already be defined to IMS using the APPLCTN macro for SYSGEN and using PSBGEN/ACBGEN when APPLCTN PSB= is specified. When APPLCTN GPSB= is specified, a PSBGEN/ACBGEN is not required.

CPI Communications driven application programs must use CPI Communications calls to accept the incoming conversation and to send a reply on the same conversation. The DL/I GU call is not used to retrieve the initiating transaction from the LU 6.2 application program. No IMS resources are allocated when the application program is scheduled. Instead, the application program can use the DL/I APSB call to allocate a PSB that provides access to IMS databases and to alternate PCBs. A CPI Communications driven application program can send messages to other terminals (either LU 6.2 or non-LU 6.2) or other IMS transactions

## APPC/IMS Application Programs

(either local or remote) by inserting to an alternate PCB, after allocating the appropriate PSB. Both the explicit and implicit API can be used on the same application program.

IMS provides the following services for CPI Communications driven application programs:

- Schedules the transaction.  
IMS does not receive input before scheduling. It does not interact with the conversation at any time other than to possibly reject the inbound allocate request. If IMS rejects the inbound allocate request, the transaction is not scheduled.
- Provides sync point of local resources
- Schedules PSB if called by application program
- Processes calls to alternate or database PCB made by the application program

### **Related Reading:**

- For more information on RRS/MVS, see “Introducing RRS/MVS” on page 363.
- For more information on CPI Communications driven application programs, see:
  - “Chapter 17. CPI Communications” on page 359
  - *IMS/ESA Application Programming: Database Manager*
  - *IMS/ESA Application Programming: Transaction Manager*
  - *IMS/ESA Application Programming: Design Guide*

## Using the MOD Name and LTERM Interface

Your LU 6.2 application program can use an interface to emulate MFS. For example, the application program can use the MOD name to communicate with IMS to specify how an error message could be formatted. For non-LU 6.2 application programs, the MOD name is given to the MFS formatting modules in IMS; for LU 6.2 application programs, the MFS modules are not called, and the MOD name is given to the LU 6.2 Edit exit routine (DFSLUEE0) as a parameter. The LU 6.2 Edit exit routine can do whatever the programmer specifies with the MOD name, such as format an error message.

Your LU 6.2 application program uses the LU name to send data to an LU 6.2 application program. However, if you want to send data to a non-LU 6.2 device such as a printer, you can use the LTERM instead of the LU name.

The Initialization exit routine (DFSINTX0) can be used to create a user table of MOD names that you might want to use for formatting messages, and LTERMs that you might want to use as printers. This user table can be used by DFSLUEE0 to find the appropriate MOD name or LTERM for your application program.

LU 6.2 application programs can send both the LTERM and the MOD name in the first segment of the message. The LU 6.2 Edit exit routine (DFSLUEE0) checks the contents of the first message segment. Based on the information it finds in a user table, the exit routine decides whether to return the LTERM and the MOD name to IMS. IMS saves the LTERM and the MOD name in the I/O PCB. For formatting output, IMS provides the address of the MOD name in the first segment of the message to the LU 6.2 Edit exit routine (DFSLUEE0). For changing the destination to a non-LU 6.2 device, IMS provides the LTERM in the first segment of the message to the LU 6.2 Edit exit routine (DFSLUEE0). The Initialization exit routine

## APPC/IMS Application Programs

(DFSINTX0) can be used to create the user table. This exit routine must pass the address of the user table to IMS, and IMS passes the address to DFSLUEE0.

---

## Establishing APPC/IMS

Before activating APPC/IMS, an IMS system definition is needed to specify MVS 4.2 as the third parameter of the SYSTEM keyword for the IMSCTRL macro. CPI Communications driven application programs and LU 6.2 application programs cannot be defined in a system definition. LU 6.2 application programs are only defined to VTAM.

Start APPC/IMS by specifying APPC=Y on the IMS startup parameter. The default is APPC=N. When 'N' is specified, a connection to APPC/MVS services is not established during IMS initialization. When 'Y' is specified, IMS establishes a connection with APPC/MVS during IMS initialization. The /START APPC command overrides APPC=N.

## TP\_Profile

The TP\_Profile is a VSAM data set owned by APPC/MVS and maintained by the MVS System Data File Manager utility (ATBSDFMU) or by the administrator using TSO/ISPF dialogs. The purpose of the TP\_Profile entries is to provide attribute information for TP names.

**Related Reading:** For more information on this utility, see *MVS/ESA Programming: Writing Transaction Programs for APPC/MVS*.

CPI Communications driven application programs must be defined in the APPC/MVS TP\_Profile. IMS system-defined transaction codes can optionally be defined in a TP\_Profile. The existence of an IMS definition (in the IMS GEN or by online change) causes the transaction to be considered a standard DL/I or modified-standard application.

The TP\_Profile, an APPC/MVS resource, contains definitions of transaction program names (TPNs) and their characteristics. You can define a TP\_Profile to schedule an IMS transaction program that uses a transaction code that is different from the TPN.

IMS uses the TP\_Profile to establish transaction scheduling characteristics for CPI Communications driven application programs. Based on the IMS dependent section of the APPC/MVS TP\_Profile definition, IMS dynamically defines these characteristics when a transaction is presented for scheduling after restart by APPC/MVS.

CPI Communications driven transaction programs are defined only in the TP\_Profile. The definition is by TPN. The same TPN can be defined differently for different LU names by using a different TP\_Profile data set. The LU name is associated with an IMS system image.

The default TP\_Profile data set name is SYS1.APPCTP. The LUADD TPDATA option in the SYS1.PARMLIB (APPCPMxx) member specifies the TP\_Profile data set name used for this LU.

Use TP\_Profile dialog or the MVS System Data File Manager utility (ATBSDFMU) to define a TP\_Profile.

**Example:** Figure 54 is an example of the IMS-specific area of the TP\_Profile definition (Panel 1):

```

----- IMS TP_Profile Panel -----
TP Name . . . : INQUIRY_Part
Transaction Code . . . . . PART
Security Type . . . . . ____ (NONE,CHECK,FULL, default=CHECK)

CPI Communications Driven Options
Transaction Class . . . ____ (Range 1 - 255, default=1)
Maximum Regions . . . ____ (Range 0 - 255, default=1)

Comments . . . (Optional 1 to 10 lines)
> TP_PROFILE Created 10/8/91 <
> Access IMS Sample Parts DATABASE via. program DFSSAM02 <
> _____ <
> _____ <
> _____ <
> _____ <
> _____ <
> _____ <
> _____ <
> _____ <
> _____ <
> _____ <

PF01 = Help    PF03 = Exit    PF12 = Cancel    Enter = Accept
  
```

Figure 54. IMS-Specific TP\_Profile Dialog Panel 2

To maintain IMS TP\_Profiles using ISPF, do each of the following:

1. Enter TSO ICQASRM0 from the TSO command line of the TSO/E to start the ISPF TP\_Profile System Data Facility Maintenance Utility from a TSO user ID. If this utility is not available, contact your MVS system programmer.
2. Enter **S** next to the TP\_Profile selection and the TP\_Profile data set name specified on the TPDATA keyword on the LUADD statement for your IMS LU. (The LUADD statement is in the APPCTPxx parmlib member, where xx is the APPC suffix.)
3. A list of TP\_Profiles is displayed. Select **A** to add a new TP\_Profile or **E** to edit an existing TP\_Profile. If you are adding a TP\_Profile, you must supply a scheduler name. This name was set at IMS installation time. The recommended name is IMS.
4. After the general TP\_Profile characteristics are supplied, the ISPF editor panel is displayed. Enter DFSTPROF on the command line to display the IMS TP\_Profile Maintenance panel.
5. Supply IMS scheduler-dependent characteristics. Press enter to save your changes or PF3 or PF12 to cancel your changes. You can press PF1 for online help on fields supplied in this panel.

The TP\_Profile name is not available on all releases of TSO/E, so a value of “Not Available” is displayed. This does not suggest that a problem exists.

**Example:** Figure 55 on page 376 is an example of the IMS-specific area of the TP\_Profile definition (Panel 2):

## Establishing APPC/IMS

```
----- IMS TP_Profile Help Panel -----

Use this panel to fill in the IMS Scheduler dependent data necessary for an
APPC/MVS TP_Profile entry. This maps an APPC TP name (1-64 bytes, case
to an IMS Transaction Code (1-8 bytes, case independent). A profile
Security Type can be specified (default CHECK). This security type is used
when the IMS global security option is set to PROFILE.

Supply initial transaction class and maximum region values for CPI
Communications Driven transactions. IMS only uses these for CPI Communications
Driven transactions on the transaction's first execution. If the values
change on subsequent allocates, the IMS system's values are not affected.

Comments describing this TP_Profile entry can consist of one to ten lines.
Useful data can be: last date changed, reason for change, and name of person or
group making the change. Note that the (") character cannot be kept in the
comments. It is automatically translated to a (') before saving the profile
information.

For further information on the IMS related TP_Profile, see the IMS/ESA
Data Communications Administration Guide.

PF01 = Help      PF03 = Exit
```

Figure 55. IMS-Specific TP\_Profile Dialog Panel 2

## MVS System Data File Manager Utility (ATBSDFMU) Example

Following is an example of an MVS System Data File Manager utility (ATBSDFMU) entry:

```
TPADD TPSCHED_EXIT(DFSTPPE0)
      TPNAME(INQUIRY_PART)
      SYSTEM
      ACTIVE(YES)
      TPSCHED_DELIMITER(##)
      TRANCODE=PART
      CLASS=1
      MAXRGN=1
      ##
```

In this example, the IMS section starts with TRANCODE=PART. The other control statements are shown for completeness.

**Related Reading:** For more information on using the MVS System Data File Manager utility (ATBSDFMU), see *OS/390 V1R1 MVS Planning: APPC Management*.

The IMS TP\_Profile parsing module, DFSTPPE0, performs validity checking and parses the input data in IMS. This module should be loaded into the STEPLIB data set of the step that adds the TP\_Profile. The MVS System Data File Manager utility (ATBSDFMU) requires that STEPLIB be APF authorized.

The following four keywords are used to add an IMS section to the TP\_Profile entry. The keyword-parameter sets must be separated by one or more blanks. The keyword-parameter sets must be specified between columns 1-72. An asterisk (\*) in column 1 indicates a comment.

### **TRANCODE= 1 - 8 characters**

Name of the IMS transaction code associated with this TP name consisting of alphanumeric or '#', '\$', '@'. IMS translates the TP name to the TRANCODE.



IMS scans for valid characters (00640 character set). If invalid characters exist, IMS uses the default transaction code, IMSTRAN, instead of a transaction code with the non-00640 characters.

### **CLASS= 1 - 255**

Specifies the class used for scheduling. The default value is 1.

**Recommendation:** Define CPI transactions with a different message class from that used for non-CPI transactions. IMS handles all CPI transactions as priority zero within the transaction class.

### **MAXRGN= 0 - 255**

Restricts the number of dependent regions that this CPI Communications driven transaction program can use. The default value is 1.

### **RACF=NONE, CHECK, or FULL**

RACF=NONE causes IMS to call the Transaction Authorization exit routine (DFSCTRNO).

RACF=CHECK causes IMS to call RACF for security checking when IMS receives a transaction (using RCLASS of TIMS or CIMS), but does not clone the security environment into the dependent region when the transaction executes.

RACF=FULL clones the security environment to the dependent region at execution time. Specifying this parameter and issuing the IMS command /SECURE APPC PROFILE enables APSB SAF Security for this CPI-C application program.

You can use the TP\_Profile entry in two ways:

- To specify an IMS transaction code that is defined in the IMS system. The CLASS and MAXRGN parameters in the TP\_Profile are ignored and the transaction values in IMS remain unchanged. The TP\_Profile entry provides mapping for a 64-character TPN into an 8-character transaction code.
- To specify an IMS transaction code that is not defined in the IMS system. The IMS transaction code is a CPI Communications driven transaction, and is used as the load module name of the scheduled application program and the dynamically built transaction name.

When a TP\_Profile is not defined, IMS uses the first 8 bytes of the TPN translated to the IMS character set as the transaction code.

The allocate request is rejected if the transaction code is not valid.

## Side Information — Outbound

### **Definitions:**

- APPC/MVS *side information* supplies destination information, such as the name of the partner program, the name of the LU at the partner's node, and the logon mode name. CPI Communications provides a way to use system-defined values for these required fields; these system-defined values are the side information. This information can be used by an IMS application program allocating (establishing) an APPC conversation using CPI Communications, an IMS LU 6.2 descriptor, a DL/I change call (CHNG), or a DFSAPPC message switch. System programmers supply and maintain the side information for CPI Communications programs.

## Establishing APPC/IMS

- Side information is accessed by a *symbolic destination name*. The symbolic destination name, called *sym\_dest\_name* in this book, corresponds to an entry in the side information file containing the following information:

### partner LU name

Shows the name of the LU where the partner program is located. This LU name is any name for the remote LU that is recognized by the local LU for allocating a conversation. An example is a USERVAR name.

This LU name can be a 17-byte network-qualified LU name.

### logon mode name

Used by LU 6.2 to designate the properties for the session that will be allocated for the conversation. The properties include the class of service to be used on the conversation. The network administrator defines a set of mode names used by the local LU to establish sessions with its partners. The system programmer uses one of these values in a side table entry. An invalid mode name prevents a conversation from being allocated.

### TP name

*Transaction program (TP) name* specifies the name of the remote application program.

IMS and MVS do not accept blank *sym\_dest\_name* values on the Initialize\_Conversation call.

**Related Reading:** For more information on APPC calls, see *Common Programming Interface Communications Specification*.

The default name for the side information file is SYS1.APPCSI. Define this file in the SYS1.PARMLIB(APPCLxx).

```
SIDEINFO
  DATASET(SYS1.APPCSI)
```

The destination name, partner LU name, mode name, and TP name can be defined using the MVS System Data File Manager utility (ATBSDFMU).

```
SIADD
  DESTNAME(DESTX)
  TPNAME(LU62USER_TPX)
  MODENAME(APPCLxx)
  PARTNER_LU(APPCLxx)
```

## PARMLIB Member

The APPC address space uses the APPCLxx member of SYS1.PARMLIB. Define IMS as a local APPC component LU that is controlled by the APPC address space. The scheduler name is the same IMSID used in the IMSCTRL macro. When IMS identifies to APPC, it passes its IMSID as the scheduler name SCHED(IMS1) in APPC member APPCLxxx. The following is an example of the APPCLxx member:

```
LUADD
  ACBNAME(IMSLU62)
  SCHED(IMS1)
  BASE
  TPDATA(SYS1.APPCLxx)
  TPLEVEL(SYSTEM)
```

For XRF add:

USERVAR=uservar\_name ALTLU=luname

The LUADD option keywords are defined below:

**ACBNAME**=local LUNAME of IMS

**SCHED**=IMS id

**BASE** mandatory parameter

**TPDATA**(TP\_Profile dataset name)

**TPLEVEL**(system) suggested value

**USERVAR**=(uservar\_name)

**ALTLU**=(LUNAME)

**Related Reading:** For more information on these keywords, see *OS/390 MVS Planning: APPC Management*.

Communication between VTAM and its application programs requires an ACB (application control block) whose name must be identically defined in the SYS1.VTAMLST APPL statement and in the APPCPMxx LUADD statement ACBNAME parameter.

APPC manages the IMS ACB. When IMS identifies to APPC, APPC gives IMS the name of the APPC-managed ACB name (LUNAME). The APPC LUNAME is not defined in IMS, because IMS does not manage the ACB. The entries in the SYS1.PARMLIB member APPCPMxxx include both the IMS scheduler name (IMSID) and the LUNAME ACBNAME(xxxxxxx) that ties an IMS to an LUNAME.

This ACBNAME must be different from the ACBNAME used by IMS for non-LU 6.2 terminals. APPC/MVS expects its LUs to be defined as VTAM resources so that these LUs can access the SNA network. A VTAM application program (APPL) definition macro must be coded for each APPC/MVS LU. LU 6.2 application programs are only defined to VTAM, not to IMS. The SYS1.VTAMLST member example follows:

```
IMSLU62 APPL ACBNAME=IMSLU62
        APPC=Y
        ...
```

### APPC/MVS Timeout Service

When APPC/MVS doesn't respond to an APPC call that is issued under the dependent region task control block (TCB) due, for instance, to a network delay, the dependent region hangs and the application transaction program cannot regain control. Using APPC/MVS Timeout Service, you can indicate the maximum time interval an application waits before terminating a conversation and regaining control from APPC/MVS callable services. The timeout feature is activated at startup by including the APPCIOT=xx parameter in IMS proclib member DFSDCxxx, where xx specifies the number of minutes (between 0–1440) before the transaction is terminated. When a transaction is terminated due to timeout, messages DFS1965E

## Establishing APPC/IMS

and DFS1995E are sent to the MTO terminal and the MVS console. The timeout value can be changed using the /CHANGE command.

For synchronous APPC conversations, if APPC timeout is active, then IMS uses ATBSTO5 service (SET\_TIMEOUT\_VALUE) to set the timeout value for each conversation.

For asynchronous APPC conversations, if APPC timeout is active, then IMS sets the timeout value when the conversation gets allocated. In either case, the timeout value is active until the conversation is deallocated, which occurs, in the case of IMS conversational transactions, when the IMS conversation ends.

Common Programming Interface Communications (CPI-C) transactions are not automatically supported by APPC/MVS Timeout service, but can exploit APPC/MVS Timeout service using ATBSTO5 service provided the proper coding is supplied.

**Related Reading:** For more information on activating APPC/MVS Timeout Service, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

For more information on programming MVS services, see *MVS Programming: Writing Transaction Programs for APPC/MVS*

## APPC/MVS Error Extract Service

Whenever an APPC/MVS service call returns an unexpected return code, IMS issues APPC/MVS Error Extract Service call ATBEES3 with a DFS1995E prefix.

For more information on ATB return codes, see *OS/390 Messages and Codes*.

---

## Initializing LU 6.2 Descriptors

LU 6.2 descriptors are optional, but are required if you want to dynamically create queue control blocks and define processing options. LU 6.2 descriptors allow the system programmer to specify an LTERM that associates an output destination with an LU 6.2 application program. This allows the system programmer to change the application program's destination using alternate PCBs to LU 6.2 application programs, without requiring any application program coding changes. The application program uses an LTERM name as a symbolic destination; only the system programmer needs to be aware of the actual term associated with this name.

The LU 6.2 descriptor entry contains:

- LTERM name
- LU name of the destination of the LU 6.2 application program (overrides side information); this can be a network-qualified LU name up to 17 bytes in length
- APPC/MVS side information entry name; this parameter can be omitted
- VTAM mode table entry to be used (overrides side information)
- TP name to be scheduled (overrides side information)
- APPC conversation type (BASIC or MAPPED)
- APPC Sync\_Level options (NONE, CONFIRM)

Do not code a parameter and leave it blank (such as SIDE=b), or an error message is issued. Instead, omit the parameter completely.

**Related Reading:** For more information on coding these parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

These LU 6.2 descriptor LTERMs are only for output and are never used by IMS as an LTERM name associated with an input message. DFSAPPC is an IMS-reserved name for the message switch facility.

The LU 6.2 descriptors are built during IMS initialization. You can specify any number of descriptors. If an error occurs, the MVS system console and the IMS JOBLIB record the error messages. IMS initialization continues regardless of any errors during descriptor initialization.

**Related Reading:** For information on building descriptors, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

---

## Using MSC in an APPC/IMS Environment

APPC/IMS uses the services of APPC/MVS and MSC to provide the communication interface for an MSC configuration. Together, MSC and APPC/IMS allow:

- LU 6.2 programs to use the TP name of an IMS remote standard application program or an IMS remote modified application program. (The transaction is sent to the remote IMS system and executes. The transaction's reply is sent across the MSC links to the local IMS system and then on to the LU 6.2 application program.)
- A message switch to a remote logical terminal (LTERM) through the DFSAPPC System Service.
- Use of DFSAPPC for sending IMS remote transactions and data.
- Immediate or deferred program-to-program switching to an MSC-routed remote application program.

CPI Communications driven application programs cannot include transactions that execute on remote IMS systems.

All IMS transaction types except Fast Path are supported: conversational, nonconversational, response mode, and nonresponse mode.

MSC links that send or receive LU 6.2 transactions need larger link buffers to accommodate the processing of the LU 6.2 message prefix. IMS adds a prefix to the LU 6.2 message when it is sent over an MSC link. The minimum size of this prefix is 480 bytes.

**Related Reading:** For information on specifying the BUFSIZE parameter on the MSPLINK macro, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

To change the input message destination to any IMS local or remote destination after a message is received but before it is processed, use the Input Message Routing exit routine, DFSNPRT0.

**Definitions:** Using MSC with APPC/IMS requires you to understand the terminology used for the different MSC systems:

- The *originating system (local)* is the system from which the LU 6.2 program enters the IMS transaction.
- The *remote system* is the system in which the remote transaction executes.

## Using MSC in an APPC/IMS Environment

- The *intermediate system* is the IMS system that routes messages between the local and remote systems.

At any time, any of these three systems can receive LU 6.2 transactions.

---

## Recovering APPC Transactions in an MSC Environment

You can determine the recoverability of APPC messages in an MSC environment. Resource failures affect recovery. The recoverability of an IMS LU 6.2 transaction depends on whether the message is recoverable, irrecoverable, discardable, or nondiscardable, and when an error occurs.

To recover APPC transactions in an MSC environment, analyze the types of failures that can occur. How you handle the error depends on the following:

- The resource that fails: Was it an LU 6.2 session failure, an IMS failure, an application program failure, or an MSC link failure?
- Transaction mode: Was it recoverable or irrecoverable?
- Transaction type: Was it local or remote?
- LU 6.2 conversation mode: Was it asynchronous or synchronous?

You are in control of recovery by the way you define the transaction. To better understand the impact of your choices, read the information below on recoverability and irrecoverability, discardability and nondiscardability, and the possible points of failure. This information highlights pertinent facts, and then points you to other areas in the IMS library where the subjects are explained in greater depth.

## Recoverable versus Irrecoverable Transactions

By coding the inquiry parameter on the TRANSACT macro, you tell IMS the recovery status of a transaction. Non-inquiry mode transactions are recoverable; inquiry-mode transactions are not recoverable unless the RECOVER parameter is specified on the TRANSACT macro. Recoverable transactions are recovered across any IMS failure, shutdown, or restart unless a COLDSTART, COLDSYS, or COLDCOMM restart is performed.

You must define remote transactions with identical recoverability attributes on the local system where the LU 6.2 session originates and on the remote system where the transaction is processed by the application program. You do not need to define the transaction on any intermediate IMS system.

Message switches (messages from one LTERM to another) are always recoverable.

**Related Reading:** For more information on transactions and recoverability, see "Recovery Considerations" on page 210.

## Local APPC Transaction Discardability versus Nondiscardability

The LU 6.2 protocol that you choose for sending a transaction to IMS and the transaction mode (recoverable or irrecoverable) you choose determine if a local APPC transaction is discardable or nondiscardable.

IMS discards a local APPC transaction when it is any of the following:

- A CPI Communications driven application program (without SYNCLVL=SYNCPT specified)

## Recovering APPC Transactions in an MSC Environment

- It is defined as inquiry-only and nonrecoverable
- It is the result of synchronous input from the LU 6.2 application program
- It uses the APPC Sync\_Level option NONE

Otherwise, the transaction is nondiscardable. IMS recovers nondiscardable transactions whenever possible; it never recovers discardable transactions.

## OTMA Transaction Discardability versus Nondiscardability

IMS discards a local OTMA transaction if it was submitted with a Send-then-Commit synchronization flag (Commit mode 1).

IMS recovers an OTMA transaction if it was submitted with a Commit-then-Send synchronization flag (Commit mode 0).

**Related Reading:** For more information on OTMA, see *IMS/ESA Open Transaction Manager Access Guide*.

## Transaction Processing Point of Failure

The point of failure in the processing of a transaction also affects its recoverability. For example, when a local or remote transaction processes and reaches a commit point (sync point), IMS recovers the output response (from the log) even if you have defined the transaction as irrecoverable. Local APPC discardable transactions reach a commit point after IMS sends the output response message to the inputting APPC application program. In this situation, IMS has no output response message to recover or discard if a failure occurs after the commit point. If IMS has queued the transaction on an MSC link, IMS recovers the transaction across link failures.

A message can be either recoverable or irrecoverable, and either discardable or nondiscardable, according to the type of failure that might occur. The descriptions below show you what happens to your transaction when the LU 6.2 session, MSC link, local IMS, intermediate IMS, remote IMS, or application program fail. This information assumes that you can recognize where a failure has occurred and what you need to do to recover.

### Recovering Transactions after an LU 6.2 Session Failure

If an LU 6.2 session fails while IMS is receiving an input message, IMS discards the message. If IMS receives the complete message, processing depends on whether the conversation is:

- CPI-C or not CPI-C
- Synchronous or asynchronous
- Local or remote

**CPI-C Transaction:** If an LU 6.2 session fails while processing a CPI-C transaction, the application program can choose to end the conversation or to continue processing. IMS TM is not involved, and places no restrictions on the choice of committing or backing out updates. The application program makes the decision. Because IMS TM does not know about the session failure, it takes no action. The normal processing rules for commit and backout apply. IMS does not recover the LU 6.2 conversation.

### Related Reading:

- For information on designing your CPI-C LU 6.2 application program, see *IMS/ESA Application Programming: Design Guide*.

## Recovering APPC Transactions in an MSC Environment

- For information on CPI-C recovery and backout, see “Chapter 17. CPI Communications” on page 359.

**Not a CPI-C Transaction:** If an LU 6.2 session fails while a local IMS is sending transaction output that is not CPI-C to the LU 6.2 program, and the conversation is synchronous, IMS calls the Message Control/Error exit routine to determine whether to abort and back out, or to continue processing. The default action is to stop the transaction and discard the output message (this is the mode of operation for all protected conversations; that is, conversations allocated using SYNCLVL=SYNCPT). If the conversation is asynchronous, IMS does not call the Message Control/Error exit routine, but queues the output on the message queue to the TP name of DFSASYNC.

**Related Reading:** For information on coding the Message Control/Error exit routine, see *IMS/ESA Customization Guide*.

**Remote APPC Transaction:** If an LU 6.2 session fails while processing a remote APPC transaction, IMS recovers the output message if it has been enqueued on the local system’s MSC link. If the transaction has not at least reached the point of being enqueued on the MSC link, IMS discards it. IMS discards the transaction regardless of the recoverability mode and regardless of whether the LU 6.2 conversation is synchronous or asynchronous. IMS does not call the Message Control/Error exit routine:

- If the transaction is asynchronous, when the output from a remote transaction for a failed LU 6.2 session returns from the remote system to the originating system, IMS sends the response asynchronously to the LU 6.2 application program by using the DFSASYNC TP name.
- If the transaction is synchronous, when the output from a remote transaction for a failed LU 6.2 session returns from the remote system to the originating system, IMS calls the Message Control/Error exit routine to either discard or re-route the transaction output. The default action is to discard the output.

**Related Reading:** For information on using DFSASYNC in your application program, see *IMS/ESA Application Programming: Design Guide*.

### Recovering Transactions after an MSC Link Failure

When an MSC link failure occurs, IMS always recovers all messages, including IMS transactions and responses that are enqueued, about to be enqueued, or being sent across an MSC link. This recoverability is guaranteed, regardless of whether the message is en route to a local, intermediate, or remote MSC system. The recoverability is not affected by the transaction mode (recoverable or irrecoverable) or the discardable or nondiscardable characteristics of the LU 6.2 protocol used to send the transaction to the local IMS.

Link failures can delay messages from other IMS systems and cause the synchronous LU 6.2 conversation to wait longer than expected for the response.

**Related Reading:** For information on restarting a logical link, see “Restarting a Logical Link” on page 208.

### Recovering Transactions after a Local IMS Failure

IMS discards local APPC transactions across a local IMS failure if they meet the discardability test above. Otherwise, IMS does not discard local APPC transactions, because they are nondiscardable.



## Recovering APPC Transactions in an MSC Environment

**Related Reading:** For more information on transaction discardability versus nondiscardability, see “Local APPC Transaction Discardability versus Nondiscardability” on page 382.

If you define your remote APPC transactions as inquiry-type transactions and do not specify RECOVER on the TRANSACT macro definition in the local IMS, IMS does not recover them after a local IMS failure. Otherwise, IMS recovers all recoverable, nonconversational transactions.

After the local IMS system sends the transaction message to an intermediate or remote IMS system, a local IMS system failure has no effect on your transaction's recoverability. The transaction continues to its destination and is processed. When the remote IMS sends the transaction response to the originating IMS after the failure, IMS sends the response to its destination asynchronously through the DFSASYNC TP name. The LU 6.2 application programmer needs to plan for this situation.

**Related Reading:** For information on using DFSASYNC in your application program, see *IMS/ESA Application Programming: Design Guide*.

### Recovering Transactions after a Remote IMS Failure

When a remote IMS failure occurs, based on the recoverability attributes of an APPC transaction in a remote IMS system, IMS recovers the transaction if it is queued for processing or is being processed in the remote IMS system. Recoverable transactions are recovered; irrecoverable transactions are not. After the transaction reaches a commit point, IMS recovers the output response message regardless of the recovery attributes of the transaction. The discardable and nondiscardable characteristics of the APPC conversation in the originating IMS system have no bearing on the transaction's recoverability in the remote IMS system across a remote IMS failure.

IMS recovers transactions that are en route to the remote IMS system (meaning the transaction message is still en route in the local or intermediate IMS system) when the remote IMS system fails, regardless of the transaction's recoverability characteristics. After the failure, the remote IMS receives and processes the transactions that were en route at the time of the failure.

### Recovering Transactions after an Intermediate IMS Failure

IMS always recovers all messages en route to or from an intermediate IMS that are queued in the intermediate IMS system regardless of the recoverability characteristics of the transaction or message. If the intermediate IMS restarts with either a COLDSTART, COLDCOMM, or COLDSYS, the messages are lost.

### Recovering Transactions after an Application Program Failure

Transactions sent to IMS from LU 6.2 application programs are processed in the same way as non-LU 6.2 initiated transactions during application program failures. If an application program fails before reaching a commit point while processing a local or remote transaction from an LU 6.2 device, IMS backs out all messages except those that were inserted to an alternate express PCB and committed with a PURG call. If the failure occurs after the transaction reaches a commit point, IMS recovers everything. If the failing application program's input message was received from another application program (program-to-program switch), this prior application program's processing is still committed (as is true for non-LU 6.2 application programs).

## Recovering APPC Transactions in an MSC Environment

### Recoverability Flows of LU 6.2 Transactions

This section contains four diagrams that show synchronous and asynchronous transaction flows, and shows when the transactions are recoverable, irrecoverable, discardable, or nondiscardable.

Figure 56 shows the flow of a transaction sent from an LU 6.2 synchronous conversation to a local IMS system.

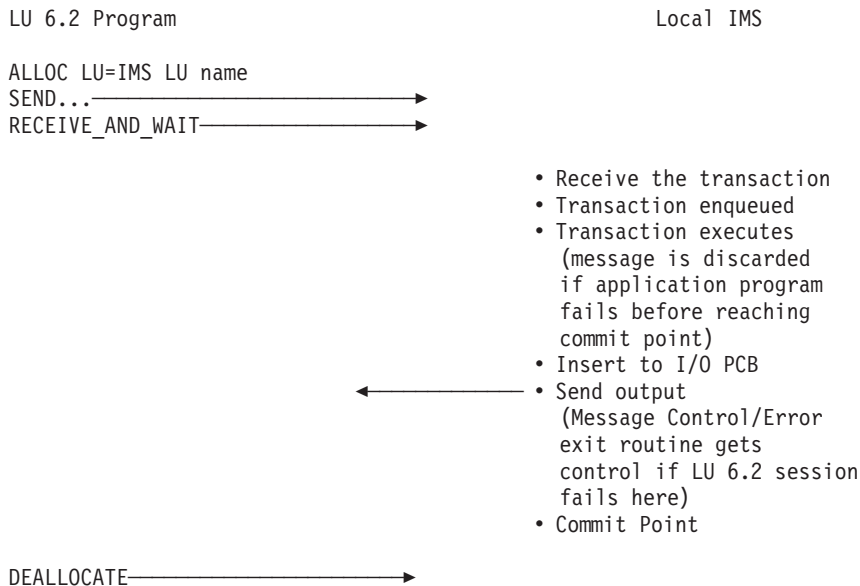


Figure 56. Example of LU 6.2 Synchronous Transaction to a Local IMS

Figure 57 shows the flow of a transaction sent from an LU 6.2 asynchronous conversation to a local IMS system.

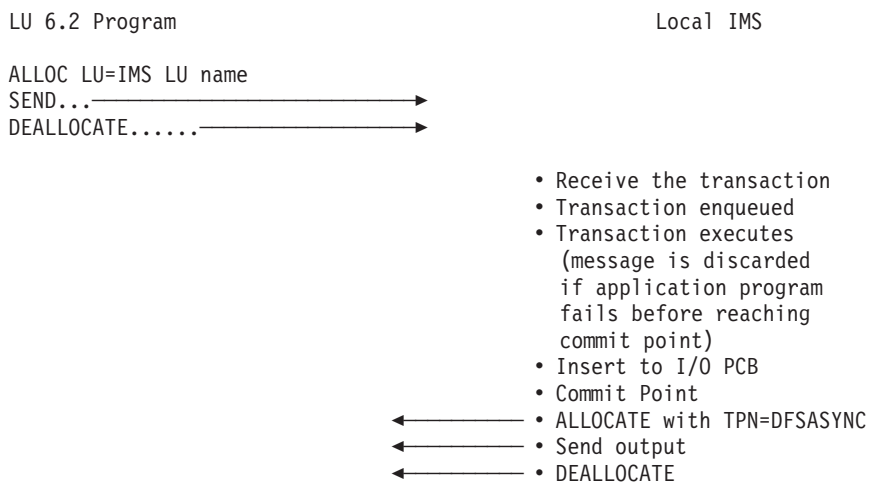


Figure 57. Example of LU 6.2 Asynchronous Transaction to a Local IMS

Figure 58 on page 387 shows the flow of a transaction sent from an LU 6.2 synchronous conversation to a remote IMS.

## Recovering APPC Transactions in an MSC Environment

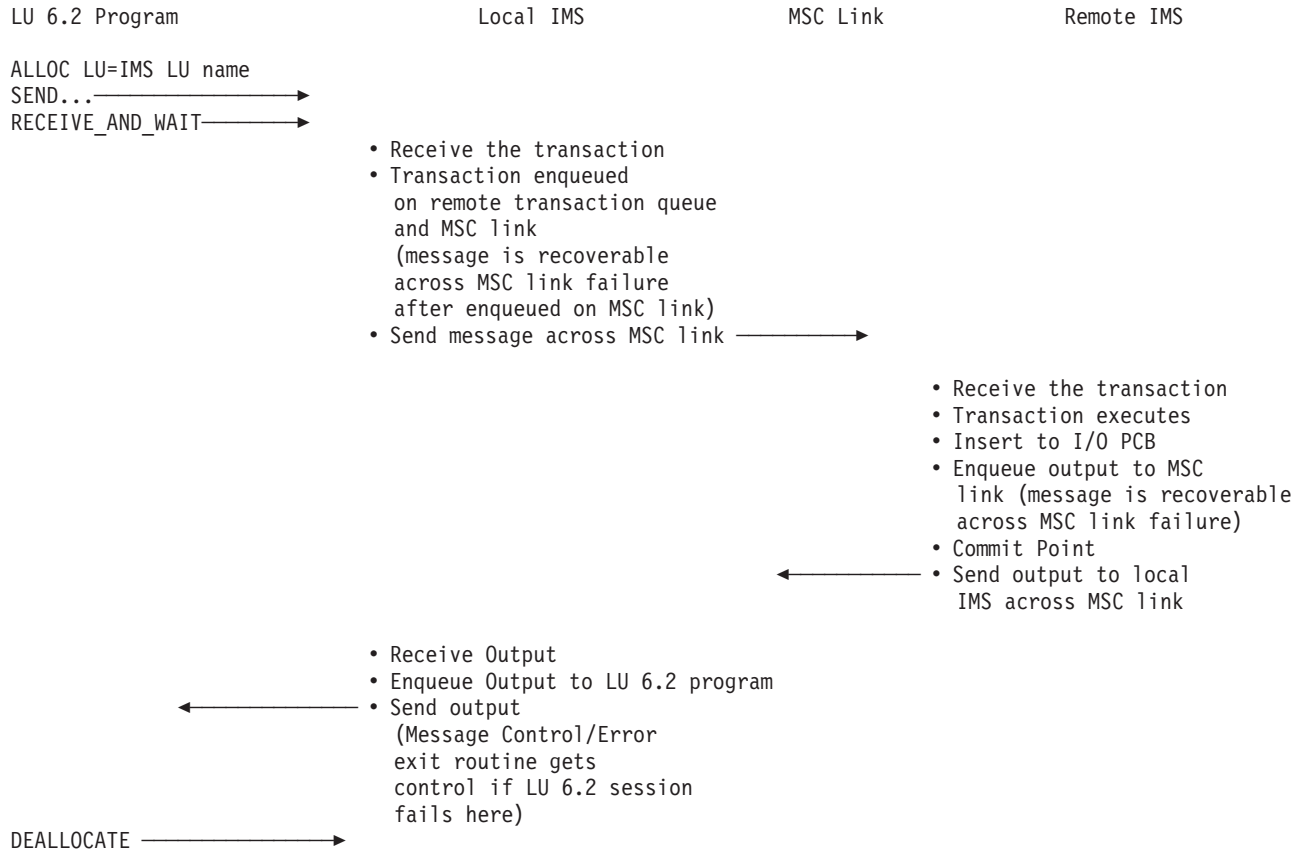


Figure 58. Example of LU 6.2 Synchronous Transaction to a Remote IMS

Figure 59 on page 388 shows the flow of a transaction sent from an LU 6.2 asynchronous transaction to a remote IMS.

## Planning for XRF

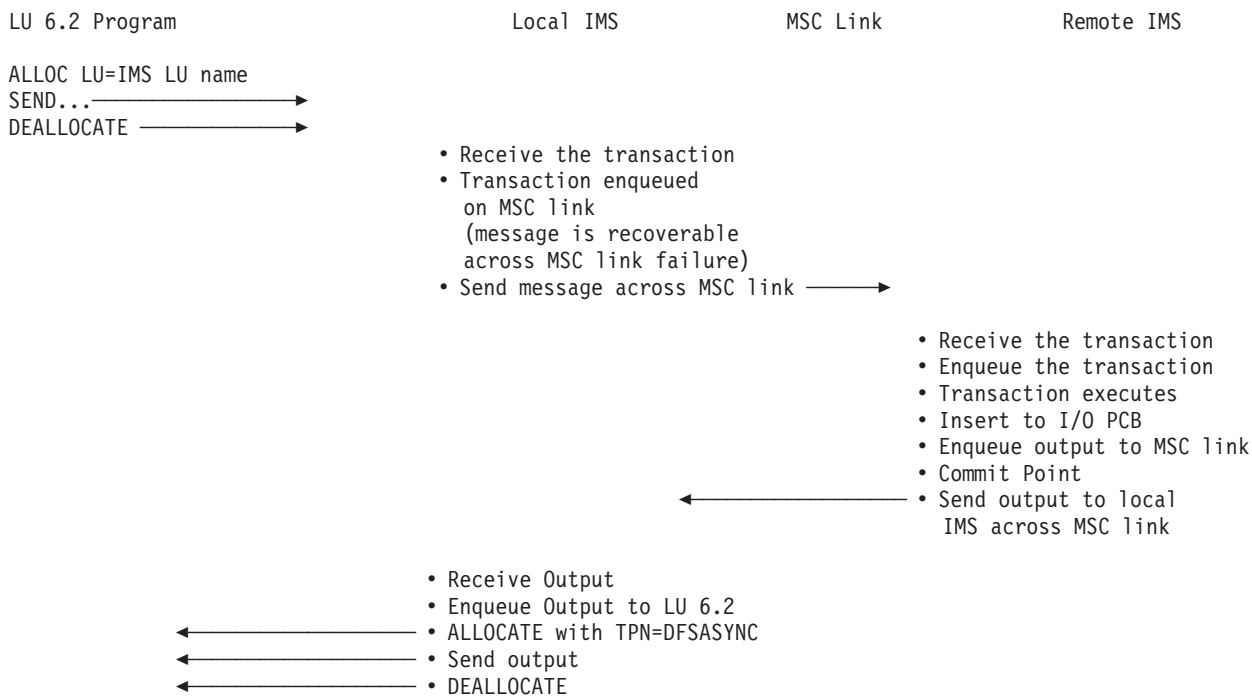


Figure 59. Example of LU 6.2 Asynchronous Transaction to a Remote IMS

## Planning for XRF

All LU 6.2 conversation messages that are sent to a failing IMS system are broken at the time of an XRF takeover. New conversations must be allocated on the new active IMS system after takeover processing completes. XRF support for LU 6.2 application programs is Class 3 service.

The application program uses the USERVAR parameter as the LU name parameter on the ALLOCATE verb. This allows the same LU name to allocate the conversation on the new active IMS. This is the same concept as the LU usage for non-LU 6.2 IMS terminals, but a different USERVAR name is used. At takeover time, the alternate IMS system issues a VTAM MODIFY USERVAR command to redirect the USERVAR from the failed active IMS system to the alternate IMS system. Also, a VTAM VARY TERM command is issued against the failed active IMS to terminate any conversations held during takeover. Then VTAM propagates the USERVAR changes to other VTAMs as needed without the use of NCCF CLISTS.

Any APPC transactions that are enqueued to a remote IMS are handled by the MSC protocols. After takeover in the local IMS, the queued irrecoverable messages are lost. If the same message is queued to a remote IMS, it is recovered when the local IMS takes over. Processing continues and the reply is sent to the active IMS.

VTAM, MVS, NetView, and any other product requirements should be carefully reviewed. APPC/IMS requires correct operation of these other products.

---

## Transaction Retry Characteristics

IMS retries certain abend conditions. Some examples of these conditions that are retried are:

- Deadlock
- Lock reject
- Fast Path retry conditions

These retry conditions still apply to standard DL/I application programs even if they receive their messages from an LU 6.2 application program. If an abend that can be retried occurs, IMS issues an APPC ATBEXAI call to APPC/MVS to determine if any established conversations exist. If a conversation has been allocated, the abend is not retried and the application program is terminated.

If any CPI Communications resource is being used by the application program, the abend condition cannot be retried. Thus, CPI Communications driven application programs and modified IMS application programs that have allocated an LU 6.2 conversation before the abend occurred can never be retried. This prohibition on retry is necessary, because IMS cannot control the state of CPI Communications resources. IMS supports the DL/I INIT STATUS GROUPB call for CPI Communications driven application programs, but not for modified IMS application programs that have allocated an LU 6.2 conversation before the deadlock is detected.

---

## Qualifying Network LU Names

You can use the same name for LUs on different systems by adding the network identifier. This eliminates the need to have unique names for every LU on every system in your complex. You can use the network-qualified LU name as the name of the partner LU to allocate remote LU 6.2 conversations and sessions.

A network-qualified LU name consists of a one- to eight-character network identifier of the originating system, followed by a period and the LU name. A network-qualified LU name must be enclosed in single quotes for commands.

**Example:** /DISPLAY LUNAME'NETID1.LUAPPC2'

Use a network-qualified LU name when transmitting data to a remote destination. If no network identifier is present, IMS allows MVS to determine the destination.

The parameter ALL for either the network identifier or the LU name cannot be substituted in a network-qualified LU name in commands.

**Example:** /DISPLAY LUNAME'NETID1.ALL'.

The LU name in the LU 6.2 descriptor can be network qualified. The network-qualified LU name is optional on commands that support the LUNAME keyword.

APPC/MVS uses the name of LU 6.2 network-qualified LUs to allocate conversations. APPC/MVS strips off the network ID and passes the 8-byte LU name to VTAM. Without APPC/MVS support for network-qualified names, the LU name must be unique for the different networks.

## Qualifying Network LU Names

The ALL parameter for either the network identifier or the LU name cannot be substituted in a network-qualified LU name in a command.

The LU name in the LU 6.2 descriptor can be network qualified.

**Related Reading:** For information on using network-qualified names in commands, see *IMS/ESA Operator's Reference*.

---

## DFSAPPC System Service

DFSAPPC is an IMS system service for exchanging messages between LU 6.2 application programs (LU 6.2 to LU 6.2), and between LU 6.2 application programs and IMS-managed LTERMs. Message delivery is asynchronous; messages are held on the IMS message queue until they are delivered.

LU 6.2 application program can use DFSAPPC to send messages to IMS-managed LTERMs. Use the LTERM option of the DFSAPPC service to select this capability.

## Message Switching

Message switching is part of the implicit APPC interface and allows IMS terminals and LU 6.2 application programs to exchange messages. Messages routed to an LU 6.2 application program initiate LU 6.2 application programs.

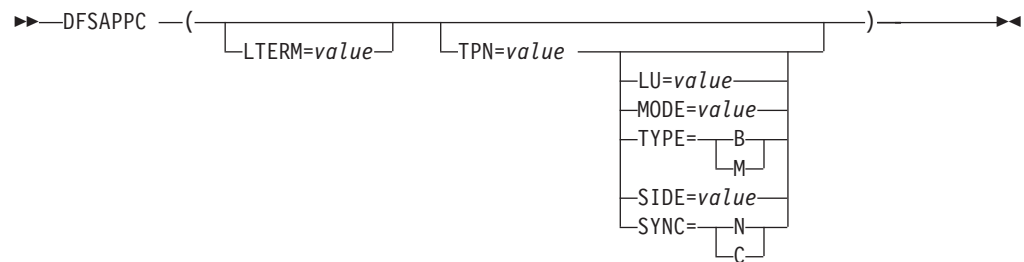
When using DFSAPPC, the remote device can choose to route a message using either the LTERM or LU 6.2 TPN option. Messages sent with the LTERM option are directed to IMS-managed local or remote LTERMs.<sup>36</sup> Messages sent without the LTERM option are sent to the specified LU 6.2 application program.

The message format for DFSAPPC is as follows:

DFSAPPC (*options*)*user\_data*.

---

36. If the LTERM is associated with an LU 6.2 destination, the message is sent as if an LU 6.2 application program had been explicitly selected.



**Notes:**

- 1 A mandatory blank is required between DFSAPPC and the options.
- 2 Use blanks anywhere within the DFSAPPC options except within keywords or values. Use commas as delimiters between keyword-parameter sets along with or in place of blanks. However, because the TP name character set allows commas, at least one blank must be used to terminate the TPN value.
- 3 Use the IMS default values for the DFSAPPC options to establish an LU 6.2 conversation with a partner program when the values are not provided by another source. If the DFSAPPC service is coded without specifying any options, use IMS default LU 6.2 conversation characteristics.
- 4 You can specify either the LTERM= or the TPN= option, but not both. Only use the other keyword options when you specify the TPN= option.

The DFSAPPC option keywords are defined as follows:

**LTERM=**

The 1- to 8-character LTERM= option is the name of an IMS LTERM. Messages sent with the LTERM option are directed to an IMS-managed local or remote LTERM. If the LTERM is associated with the LU 6.2 descriptor, it is treated as if an LU 6.2 application program has been explicitly selected. LTERM names can contain uppercase alphabetic, numerics, and national characters ('@', '\$', '#'). When LTERM= is specified, other keywords cannot be specified.

**TPN=**

The 1- to 64-character TPN= option is the partner's transaction program name used with the logical unit name to establish an LU 6.2 conversation with a partner program. Because the TP name character set allows commas, at least one blank must be used to terminate the TPN value.

TP names can contain any character from the 00640 character set except a blank. The 00640 character set, documented in the *Common Programming Interface Communications Specification* includes uppercase and lowercase letters A through Z, numerals 0-9, and 20 special characters.

When the TPN= and SIDE= options are specified, the TPN= name overrides the TP name contained in the side information entry.

Although DFSAPPC allows the use of the 00640 character set, IMS commands do not use this character set. IMS commands can only operate on TPNs that use uppercase alphabetic, numeric, and national characters ('@', '\$', '#'). IMS commands cannot operate on extended TPNs.

**LU=**

The 1- to 17-character LU= option is the partner's logical unit name used with the transaction program name (TPN) to establish an LU 6.2 conversation with a partner program.

## DFSAPPC System Service

LU names can contain any character from the APPC/MVS Type A character set. LU names can contain uppercase alphabets, numerics, and national characters ('@', '\$', '#'), and must begin with an alphabetic or national character. You can also use a 17-byte network-qualified LU name in the LU= field.

When the LU= and SIDE= options are specified, the LU= name overrides the LU name contained in the side information entry.

**Related Reading:** For more information on Type A character sets, see *MVS/ESA Programming: Writing Transaction Programs for APPC/MVS*.

### MODE=

The 1- to 8-character MODE= option is the partner's mode name used with the logical unit name and transaction program name to establish an LU 6.2 conversation with a partner program.

MODE names can contain any character from the APPC/MVS Type A character set. MODE names can contain uppercase alphabets, numerics, and national characters ('@', '\$', '#'), and must begin with an alphabetic or national character.

When the MODE= and SIDE= options are specified, the MODE= name overrides the mode name contained in the side information entry.

**Related Reading:** For more information on Type A character sets, see *MVS/ESA Programming: Writing Transaction Programs for APPC/MVS*.

### SIDE=

The 1- to 8-character SIDE= option is the side information entry name used to establish an LU 6.2 conversation with a partner program.

SIDE names can contain any character from the 01134 character set. The 01134 character set, documented in the *Common Programming Interface Communications Specification* includes uppercase alphabets A through Z and numerics 0-9.

When the SIDE= option is specified, the LU=, TPN=, and MODE= options can also be specified to override the values in the side information entry.

### SYNC=

The SYNC= option allows the application to override the LU 6.2 conversation sync level default provided by IMS.

**SYNC=N** Sync\_Level is NONE.

**SYNC=C** Sync\_Level is CONFIRM.

### TYPE=

The TYPE= option allows the application to override the LU 6.2 conversation type default provided by IMS.

**TYPE=B** Conversation type is BASIC.

**TYPE=M** Conversation type is MAPPED.

If an error is found while processing the options list, error message DFS1957 DFSAPPC ERROR is sent to the terminal.

**Related Reading:** For more information on error message DFS1957, see *IMS/ESA Messages and Codes*.



## Asynchronous Output Delivery

When creating a message destined for an LU 6.2 application program, IMS establishes conversation characteristics. These characteristics are extracted from the:

- LU 6.2 descriptor
- DL/I Change call option list
- DFSAPPC message switch options

If IMS cannot extract a particular conversation characteristic from the above list, IMS uses the defaults shown in Table 39. If a side table name is extracted, the default mode name is not used. IMS assumes that side table entries contain a mode table entry name. If an /ALLOCATE command for a particular LUNAME - TPNAME destination specifies a mode table entry name, that entry name overrides the mode table name specified for the message.

Table 39. APPC/IMS Default Conversation Characteristics

Characteristics	Default Value
Conversation_Type	Mapped
Deallocate_Type	Deallocate_Sync_Level
Error_Direction	Receive_Error
Fill	Fill_LL
Log_Data	Null
Log_Data_Length	0
Mode_Name	'DFSMODE' <sup>1</sup>
Mode_Name_Length	7
Partner_LU_Name	'DFSLU'
Partner_LU_Name_Length	5
Prepare_to_receive_type	Prep_To_Receive_Sync_Level
Receive_type	Receive_and_Wait
Return_Control	When_Session_Allocated
Send_Type	Buffer_Data
Sync_Level	Confirm
TP_Name	'DFSASYNC'
TP_Name_Length	8

**Note:**

<sup>1</sup>IMS uses the same mode name provided by the inputting LU 6.2 partner to allocate the outbound conversation. That is, whatever mode name the inputting conversation uses, IMS also uses it for outbound allocates.

IMS uses this information to initiate a conversation with the LU 6.2 application program associated with the alternate PCB. Certain fields, such as LU name, are application specific. Default values are provided but can be overridden by parameters associated with the message. A default value is used by IMS only if no value is provided by any other source. The application program can modify the default conversation characteristics using an expanded interface to the DL/I CHNG call.

### APPC Transaction Security

The security options for APPC/IMS and LU 6.2 application programs are quite extensive. The partner systems can range from a single-user terminal, such as a PS/2, to a multi-user system, such as VM/ESA. All systems can have their own complex security environment. Security for IMS can be simple or complex.

Every transaction program name (TPN) must pass a security check before it is executed. The user ID that initiates the transaction is identified on the LU 6.2 format header (FMH5). If no user ID exists because you specify SECURITY=NONE, you can only access resources that are not defined with UACC (NONE). Any TPNs that are accessible in all circumstances should not be defined with UACC (NONE). The TPN security definition is required.

MVS security consists of two parts. First, MVS authenticates the transaction user. The LU 6.2 transaction contains security information. The FMH5 contains the user ID, a "profile" name, which is used as the group name, and security options. You supply both the user ID and password. The user ID is defined to RACF, and the password must be valid for the user ID.

If Already\_Verified is specified in the FMH5, the sending system verifies the user ID. This user ID must be defined to RACF on the receiving MVS system. No password is needed in this case.

If SECURITY=NONE is specified, MVS does no checking. Instead, MVS builds a special security profile that corresponds to SECURITY=NONE. This allows access to MVS and APPC/IMS resources that have UACC specified at any level other than NONE. Resources with UACC (NONE) or without a UACC specified cannot be accessed.

After the user ID is established, MVS verifies that the user ID can execute the specific transaction. MVS verifies that the user ID's access profile has ACCESS (EXECUTE) for the entity *dbtoken.x.tpname* in the CLASS (APPCTP). The value of *dbtoken* is the dbtoken value specified in the TP\_Profile data set. Based on the APPCTPxx parameter specified for this LU, the value of *x* is either the *user ID*, *group* or *SYS1*.

If either of these security checks fails, MVS rejects the transaction, and IMS is not informed of it. MVS can also check:

- Session-level security (RACF resource class APPCLU)
- Port of entry (RACF resource class APPCPORT)
- Local application (RACF resource class APPL)

The IMS administrator should verify that these security checks are successful.

**Related Reading:** For more information on coding the RACF resource classes, see *OS/390 V1R1 MVS Planning: APPC Management*.

The security check in IMS is based on the IMS transaction code or executed command name. If the TPN is DFSAPPC, no additional security check occurs. If RACF is used on your system, MVS rejects the transaction if RACF is not active. The IMS command name or transaction code associated with the TPN is used in the RACF resource class associated with this IMS system ('C' or 'T'). RACF checks IMS commands and transactions for all other IMS terminal types in the same way.

## APPC Transaction Security

If the RACF check is successful, the Transaction Authorization exit routine (DFSCTRN0) is called for transactions, and DFSCCMD0 is called for command authorization. However, the following rules apply to RACF:

- For commands, default security only applies if RACF is not used.
- For remote transactions, RACF is optional.

Otherwise, the exit routines make the security decision.

The intended environment executes APPC/IMS with RACF security active. It is possible to run with RACF not active in the APPC/IMS system, but it is not possible to run with RACF not active in the MVS system. In this sense, RACF is mandatory for LU 6.2.

The complexity of the security environment is derived partly from the many resources involved (VTAM, MVS, and IMS) and the granularity of protection that is possible. The security definitions must be closely coordinated for successful operation of the application system.

**Related Reading:** For more information on using IMS Security, see *IMS/ESA Administration Guide: System*.

## APPC Transaction Security

---

## Part 6. Administering SLU P and Finance Communication

<b>Chapter 19. Overview of SLU P and Finance Communication</b>	399
The IMS-SLU P Network	400
System Configuration	400
SLU P and Finance Workstations	401
System Controller Application Program	402
Writing the Controller Application Program with MFS and XRF	402
Writing the Controller Application Program for MFS	402
Writing the Controller Application Program for XRF Systems	403
Converting Controller Application Programs from Finance to SLU P	404
VTAM Facilities Used	404
VTAM Commands and Indicators	405
Request-Recovery Command	406
Change-Direction Indicator	406
Establishing Connection and Specifying Logon Modes	407
Establishing Connection with the XRF Complex	408
Bracket and Send/Receive Management	408
<b>Chapter 20. IMS Facilities Used for SLU P and Finance</b>	411
Component Definition	411
LTERM Naming	411
Output Component Selection	412
Input Component Determination	412
Terminal-Response Mode	413
Defining a Workstation for Terminal-Response Mode	415
Output Messages Sent While in a Between-Brackets State	416
Designing for Output Messages Sent While in Between-Brackets State	418
IMS Message Format Service	418
Designing MFS for the Workstation Environment	418
MID/MOD Chaining	419
MFS Output Formatting for the SLU P System	419
MFS Message Recovery	419
MFS Control Functions (Finance)	420
MFS Control Functions (SLU P)	420
MFS Paging and BID Options	421
Display Screen Protection for Finance Stations	421
Extended Output Component Protection (SLU P)	422
Input and Output Editing Options (SLU P)	423
Use of Responses or Brackets to Acknowledge Recoverable Input	424
Message Recovery	425
Message Resynchronization	426
Finance and SLU P in an XRF Complex	427
Fast Path Messages with Finance and SLU P	427
Fast Path Output Messages (Finance)	427
Fast Path Output Messages (SLU P)	428
Fast Path Message Resynchronization	429
<b>Chapter 21. Network Operation for SLU P and Finance</b>	431
Starting an IMS Network	431
Making IMS Ready	431
Session Initiation (Starting Workstations)	431
Session-Initiation Transmission Sequence	432
Controller Application Program Involvement in Message Resynchronization	433
Design Considerations	433

## Administering SLU P and Finance Communication

Sequence Number Management . . . . .	434
Set-and-Test-Sequence-Numbers (STSN) . . . . .	435
Suspending Output from IMS . . . . .	438
Session Termination . . . . .	439
Orderly Termination . . . . .	440
Immediate Termination . . . . .	440
Shutting Down an IMS Network . . . . .	441
SLU P Messages . . . . .	441
Send/Receive and Bracket Protocol . . . . .	441
<b>Chapter 22. SLU P Message Protocols . . . . .</b>	<b>443</b>
General Format of Input Function Management Headers (Finance) . . . . .	443
Input Message Descriptor Byte (Finance) . . . . .	444
General Format of Input Function Management Headers (SLU P) . . . . .	444
Input Message Descriptor Bytes (SLU P) . . . . .	445
Input Component Identification (SLU P) . . . . .	446
Input Bracketing Protocol . . . . .	446
Activating MFS Input Formatting for Finance Workstations . . . . .	446
Output Messages . . . . .	447
MFS Distributed Presentation Management Output (SLU P) . . . . .	449
General Format of Output Function Management Headers (Finance) . . . . .	450
Output Message Descriptor Byte (Finance) . . . . .	450
Output Component ID Byte (Finance) . . . . .	451
MFS Data Bytes (Finance) . . . . .	451
General Format of Output Function Management Headers (SLU P) . . . . .	451
Output Message Descriptor Bytes (SLU P) . . . . .	452
MFS Data Bytes (SLU P) . . . . .	452
Output Bracketing Protocol . . . . .	453
Activating MFS Output Formatting . . . . .	454
Response Requests (Finance) . . . . .	454
Response Requests (SLU P) . . . . .	455
Input Response Requirements . . . . .	455
Output Response Requirements . . . . .	456
IMS Transaction Types . . . . .	456
Recoverable-Inquiry Transactions . . . . .	457
Irrecoverable-Inquiry Transactions . . . . .	457
Verifying IMS Receipt of Irrecoverable Messages . . . . .	458
IMS Message Switches . . . . .	458
IMS Commands . . . . .	458
VTAM Commands and Indicators . . . . .	458
MFS Control Requests . . . . .	459
Error Handling . . . . .	459
IMS-Detected Errors . . . . .	459
System Sense Field (SSENSE) . . . . .	459
User Sense Field (USENSE) . . . . .	459
Controller or Station-Detected Errors . . . . .	460
System Sense Field . . . . .	460
User Sense Field . . . . .	461
VTAM Logical Unit Status (LUSTATUS) Command . . . . .	461
VTAM Ready-to-Receive (RTR) Command . . . . .	462
VTAM CANCEL Command . . . . .	462
VTAM Request-Recovery Command . . . . .	463

---

## Chapter 19. Overview of SLU P and Finance Communication

This part (“Part 6. Administering SLU P and Finance Communication”) contains all the information that you require for designing and implementing a SLU P and Finance Communication System under an IMS system. This chapter presents an administrative overview of SLU P and Finance Communication Systems, and explains how IMS implements this architecture.

### **In this Chapter:**

- “The IMS-SLU P Network” on page 400
- “System Configuration” on page 400
- “SLU P and Finance Workstations” on page 401
- “System Controller Application Program” on page 402
- “Writing the Controller Application Program with MFS and XRF” on page 402
- “Converting Controller Application Programs from Finance to SLU P” on page 404
- “VTAM Facilities Used” on page 404
- “VTAM Commands and Indicators” on page 405
- “Establishing Connection and Specifying Logon Modes” on page 407
- “Establishing Connection with the XRF Complex” on page 408
- “Bracket and Send/Receive Management” on page 408

User-written application programs for IBM Finance Communication Systems can be defined to operate in two different ways with IMS—as an IBM 3600/4700 Finance Communication Controller or as a secondary logical unit type P (SLU P). These systems are described as UNITYPE=FINANCE and UNITYPE=SLUTYPEP, respectively, on the TYPE macro statement.

### **Definitions:**

- This chapter uses the term *SLU P* when describing support that is applicable to both system types.
- This chapter uses the terms *Finance* and *SLUTYPEP* when it is necessary to distinguish between the systems.

A major difference between SLU P and Finance systems is the level of MFS support available to workstations in the programmable control unit.

- Finance—Workstations can be identified as displays, printers, passbook printers, and ATMs (automated teller machines), such as the 4730 systems. MFS can provide detailed support for display paging, printer page formatting, and ATMs.
- SLU P—Each LU in the controller has a program name and its unique device characteristics are unknown to IMS. The programmable controller is responsible for device control and formatting. The MFS administrator and the program administrator for the remote controller need to establish data field structures for exchanged messages. MFS accepts the input structure and rearranges it as required for the IMS application program. At output, MFS accepts application program-provided data and converts it to the correct format for transmission to the controller. This MFS facility is called distributed presentation management (DPM).

**Related Reading:** For more information on ETO and SLU P and Finance systems, see “ETO and 3600/Finance and SLU P” on page 168.

### The IMS-SLU P Network

The three major elements of an IMS-SLU P network are:

- The controller and the terminals executing the remote program
- The communication link
- The central processor executing the host IMS system

Each element includes programming to perform a part of the total data processing performed by the system. IMS resides in the host processor and communicates with an application program in the SLU P system's controller through the communication link. The controller's application program monitors the terminals attached to the controller.

IMS supports 4700 terminals attached to an IBM 4701/4702 Finance Communication Controller. IMS supports the IBM 4730 Personal Banking Machine, both directly attached to a 37x5 as a SLU P and attached to a 4701/4702 as part of the SLU P system.

**Related Reading:** For a list of 4700 terminals, see *IBM 4700 Finance Communication System: System Summary*.

IMS also supports 3600 terminals attached to either an IBM 3602 or 4701/4702 as part of a Finance system. (IMS does not distinguish between 3600- and 4700-series terminals.)

Examples of other IBM products that connect to IMS with the SLU P protocols are the Series/1, the IBM 3650, and the IBM 8100.

Each of these terminals, or workstations, can be defined to IMS as a component of the appropriate SLU P system using the COMPT or ICOMPT keyword on the TERMINAL macro statement.

---

## System Configuration

Figure 60 on page 401 shows a sample IMS-SLU P system configuration. Before configuring a SLU P system, you must identify the financial operations that are required of the planned system. After you have done this, you can define to IMS the configuration of the logical units and components. The SLU P system is regarded by IMS as a subsystem and consists of one or more logical units.

**Definition:** When configuring the system, you can define one or more *logical units*. These logical units<sup>37</sup> can consist of devices, storage, and application programs.

---

37. These logical units are referred to as *logical workstations* in Finance Communication System publications.



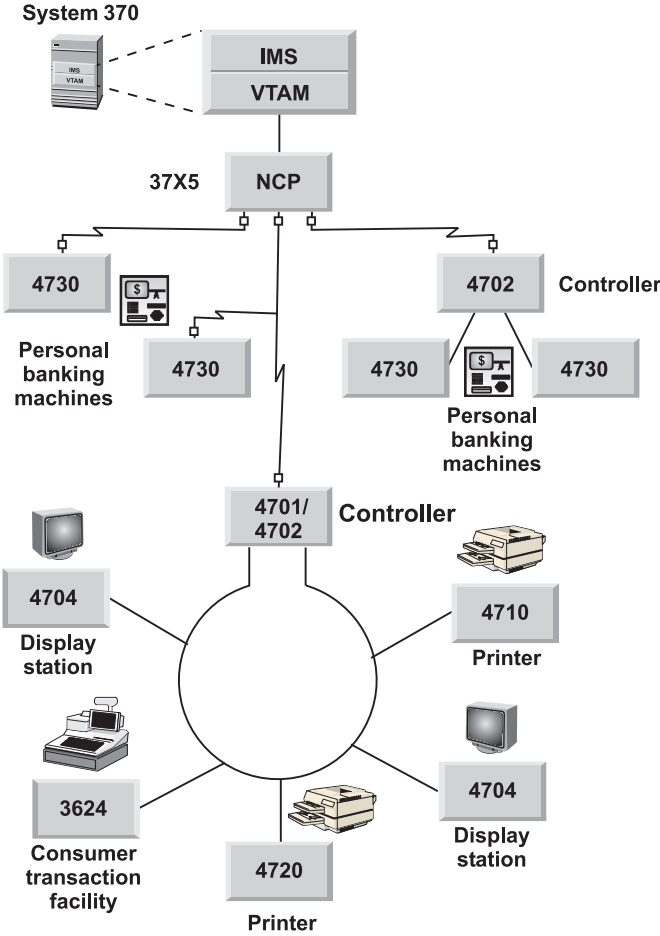


Figure 60. Sample 4700 Configuration with IMS

---

## SLU P and Finance Workstations

A workstation can consist of one or more terminals. Each workstation performs a specific type of financial operation, such as teller operation (deposits and withdrawals), report printing, or cash dispensing. A typical set of workstations is shown in Figure 61 on page 402. Workstations 1 and 2 in that figure are examples of workstations that might be used for teller operations. Workstation 3 is an output-only station that could be used to print reports such as trial balances and exception reports. Workstation 4 consists of a 4730 terminal used directly by the customer of the financial institution to accomplish teller-type functions. The workstation, as defined to IMS, does not need to reflect the actual devices attached to it at the 4701/4702, but the workstation must reflect IMS's view of the workstation.

## SLU P and Finance Workstations

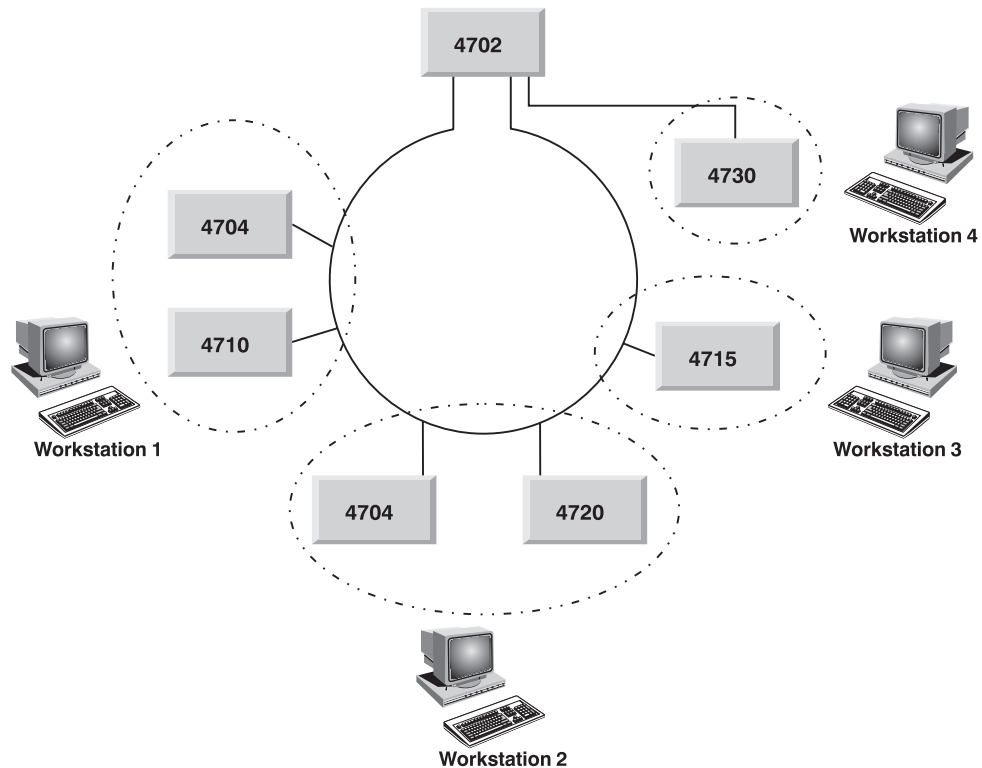


Figure 61. Sample 4700 Workstation Configuration

A sample configuration is available from the IMS library (IMS.GENLIBB; member name=DFSCP360).

---

## System Controller Application Program

The operation of a workstation is controlled by a user-written application program that resides in the SLU P system's controller. The application program can be designed to control one or more workstations and needs only to perform the functions required by its workstations. The functions available to the application program include:

- Reading and writing to the terminals associated with the workstation
- Editing and verifying the data received from the terminal
- Reading and writing to the controller diskette
- Reading and writing to the host processor
- Editing and verifying data received from the host processor
- Operating offline when the host processor or IMS is unavailable

---

## Writing the Controller Application Program with MFS and XRF

If your system uses MFS and XRF, this section provides information that pertains to writing the controller application program.

## Writing the Controller Application Program for MFS

When writing the controller application program for a system that uses MFS, do each of the following:

## Writing the Controller Application Program with MFS and XRF

- Do not specify the WL (warning line) option on the 4700 controller's TERMINAL macro instruction. Specifying the WL option can cause an unexpected Finance system data exception error to occur on an LWRITE instruction.
- Specify the PS (page size) option on the 4700 controller's TERMINAL macro only if EJECT is specified on the IMS MFS DEV statement. If PS is not specified, the EJECT from IMS results in a new line function. If the "end-of-page" condition occurs on a workstation, an unexpected Finance system data exception error can occur. An EJECT from IMS resets the line count, resulting in an "end-of-page" condition.

If the CFOLD option is specified on the 4700 controller's TERMINAL macro instruction, the formats defined for MFS should not cause spacing for the center fold on a passbook. The SLU P system performs this function automatically.

- Check to see if the IMS MFS DFLD statement specifies ATTR=YES. If it does, any attempt to print an underscore (X'6D') can result in either of the following:
  - A data check on a 3610/3612  
Specify a print position for the EBCDIC value (X'6D') when the OUTRTBL statement specifies other than the 128-character set for the 3610/3612 device.
  - Printing of a blank on a 3618 device  
Specify a print position for the EBCDIC value (X'6D') when the OUTRTBL statement specifies the 48-character set for a 3618 device.
- Check to see if the MFS DEV statement option of EJECT, BGNMXG, BGNPP, or ENDPP is used to send eject characters (X'0C') within one IMS output transmission (other than the last character). If so, an "intervention-required" condition occurs on the passbook printer after the eject. In this situation, your application program in the controller must be capable of issuing multiple LWRITE instructions to the passbook printer to print the IMS output transmission. Each LWRITE should include the data up to and including an eject character, or up to the end of the transmission.

**Related Reading:** For more information on coding the controller's macros, see *IBM 4700 Finance Communication System: System Summary*.

## Writing the Controller Application Program for XRF Systems

If your SLU P system operates in an XRF complex, your controller application program must be able to handle messages lost during an XRF takeover. Many messages that are "in-flight" when the alternate system takes control are recovered with no action from the application program or terminal operator. In this case, the takeover is transparent.

When the takeover is not transparent, message DFS3861I is issued in one of the following combinations:

- Message DFS3861I only—If the output message is recoverable, this in-doubt message is issued. If irrecoverable, this message is not issued. If you are receiving output from IMS, and the new active IMS system cannot determine if you received the message, IMS requests an exception response but is not receiving one from your program.
- A CANCEL command followed by message DFS3861I—multi-segment output is in progress (last-in-chain not yet sent). If the in-flight message is recoverable, you should receive the in-flight message again. If the in-flight message is irrecoverable, the next in-flight message (if one is on the queue) is issued.
- An exception response followed by message DFS3861I—your input message is lost. Resend your last input message.

### Converting Controller Application Programs from Finance to SLU P

All current functions available for logical units defined to IMS as Finance, except the SCAN/NOSCAN option and Finance MFS, have similar functions for logical units defined as SLU P. IMS does not scan output for the control character sequence when sending output to a secondary logical unit type P. Therefore, the implied option is NOSCAN. If an application program uses MFS, it must be converted to use the distributed presentation management (DPM) option, which divides responsibility for message formatting between MFS and a user program residing within the logical unit. With the use of the DPM option, physical terminal characteristics are not defined to MFS. MFS formats and presents data to the user program component of the SLUTYPEP. The user program must complete the formatting, if necessary, and present the data to a physical device.

To execute correctly when defined to IMS as SLU P, existing remote user-written programs that run in a Finance controller must be converted as follows:

- Any input message headers must be converted to SLU P message headers.
- Any input message indicating only begin-bracket must also indicate change-direction.
- If the TERMINAL macro SCAN option is specified, the Finance controller application program must be converted to detect and process the same options provided by the SCAN option.
- If the controller application program uses MFS, the remote application must be converted to use the device-independent MFS distributed presentation management option.
- When converting from Finance-specific MFS to DPM, the MFS format descriptions must be redefined and recompiled using the IMS MFS Language utility.

---

### VTAM Facilities Used

The physical transmission of data between IMS and a SLU P system is controlled by VTAM.

**Related Reading:** For more information on the communication facilities that govern data transmission between IMS (a VTAM application program) and the controller, see *Network Program Products General Information* and *VTAM V4R3 for MVS/ESA Programming*.

The VTAM facilities that IMS uses, particularly for a SLU P system, include:

- Connection, disconnection, and establishing logon mode
- Messages and responses
- Definite response 1 and definite response 2 <sup>38</sup>
- Sequencing and chaining
- Quiescing

---

38. All data transmitted between IMS and a workstation must request a definite response 1, definite response 2, definite response 1 and 2, or exception definite response 1 or 2. VTAM commands must request definite response 1.

- Facilities for ensuring orderly communication, including the use of brackets<sup>39</sup> and change-direction indicators<sup>40</sup>
- Sequence-number recovery
- Receiving input, sending messages
- Unconditional bracket termination
- Primary Error Recovery Procedure (ERP)

The ERP is terminated by an IMS output error message sent to the workstation.

**Related Reading:** For more information on IMS error handling, see “Error Handling” on page 459.

IMS also supports the class of service (COS) and session outage notification (SON) facilities.

**Related Reading:**

- For more information on the COS and SON facilities, see “Using SON/COS Support in IMS” on page 74.
- For more information on communications concepts and facilities, see *Systems Network Architecture Technical Overview* and *Systems Network Architecture Format and Protocol Reference: Architectural Logic*.

---

## VTAM Commands and Indicators

VTAM commands and indicators (communication control information) are necessary for data transmission between an IMS application program and another VTAM logical unit. Table 40 shows which VTAM commands and indicators IMS sends to and receives from the SLU P controller. Results are unpredictable if any unsupported commands or indicators are used.

With the exception of the request-recovery command and the change-direction indicator, which are described following the table, the commands and indicators operate as described in *Network Program Products General Information*.

Table 40. VTAM Commands and Indicators Sent and Received by IMS

VTAM Command/Indicator	IMS Sends to Controller	IMS Receives from Controller
INDEPENDENT OF SESSION		
Initiate Session		X
Procedure Error	X	
Terminate Session		X
SESSION CONTROL		
Bind	X	
Clear	X	
Request-Recovery (RQR)		X
Set-and-Test-Sequence Numbers (STSN)	X	
Start Data Traffic (SDT)	X	

39. In SNA protocol, a bracket is one or more chains of request units (RU) and the responses that are exchanged between two LU-to-LU half sessions that represent a transaction.

40. In VTAM, a change-direction indicator means that the sender has finished sending and is prepared to receive.

## VTAM Commands and Indicators

Table 40. VTAM Commands and Indicators Sent and Received by IMS (continued)

VTAM Command/Indicator	IMS Sends to Controller	IMS Receives from Controller
Unbind	X	
NORMAL (synchronous) FLOW		
Begin-Bracket (BB)	X	X
BID	X	
CANCEL	X	X
CHASE		X
End-Bracket (EB)	X	X
Logical Unit Status (LUS)		X
Quiesce Complete (QC)	X	
Ready-to-Receive (RTR)		X
Change Direction (CD)		X
EXPEDITED (asynchronous) FLOW		
Quiesce-at-end-of-chain (QEC)		X
Release Quiesce (RELQ)		X
Request-Shutdown (RSHUTD)		X
Shutdown (SHUTD)	X	
Shutdown Complete (SHUTC)		X
Signal		X

**Recommendation:** During a SLU P session, every input message must have either a change-direction (CD) or end-bracket (EB); otherwise, the session is terminated.

## Request-Recovery Command

Upon receipt of the VTAM Request-Recovery (RQR) command, IMS issues a CLEAR. The SLU P application programmer must exercise care when sending this command while a recoverable IMS output message is in progress. To send an RQR command to IMS, the application program must first respond to the output operation that is in progress. If the application program does not send the reply but instead sends only the RQR command, the controller automatically sends the outstanding DR2 reply at the LEXIT or next LREAD statement. If IMS receives the RQR command before the DR2 reply, the CLEAR generated by IMS can cause the DR2 reply to be lost. The status of the current output message is then unpredictable.

**Related Reading:** For an important restriction, see “Message Resynchronization” on page 426.

## Change-Direction Indicator

IMS supports use of the change-direction indicator on the last chain element of an input message or a VTAM command. This allows device support to be more consistent with VTAM operation. The operation of IMS is not affected by the indicator.

---

## Establishing Connection and Specifying Logon Modes

When establishing connection using the VTAM OPNDST macro instruction, IMS specifies session parameters that define the rules that a logical unit must follow when communicating with IMS.

IMS examines either a user-supplied or a VTAM default set of session parameters, and overlays only those parameters on which IMS has dependencies. The remaining bytes are not changed. You can include user data with the BIND parameters.

**Related Reading:** For information on the BIND parameters for SLU P, see “Appendix B. Bind Parameters for SLU P and LU 6.1” on page 471.

User data included within the BIND is used by IMS as input to the signon process and is similar in function to the IMS /SIGN command. This is required for ETO Finance terminals and SLU P terminals, but is optional for static terminals.

If you do not specify a mode name on the TERMINAL macro statement during IMS system definition or on the logon descriptor, you can supply a mode name using any of the following methods:

- On the VTAM network operator VARY command
- On the IMS OPNDST command
- On a request for system initialization by another logical unit

If you specify a mode name on the IMS TERMINAL macro statement or on the logon descriptor, it is used, unless you override it using the IMS /OPNDST command or the VTAM VARY command.

If you do not supply a mode name during IMS system definition using a logon descriptor, on an IMS command, or using CINIT, VTAM assumes a default mode name.

**Related Reading:** For more information on establishing logon mode tables and defining logon mode table entries, see *VTAM V4R2 for MVS/ESA Resource Definition Reference*.

You can override the use of the default logon mode table entry in one of two ways:

- At system definition, you can specify the logon mode entry on the TERMINAL macro using MODETBL=keyword.

**Related Reading:** For information on specifying the MODETBL keyword, see *IMS/ESA Installation Volume 1: Installation and Verification*.

- You can specify on the MODETBL keyword of the /OPNDST command that the logon mode table entry replace the table entry defined at system definition.

**Related Reading:** For information on specifying the /OPNDST command, see *IMS/ESA Operator's Reference*.

IMS overrides the session parameters for function management, transmission services, and primary and secondary network protocol.

### Establishing Connection with the XRF Complex

To establish a session with IMS in an XRF complex, your logon request to VTAM must specify the IMS USERVAR. (VTAM matches the USERVAR to the IMS application that is currently active.)

**Related Reading:** For information on defining USERVAR to VTAM, see *IMS/ESA Administration Guide: System*.

If your SLU P terminals have programs that read the primary logical unit name (PLUNAME) field of the BIND to determine which application with which to open a session, you cannot continue to use this field in the XRF environment. Instead, the USERVAR is added to the user data field of the BIND data.

**Related Reading:**

- For information on the contents of the BIND data, see “Finance Communication System Bind Parameters” on page 471.
- For specific information on where IMS adds the USERVAR variable to the user data field, see Note 5 in Table 46 on page 471.

SLU P systems are normally defined as Class 1 terminals; the primary sessions switch to backup sessions when the alternate IMS system takes over the processing of the IMS workload. In most cases, the takeover is accomplished without loss of the session. The terminals might or might not be aware of the takeover.

**Related Reading:** For information on takeovers that are not apparent, see “Writing the Controller Application Program for XRF Systems” on page 403.

If you are connecting the 4730 Personal Banking Machine to an XRF IMS as a SLU P device, use the following machine data configuration options:

Invalid Message - 10 or 11  
Multiple Commands - 10 or 11

The first digit (1) prevents the 4730 from sending an exception response to certain network messages, including the DFS3861 message sent after an XRF takeover. Exception responses to these messages terminate the SLU P session between the 4730 and IMS. The second digit (0 or 1) indicates whether errors should be logged.

**Related Reading:** For more information on the 4730, see *IBM 4730 Personal Banking Machine Operations Support Manual*.

---

## Bracket and Send/Receive Management

The change-direction and bracket indicators are used to begin, end, and control the direction of synchronous transmissions between a SLU P system and IMS. IMS support of these indicators is summarized in Table 41.

*Table 41. Use of VTAM Bracket and Change-Direction Indicators*

<b>X = supported</b>	<b>BB</b>	<b>EB</b>	<b>BB/EB</b>	<b>BB/CD</b>
Received by IMS with data	X		X	X
Sent by IMS with data		X	X	



## Bracket and Send/Receive Management

Table 41. Use of VTAM Bracket and Change-Direction Indicators (continued)

<b>X = supported</b>	<b>BB</b>	<b>EB</b>	<b>BB/EB</b>	<b>BB/CD</b>
Received by IMS with data flow synchronous commands (normal flow)		X		

IMS does not support the absence of bracket indicators or the use of end-bracket-only on inbound synchronous data. IMS does not send output synchronous data with begin-bracket (BB) only, change-direction (CD) only, or BB/CD. Because IMS always ends a bracket (unconditional bracket termination) on input synchronous data, it is unnecessary to send either EB or CD on output synchronous commands. Input synchronous commands can specify EB or CD, but must be consistent with the current bracket and send/receive states; otherwise, the session is terminated. IMS does not send VTAM indicators with data flow synchronous commands (normal flow).

## Bracket and Send/Receive Management

---

## Chapter 20. IMS Facilities Used for SLU P and Finance

This chapter describes in detail the IMS facilities that support the Systems Network Architecture (SNA) environment.

### In this Chapter:

- “Component Definition”
- “Terminal-Response Mode” on page 413
- “Defining a Workstation for Terminal-Response Mode” on page 415
- “Output Messages Sent While in a Between-Brackets State” on page 416
- “Designing for Output Messages Sent While in Between-Brackets State” on page 418
- “IMS Message Format Service” on page 418
- “Display Screen Protection for Finance Stations” on page 421
- “Extended Output Component Protection (SLU P)” on page 422
- “Input and Output Editing Options (SLU P)” on page 423
- “Use of Responses or Brackets to Acknowledge Recoverable Input” on page 424
- “Message Recovery” on page 425
- “Message Resynchronization” on page 426
- “Finance and SLU P in an XRF Complex” on page 427
- “Fast Path Messages with Finance and SLU P” on page 427

---

## Component Definition

**Definition:** IMS considers a *workstation* to be one physical terminal. If a workstation is made up of more than one device, each physical device that makes up the workstation must be defined as a component of that workstation. As such, each component can have an IMS logical terminal (LTERM) associated with it. One LTERM is associated with each component. A user-written MPP can address specific components of a workstation through the appropriate LTERM.

IMS assumes that all input is from the first LTERM in the component list that passes the necessary operational and security checks. Message switches, broadcast messages for specific logical terminals, and data replies to transactions are directed to the component associated with the specified output LTERM.

## LTERM Naming

To facilitate the use of the CHNG call, define a convention for naming LTERMs. One method is to set the LTERM name to be a combination of the workstation and component identifiers.

**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 can have three components: WS100DS (4704), WS100MS (4706), and WS100RP (4710). Such a standard permits an MPP to interrogate the I/O PCB (LTERM name field) to identify the workstation and then to specify the proper alternate PCB for output using the CHNG call.

## Component Definition

For a multiple-component terminal, such as a SLU P, you must ensure that IMS creates enough queues to service the terminal. You can use an exit routine or a specific user descriptor for that unique logical unit to ensure that IMS creates the required number of queues.

## Output Component Selection

IMS system definition allows a workstation to have a maximum of four output components. If more than four output components are required for a workstation, or multiple LTERMs are not desired, a user-defined MPP-to-workstation protocol and data format must be provided. Workstation components are assigned an identification number: X'01', X'02', X'03', and X'04'. For Finance terminals, the identification number is based on the order in which they are defined. For SLU P terminals, the identification number defaults to Comp<sup>1</sup>, (Program1, Basic). IMS nonqueued system messages have no specific destination and are directed to the first component.

An IMS Transaction Input edit routine can be used to append the station's node name to the input message. The MPP can reference the node name to determine which LTERM name to use for output. For terminals created using the Extended Terminal Option (ETO) feature, append the user name to the input message because the user could be signed off.

**Related Reading:** For more information on the ETO feature, see "Chapter 7. Administering the Extended Terminal Option" on page 123.

To return a reply message to a workstation operating in terminal response mode, or to a conversational transaction, the MPP must insert the reply to the I/O PCB. However, for some stations, that reply message might be intended for a component other than the one associated with the LTERM specified in the I/O PCB. Because the destination of the I/O PCB cannot be modified by the CHNG call, an alternate PCB type is required. The alternate PCB type must fulfill the above response requirements and also be modifiable. The PCB type, called the response alternate PCB, can be defined using PSBGEN. This PCB can be used instead of the I/O PCB to return reply messages to conversational transactions and to stations operating in terminal response mode. It can also be defined as modifiable. This allows the CHNG call to be used with this PCB to select the appropriate destination for the reply. When the CHNG call is used, the LTERM specified in the call must be assigned to the same physical terminal as the LTERM specified in the I/O PCB. If this is not the case, a status code is returned on the ISRT call.

## Input Component Determination

Proper relationships between input and output components can be established using the NAME macro during IMS system definition, or by using an ETO user descriptor and the Signon exit routine. This relationship allows the terminal to specify its input component and causes output to be returned to a component that is defined 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.

A user can establish a connection for SLU P usage by defining a component for each of the operators, devices, or processing requirements controlled by a remote application program.

IMS performs input component selection for the remote logical units and assumes that all input is from the first LTERM in the list that passes the necessary operational and security checks. For input from SLU P, input component selection is performed based on the component indicated in an optional input message header. If no function management header is received, IMS assumes the input is to be associated with the LTERM for component 1. Message switches, broadcast messages for specific LTERMs, and data replies to transactions are directed to the component associated with the specified output LTERM.

---

## Terminal-Response Mode

**Definition:** *Terminal-response mode* is a mode of operation that can be defined for transactions, users, and terminals attached to IMS. When a SLU P station is operating in terminal-response mode, all operations are stopped between the workstation and IMS from the time IMS receives a transaction until IMS receives acknowledgment that the reply message has been received by the workstation. For normal IMS output messages, this acknowledgment is the receipt of the DR2 response requested for recoverable transaction output. For Fast Path output messages, this acknowledgment is the receipt of the next input message or an RTR command. For MFS-paged output messages, this acknowledgment is the receipt of the next input message, MFS NEXTMSG or NEXTMSGP control commands, or a requested DR2 response (if not Fast Path output) on the last page of the message. Output caused by a nonrecoverable transaction requests an exception DR2 response and does not require a response from the workstation. Terminal-response mode can reduce the processing required by the controller application program.

Terminal-response mode can be selected by an installation during IMS system definition or with an ETO user descriptor. Your system can be defined as forced, negated, or transaction-dependent:

- If it is defined as forced, every input transaction is in terminal-response mode.
- If it is defined as negated, no input transaction is in terminal-response mode.
- If it is defined as transaction-dependent, terminal response mode is determined on a transaction-by-transaction basis. The use of transaction-dependent terminal-response mode can make the controller application programs more complex, because they must handle communication protocols resulting from both forced and negated terminal-response mode environments.

Terminal-response mode can only be invoked by valid transactions, not by message switches, IMS commands, VTAM commands and indicators, or MFS control requests. Transactions that are unacceptable (for example, because of a security violation or an invalid transaction code) cannot invoke terminal-response mode.

When a workstation is operating in terminal-response mode, the following processing occurs after the workstation has established a session with IMS (see Figure 62 on page 414):

- 1** The workstation sends an input transaction.
- 2** IMS places the workstation in terminal-response mode.
- 3** IMS passes the transaction to a message processing program (MPP).
- 4** The MPP processes the transaction.
- 5** The MPP returns a reply, using either the I/O PCB or an response alternate PCB.
- 6** IMS returns a DRx response, if one is requested.
- 7** IMS sends the reply to the workstation.

## Terminal-Response Mode

- 8** The workstation returns a DR2 response, if one is requested.
- 9** IMS removes the workstation from terminal-response mode.

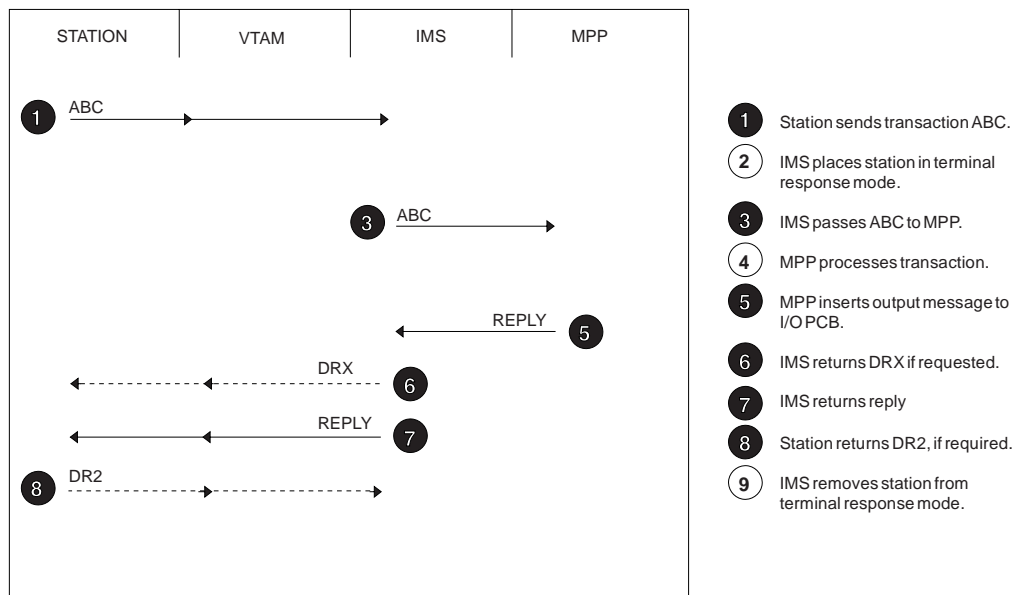


Figure 62. Terminal-Response Mode Operation

While a workstation is in terminal-response mode, IMS accepts no input from it, and sends no output to it other than the reply from the MPP. If the MPP abends while processing the input transaction, IMS might send an exception response and the associated IMS error message. Any output that is not in reply to the transaction that initiated terminal-response mode is held in IMS's output queue. After IMS removes the workstation from terminal-response mode, it sends the unsolicited output.

When Fast Path is used, the following processing occurs (see Figure 63 on page 415):

- 1** The workstation sends an input transaction.
- 2** IMS places the workstation in terminal-response mode.
- 3** IMS passes the transaction to a Fast Path message processing program (IFP).
- 4** The IFP processes the transaction.
- 5** The IFP returns a reply, using either the I/O PCB or a response alternate PCB.
- 6** IMS returns DRx, if one is requested.
- 7** IMS sends the reply to the workstation.
- 8** An exception DR2 response is required on output; therefore, the workstation sends the next input message or RTR command. If IMS has output for the workstation on the message queue, a definite response is required. The workstation sends DR2.
- 9** IMS removes the workstation from terminal-response mode.

## Defining a Workstation for Terminal-Response Mode

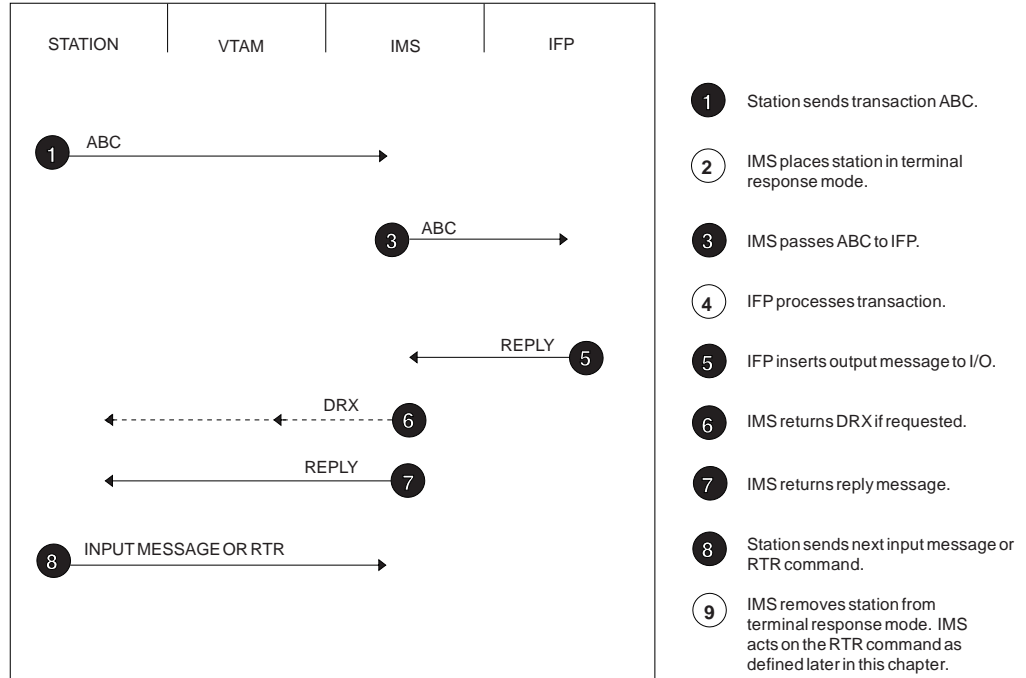


Figure 63. Fast Path Transaction

## Defining a Workstation for Terminal-Response Mode

You must understand the operation sequences depicted in Figure 62, Figure 63, and Figure 64 on page 417 before defining a workstation to operate in terminal-response mode. Less processor time is required for stations that operate in terminal response mode, and the controller application program that controls such a workstation might also be less complex. The following considerations also apply:

- Typical data collection applications cannot be performed from stations that operate in terminal-response mode. In this mode, a reply from the MPP is required for each transaction before IMS accepts another transaction. Waiting for this response extends the time of the data entry process.
- IMS does not send a response to an input message until the output reply is available. Thus, when the response is returned, it indicates that the input has been processed and the application has reached a sync point.
- While a workstation is in terminal-response mode, IMS does not attempt to obtain any more input from that station. Master terminal operator intervention is required if an error prevents creation or transmission of a reply. Some conditions can prevent a reply being sent to the workstation. These include:
  - The LTERM stopped.
  - IMS was unable to schedule an MPP (database stopped, MPP stopped).
  - An MPP logic error caused no reply to be returned except for EMH (expedited message handler), which generates a zero-length reply.

If any of these conditions occurs, the workstation is temporarily inoperative. Before the workstation can be used again, the master terminal operator must diagnose and correct the error.

- A response message remaining on the IMS output queue or inserted by the user-MPP after session termination is re-sent after initiation of the next session. If

## Defining a Workstation for Terminal-Response Mode

the BID option is specified, this message is preceded by the VTAM BID command. Both the begin-bracket and end-bracket indicators are sent with the message.

System analysts must evaluate these factors for their own system application programs and operating environment.

---

## Output Messages Sent While in a Between-Brackets State

If an IMS output message is sent while the workstation is not protected from output and in a between-brackets state, IMS sends either:

- The message with both the begin- and end-bracket (that is, bids for bracket with data) if the workstation is defined with the NOBID option
- The BID command to request permission to begin a bracket if the workstation is defined with the BID option.

**Related Reading:** For information on when an IMS output message is sent while the workstation is not protected from output, see “Display Screen Protection for Finance Stations” on page 421.

A workstation can accept the bracket with any requested DR1 or DR2. A DR1 to the bid causes IMS to send the output message with both begin- and end-brackets.

A workstation can reject the bracket with an appropriate exception DR1 or DR2.

**Related Reading:** For information on the IMS operation that follows a bracket rejection, see “Controller or Station-Detected Errors” on page 460.

When a workstation is ready for output after rejecting a bracket, it sends the ready-to-receive (RTR) command to request output from IMS.

Figure 64 shows how IMS handles output for a workstation that is in terminal-response mode and that is defined with the BID option when the session is between-brackets.



## Output Messages Sent While in a Between-Brackets State

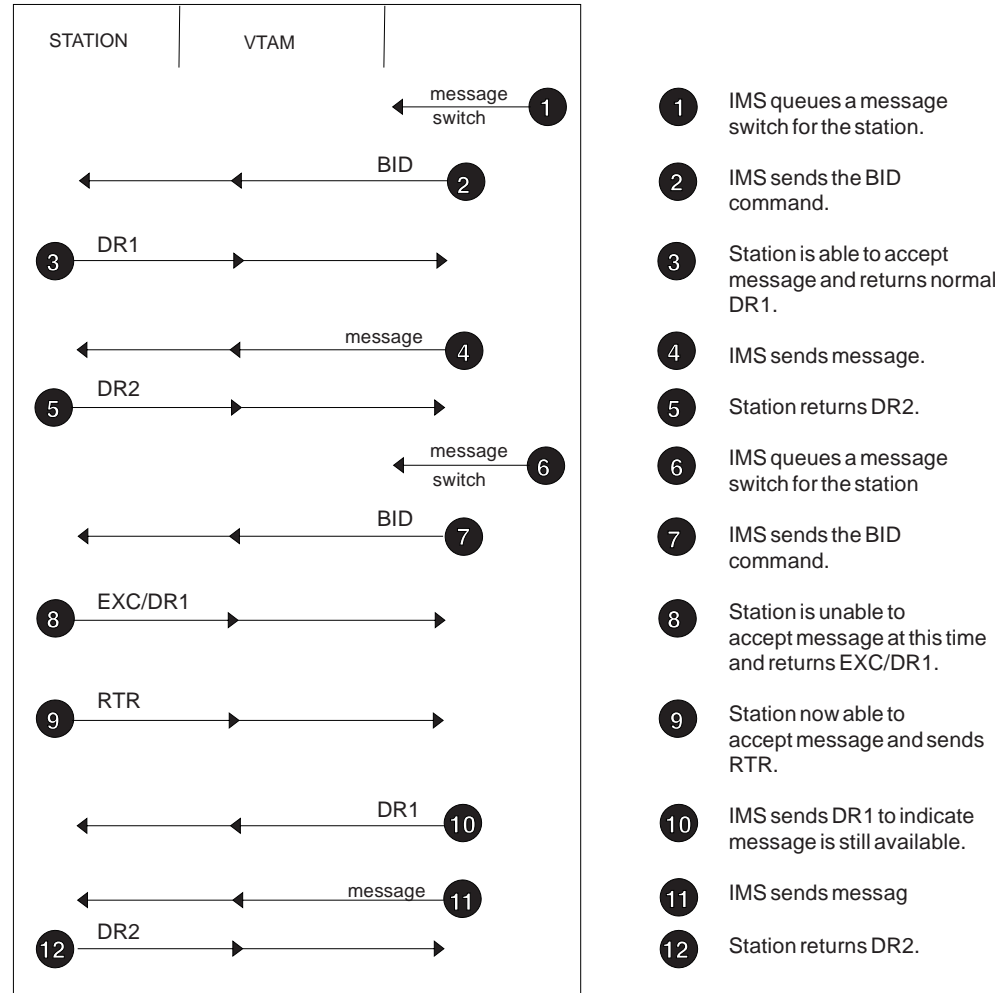


Figure 64. Output Sequences while Between-Brackets for a Workstation in Terminal-Response Mode with the BID Option

The flow shown in Figure 64 is as follows:

- ① When IMS receives a message switch for a workstation defined with the BID option, IMS places the message in the output queue.
- ② After removing the workstation from terminal-response mode, IMS sends a BID command to notify the workstation that output is pending while between-brackets. IMS requests a DR1 response to the BID command.
- ③ The workstation returns the DR1 response if it is ready to receive the output.
- ④ IMS sends the output when it receives the DR1.
- ⑤ The workstation returns a DR2 response, if requested.
- ⑥ ⑦ ⑧ If the workstation is not ready to receive the output, it returns an exception DR1 response indicating “bid rejected”. If the output is recoverable, IMS returns the message to the queue and waits until the workstation indicates it is ready to accept the output. If the exception response indicates “RTR will not follow” and the output is nonrecoverable, IMS discards it. If the exception response indicates “RTR will follow,” IMS returns the message to the queue (regardless of recoverability) and waits for a VTAM ready-to-receive (RTR) command.

## Output Messages Sent While in a Between-Brackets State

- 9** When the workstation is ready to receive the output, the workstation sends the VTAM RTR command and requests a DR1 response.
- 10 11** If the output message that caused the bid is still available, IMS returns the DR1 response, followed by the output message.
- 12** The workstation returns a DR2 response (12).

While IMS waits for the RTR command from the workstation (items **8** and **9** in Figure 64 on page 417), the workstation can perform any type of processing desired. IMS accepts and responds to transactions during this time, but does not transmit any unsolicited output until it receives the RTR command.

If IMS no longer has output available to send when the workstation sends RTR (perhaps because of an intervening /ASSIGN or /DEQUEUE command), or if the data was irrecoverable and IMS discarded it, IMS returns an exception DR1 response and an error message indicating no output is available.

---

## Designing for Output Messages Sent While in Between-Brackets State

If IMS receives an exception response to an irrecoverable output message (inquiry-only transaction), the message is lost, because IMS dequeues the message immediately upon sending it to VTAM. When IMS is between-brackets, it sends the BID command to bid the workstation before actually sending an output message. If the bid is rejected, the recoverable output message remains queued if the exception sense data indicates an RTR is forthcoming; otherwise, the message is discarded. These concepts are depicted in Figure 62, Figure 63, and Figure 64 on page 417, and should be reviewed in conjunction with the IMS actions described later under “Controller or Station-Detected Errors” on page 460.

---

## IMS Message Format Service

This section describes some IMS MFS facilities that specifically apply to the SLU P system.

**Related Reading:** For more information on IMS MFS facilities for SLU P, see *IMS/ESA Application Programming: Transaction Manager*.

## Designing MFS for the Workstation Environment

Both input and output data can be processed by MFS. The exit for the Input Transaction edit routine is available to provide additional editing capabilities.

The availability of MFS to workstations is defined on an individual basis by the system administrator during IMS system definition. When MFS is defined as available, each input or output message can optionally be processed using MFS. The format of messages to be processed by MFS is defined using the IMS-supplied MFS Language utility.

Operations using MFS can be quite different from operations using the IMS Basic Edit facilities. Device formats and operator procedures should be designed carefully, with the objectives of easy use and high-operator productivity.

MFS provides two levels of message formatting: device-level message formatting and distributed presentation management (DPM). Device-level message formatting is for Finance logical units. DPM support is for SLU P logical units.

To prevent null screens, a null record (data length = 0) produced by MFS as the result of a user-specified MFS format description is not sent by IMS.

### MID/MOD Chaining

MFS input formatting for a workstation occurs when a message input descriptor (MID) name is provided with an input message. The MFS message output descriptor (MOD) can supply a MID name to be used for formatting the next input message. It is the responsibility of the workstation to supply this MID name when it sends the input message. This results in supplying the MID name.

**Related Reading:** For more information on the MID name, see “Activating MFS Input Formatting for Finance Workstations” on page 446.

MID/MOD chaining can be accomplished with little or no intervention by the workstation operator by observing the following procedure during the design of SLU P system application programs:

1. Remove the MID name from the received output message header and save it for use on the next input message.
2. Display or print the output message.
3. Get the next operator input.
4. Add the MID name (saved above) to the transaction.
5. Send the transaction to IMS.

**Related Reading:** For more information on the MID/MOD chaining concept, see “MFS Application Design” in *IMS/ESA Application Programming: Transaction Manager*.

### MFS Output Formatting for the SLU P System

MFS can be used to format pages, produce standard headings and footings, and provide forms control. Without MFS, the message processing program (MPP) or the controller application program must provide these functions.

Using MFS can also provide MPP device independence and allow SLU P components to be used for low-volume applications invoked from any of several terminal types. This device independence allows the controller application program to concentrate on high-volume applications, reducing its complexity and maintenance work.

The availability of MFS to a workstation is defined on a station-by-station basis by the system administrator during IMS system definition or ETO logon descriptor definition.

### MFS Message Recovery

Input messages processed by MFS are edited before the message is queued and logged. Output messages are edited immediately prior to transmission; therefore, for recovery purposes, an input message is formatted by MFS and an output message is formatted by the IMS application program.

## IMS Message Format Service

### MFS Control Functions (Finance)

The control functions available to the operators of devices using MFS are:

Page Advance (NEXTTPP)	Transmit the next physical page of the current message, if one exists.
Logical Page Advance (NEXTLP)	Transmit the first or only physical page of the next logical page of the current message.
Logical Page Requests (PAGEREQ=nn where nn is the number of the logical page desired)	If the request is valid, transmit the data fields defined in the specified logical page of the current message.
Message Advance Protect (NEXTMSGP)	Discontinue printing or displaying the current output message and begin transmitting the next message in the output queue. If none exists, return an exception DR2 to notify the operator.
Message Advance (NEXTMSG)	Discontinue printing or displaying the current output message and begin transmitting the next message, if any.

These control functions can be indicated to MFS by including them in input function management headers or by defining an operator control field within the input data to MFS.

### MFS Control Functions (SLU P)

The following device-level MFS control functions are available to logical units defined as SLU P. SLU P terminals process the DPM-formatted output messages that specify paging option on the MFS device format. The paging option can be OPTIONS=DPAGE or OPTIONS=PPAGE.

Page Advance (NEXTTPP)	If a current message with OPTIONS=PPAGE is defined, transmit the data fields defined in the next presentation page. If a current message with OPTIONS=DPAGE is defined, transmit the data fields defined in the next logical page.
Logical Page Advance (NEXTLP)	Transmit the data fields defined in the next logical page of the current message if one exists and if either OPTIONS=PPAGE or OPTIONS=DPAGE is specified.
Logical Page Request (PAGEREQ=nn, where nn is the number of the logical page desired)	If the request is valid, transmit the data fields defined in the specified logical page of the current message.
Message Advance Protect and Message Advance (NEXTMSGP and NEXTMSG)	Discontinue transmitting the current output message and begin transmitting the next message in the output queue.

The control functions of NEXTTPP, NEXTLP, NEXTMSGP, and NEXTMSG can be specified in input function management headers or by defining an operator control field within the input data to MFS. Logical page requests can only be entered within the input data or using the operator control field.

## MFS Paging and BID Options

The BID option provides output protection but has a high performance cost (an extra line flow for each message). When the BID option is specified and the output occurs while between-brackets, IMS waits for a positive response to BID before sending output. To avoid sending BID while sending MFS-paged output, each input paging request should indicate begin-bracket and change-direction. Each output page then contains only end-bracket and no BID results, because the page is sent while in an in-brackets state.

---

## Display Screen Protection for Finance Stations

When a Finance station is defined with the NOBID option, IMS provides support similar to the screen protection function for the IBM 3270 Information Display System. IMS does not transmit two consecutive output messages to a display component without an intervening input message from the component. This gives the user the opportunity to view and respond to one message before another is displayed.

When IMS transmits an output message to a display component, it marks the component as protected, unavailable for output. IMS does not transmit another output message to it until an input message is received from that component. The input message can be an IMS transaction, a message switch, an IMS command, the VTAM RTR command, or an MFS control request. Upon receipt of one of these, IMS changes the display component's status to unprotected.

When a workstation is defined with the BID option, consecutive messages are not transmitted to a station. Input from the workstation or a positive response to the BID command from IMS must be received after one transmission before the next message can be sent.

When a workstation uses MFS, the screen is protected on a physical-page basis. MFS control requests or input data are used to request additional screens of data.

**Related Reading:** For more information on physical page control, see "MFS Control Functions (Finance)" on page 420.

If a workstation does not use MFS, the screen is protected on a message-by-message basis. The input of any IMS transaction, message switch, or command causes the screen to be unprotected. The VTAM ready-to-receive (RTR) command can be used to request the next output message. If no message is available, or if the node is in a status that does not allow output to be sent (output stopped or quiesced), IMS returns an exception DR1, followed by an error message indicating that no output is available.

If the RTR command is received during the transmission of MFS-paged output, IMS returns an exception DR1 response, followed by an error message indicating an invalid paging request, because IMS cannot determine the MFS control function to be performed.

After an exception response is received by IMS for a current IMS output message to a display component, the screen is unprotected. After an exception response is sent by IMS, any defined display component is automatically marked protected and unavailable for output.

### Extended Output Component Protection (SLU P)

When a SLU P is defined with the NOBID option, IMS provides support similar to the screen-protection function for the IBM 3270 Information Display System and display-screen protection for Finance components. IMS does not transmit two consecutive output messages to an output-protected component without an appropriate intervening input message from the LU. This gives the LU an opportunity to process one message, according to user-defined procedures, before another is sent to the same component. SLU P support allows the output of all components to be protected.

IMS marks the component as protected and unavailable for output under the following conditions:

- When IMS transmits a message to a component defined as PROGRAM2 on the IMS TERMINAL macro
- When IMS transmits a message (or a page of a message) formatted by MFS with a paging option defined

While a component is protected, IMS does not transmit another output message to the component until an input message from the logical unit resets the component to unprotected. Input messages that can reset a component's status to unprotected are:

- An IMS transaction with an FM header
- A message switch with an FM header
- An IMS command with an FM header
- An MFS control request with an FM header indicating a specific component to be unprotected
- Any of the above, either without an FM header or with an FM header indicating component zero (which resets protection on all components of a terminal)
- An RTR command

After the component's status is reset to unprotected, IMS sends any available output. If a component is defined as PROGRAM2 and is not described by a device format with paging OPTIONS=DPAGE or OPTIONS=PPAGE, output to the component is protected on a message-by-message basis. The input messages are used to reset output component protection.

If a SLU P component is defined to use MFS DPM and the device format used is defined with the paging option OPTIONS=DPAGE or OPTIONS=PPAGE, the output component is set to protected when a logical page or presentation page is sent to the component. MFS control requests or input data are used to reset output component protection and request additional logical or presentation pages from MFS.

**Related Reading:** For more information on logical page control, see "MFS Control Functions (SLU P)" on page 420.

During the transmission of MFS-paged output, the only allowable input is:

- Input containing no message header (resets all components' protection).
- Input containing a header indicating the specific component to which MFS is currently paging. Otherwise, the session is terminated, because IMS cannot simultaneously send output to more than one component of a terminal.

## Extended Output Component Protection (SLU P)

If a SLU P terminal is defined with the BID option, output component protection occurs after every message. In this mode of operation, consecutive messages are not transmitted to a terminal. Input from the terminal must be received; otherwise, if additional output is queued, a BID command is sent after one transmission before the next message is sent.

If a terminal does not use MFS, screen protection is performed on a message-by-message basis. The input of any IMS transaction, message switch, or command causes the screen to be unprotected.

If the ready-to-receive (RTR) command is received and no messages are available, or if the terminal is in a status that does not allow output to be sent (output stopped or quiesced), IMS returns an exception DR1, followed by an error message indicating no output is available. If the RTR command is received during the transmission of MFS-paged output, IMS returns an exception DR1 response, followed by an error message, indicating invalid paging request because IMS cannot determine the MFS control function to be performed.

After any exception response is sent by IMS, any display component defined for the logical unit is automatically marked protected and unavailable for output.

After an exception response is received for a current IMS output message to a component defined as PROGRAM2 or for an MFS DPM-paged output message, the component to which the output was sent is left unprotected.

After any exception response is sent by IMS, all components defined as PROGRAM2 are automatically marked protected and unavailable for output.

---

## Input and Output Editing Options (SLU P)

The type of editing subparameter in the COMPTn parameter of the system definition TERMINAL macro and the Extended Terminal Option (ETO) logon descriptor can be indicated at a component-by-component level for input to and output from components defined as SLU P.

**Related Reading:** For more information on the ETO feature, see “Chapter 7. Administering the Extended Terminal Option” on page 123.

The four types of editing provided are:

- |                   |  |
|-------------------|--|
| <b>BASIC</b>      | Requests that no deblocking be performed on input to IMS. MFS cannot be used on input or output.   |
| <b>BASIC-SCS1</b> | Requests that deblocking occur when an SNA character string (SCS)-defined new line (NL-X'15') or form feed (FF-X'0C') character is sent to IMS. MFS is not used on input or output.      |
| <b>MFS-SCS1</b>   | Requests that deblocking occur when an SCS-defined new line (NL-X'15') or form feed (FF-X'0C') control character is sent to IMS. MFS-SCS1 formats can be used for both input and output. |
| <b>DPM-An</b>     | Allows messages to be formatted using MFS distributed presentation management (DPM). No deblocking on input to IMS is performed. 'n' is a number from 1 to 15.                           |

For DPM, the device type symbolic name is specified to MFS as DPM-Xn, where X is A or B. Message formatting is specified on the TERMINAL macro for logical units defined as SLU P by using

## Input and Output Editing Options (SLU P)

device type symbolic names of the form DPM-An. DPM-Bn device type symbolic names refer to logical units defined for Intersystem Communication.

If BASIC or SCS1 is specified, each message received or sent by IMS as one or more related transmissions forms a VTAM chain:

- For BASIC, each input transmission to IMS, less any supplied input FM header, is treated as an IMS segment to be presented to MFS or directly to the IMS message processing program.
- For SCS1, an IMS input segment is created at each SCS-defined new line or form feed control character. IMS segments can be created from a portion of, from all of, or from more than one spanning transmission.

For either BASIC or SCS1, each output segment is sent by IMS in a single transmission, unless it and any IMS-appended FM header are larger than the logical unit's buffer size as indicated to IMS by the OUTBUF parameter on the system definition TERMINAL macro or on the ETO logon descriptor. In this case, IMS sends the segment in as many transmissions as required. Each transmission, except possibly the last transmission of a segment, is the maximum-defined output buffer size.

**Related Reading:** For a definition of SCS1, see “Appendix E. SNA Character String Controls” on page 487.

When DPM or MFS-SCS1 is specified, input and output messages can be formatted by MFS DPM or MFS-SCS1, respectively, on a message-by-message basis. Messages that are not to be formatted are edited as previously described for BASIC or SCS1.

**Related Reading:** For information on MFS formatting, see *IMS/ESA Application Programming: Transaction Manager*.

---

## Use of Responses or Brackets to Acknowledge Recoverable Input

To facilitate message resynchronization, IMS allows an input update or a recoverable-inquiry transaction to optionally request a definite response. IMS allows this request for a definite response if the user specifies the OPTIONS=OPTACK for the workstation.

**Related Reading:** For more information on requesting a response, see the description of the TERMINAL macro in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

With this operand specified, IMS can acknowledge input with the next output through use of the input and output bracket indicators.

**Restriction:** OPTIONS=ACK is not supported for SLU P terminals created using the ETO feature.

**Related Reading:** For more information on ETO, see “Chapter 7. Administering the Extended Terminal Option” on page 123. The specific effects of the OPTACK option on message resynchronization and VTAM bracket indicators are described later in this manual.



## Use of Responses or Brackets to Acknowledge Recoverable Input

If the OPTACK option is defined, performance can be improved if the workstation requests begin-bracket, change-direction, and exception DR1 or DR2 (rather than DR1 or DR2) on recoverable input to IMS. The next output from IMS acknowledges the input by indicating end-bracket only. The type of output messages that are sent from IMS depend on the type of recoverable input, the defined response mode, and the availability of output.

**Related Reading:** For information on the types of output messages, see “Chapter 22. SLU P Message Protocols” on page 443 and “MFS Control Functions (Finance)” on page 420.

Contention can occur when a session is in a between-brackets state. In this case, IMS sends an unsolicited output message indicating both BB and EB or a BID command at the same time that the workstation is sending an input message indicating begin-bracket and change-direction. This output message does not acknowledge the input message and can be either accepted or rejected by the workstation. Rejecting an IMS nonrecoverable output message can result in losing the message. If contention occurs when IMS is sending a recoverable IMS output message (output requesting DR2 response), the workstation must send either an exception or a definite response before IMS can receive the input message.

**Related Reading:** For more information on rejecting an IMS irrecoverable output message, see “Output Messages Sent While in a Between-Brackets State” on page 416.

---

## Message Recovery

The process of message recovery is accomplished within IMS by using the IMS system log and the checkpoint and restart routines. Recovery is possible, because IMS has direct control of the communication link. With a SLU P system, however, this control is shared with the controller and the controller application program. If a network failure, such as a processor failure or an IMS abend, occurs while the controller application program is receiving a message from IMS, the traditional methods of IMS recovery cannot always detect the lost message. The IMS recovery methods are extended for the SLU P systems to include the controller application program for message resynchronization.

The controller application program participates in message resynchronization so that a lost message condition, if any, can be detected and corrected. Failure of the controller application program to perform its responsibilities during message resynchronization results in a loss of message integrity in the system.

Message resynchronization is necessary for a workstation when a session with the workstation is terminated between the time it sends a recoverable message and the time it receives a reply for that message. The type of resynchronization depends on how the message is defined.

For all messages except response mode, response conversational mode, and Fast Path, the input becomes recoverable when it has been successfully enqueued and made available for scheduling. During message resynchronization, IMS indicates the last successfully received recoverable message.

For response mode, response conversational mode, and Fast Path, the input is not made recoverable and restartable until the application reaches its first sync point. If the IMS system fails before this sync point, the message is considered to be

## Message Recovery

nonrestartable. During message resynchronization, IMS indicates whether the message has reached a sync point or needs to be reissued.

If a session with a workstation is terminated after IMS sends a message to the workstation but before IMS receives the response, message resynchronization is necessary for this workstation. The output message for which no response is received must remain associated with this workstation until message resynchronization determines whether the workstation received the message. If the /ASSIGN command is used to move the message to a different workstation, message resynchronization is no longer possible.

---

## Message Resynchronization

The purpose of message resynchronization is to guarantee the integrity of messages across sessions. Message resynchronization occurs at the start of a session unless IMS is cold started. Finance and SLU P sessions warm start using the control blocks created from the original descriptor, even though that descriptor might have been changed or deleted after IMS created the control blocks. Only a coldstart ensures that the control blocks that are created represent the new or updated descriptor.

When message resynchronization is necessary because of a network failure, the resynchronization must complete successfully before IMS permits normal data transmission. To initiate message resynchronization, IMS sends the VTAM set-and-test-sequence-numbers (STSN) command. The LU must respond to this command. To be able to respond properly, a copy of the following must be maintained:

- The sequence number of the last request unit (RU) of the last inbound sync-point message that was sent by the LU. The inbound sync-point message is one of the following:
  - The last successful recoverable input message (the one that requested a DR1 or DR2) if the ACK option was defined to IMS for this LU
  - The last input message, if the OPTACK option was defined for this LU
- The sequence number of the last request unit (RU) of the last outbound sync-point message received by the LU. The outbound sync-point message is the last successful recoverable output message. Recoverable output messages are those requesting DR2 responses.

If Fast Path is used, each output message is recoverable and requests an exception DR2. These Fast Path-recoverable messages are identified by a flag within the output message header, rather than by an explicitly requested DR2 response.

Optionally, the LU can maintain a copy of the last inbound recoverable message sent by the terminal. If this is done, the SLU P system can retransmit, after resynchronization, any message not received by IMS.

**Restriction:** Session initiation and resynchronization caused by an SNA Request-Recovery (RQR) command is not allowed when the node is in response mode and the response reply message is not yet available for output; that is, the input response mode transaction is still queued or in the process of execution. Response-mode transactions are not recoverable or restartable prior to the application sync point; therefore, session input acknowledgement does not occur until input processing is complete.

Session initiation or resynchronization results in session termination while the response mode transaction is in this 'in-doubt' state, and IMS is not able to indicate that the associated input sync-point request was committed or backed out. This temporary error condition is indicated to the master terminal operator via message DFS2081I. An IMS /DISPLAY command can be used to determine when the response mode reply message is available and session initiation can again be attempted. A display status of RESP-INP indicates input is still in process; RESP indicates input processing is complete and output is available for transmission.

---

## Finance and SLU P in an XRF Complex

Finance and SLU P sessions are handled as Class 1, 2, or 3 terminals. Class 1 attempts to provide a maximum level of transparency and performance by tracking the active IMS session with a standby session on the XRF backup system. Class 2 and 3 support involves terminating the active session if the IMS system fails. At takeover, the alternate IMS system automatically reinitiates the session and automatically signs on the original active system user if necessary. Class 3 support requires manual session restart and signon after XRF takeover.

### Related Reading:

- For information on establishing a Finance or SLU P session with an XRF complex, see "Establishing Connection with the XRF Complex" on page 408.
- For more information on XRF, see *IMS/ESA Operations Guide* and *MVS/ESA Planning: Extended Recovery Facility (XRF)*.

---

## Fast Path Messages with Finance and SLU P

When using Fast Path, the following options must be specified in the TERMINAL macro:

### **OPTACK**

Allows input messages containing begin-bracket and change-direction indicators to be acknowledged by an end-bracket indicator on the next output message.

### **FORCERESP or TRANSRESP**

Allows input of a response mode message to IMS.

To obtain best performance, Fast Path input messages should not request definite responses but should be coded with begin-bracket, change-direction, and exception DR2 response.

### **FPACK/NFPACK**

Determines if special Fast Path output protocols should be used.

Fast Path transactions are single-segment transactions and are defined as recoverable. Fast Path transactions must be defined as response mode.

**Related Reading:** For more information on Fast Path output messages, see "Fast Path Output Messages (Finance)".

## Fast Path Output Messages (Finance)

When defined with the system definition TERMINAL macro option FPACK or an ETO logon descriptor, Fast Path output messages are sent requesting an exception DR2 response. Fast Path output is sent requesting DR2 response when:

- Queued output is available to be sent to any nondisplay component.

## Fast Path Messages with Finance and SLU P

- The system definition TERMINAL macro option BID or an ETO logon descriptor option BID is defined, and queued output is available to be sent to a display component.
- The system definition TERMINAL macro option NOBID or an ETO logon descriptor option NOBID is defined, and queued output is available to be sent to a display component when the Fast Path output reply message is directed to a nondisplay component. The Fast Path output does not result in the display component being screen protected.

Unlike non-Fast Path output, the output message is left outstanding (not dequeued), and the workstation remains in response mode until the workstation sends data, a ready-to-receive (RTR) command, or the DR2 response to cause the message to be dequeued. The RTR command should be considered when no input is to be generated for an abnormally long time (for example, when the terminal operator plans to leave the terminal). The input message from the workstation, the RTR command, or the DR2 response acknowledges that the preceding output has been received and is recoverable; therefore, IMS might dequeue the output message, process any input message, or send any available output.

The system definition TERMINAL macro option NFPACK or an ETO logon descriptor indicates that the Fast Path output exception DR2 and next input acknowledgment protocol should not be used. In this case, Fast Path output messages are always sent requesting standard recoverable output message response protocols (DR2 response).

When the DR2 response is received, IMS dequeues the output message, removes the terminal from response mode, and sends any available queued output. Queued output is not sent to a display component if the Fast Path output reply message is sent to the same component because of the IMS screen protection support.

**Related Reading:** For more information on IMS screen protection support, see “Display Screen Protection for Finance Stations” on page 421.

## Fast Path Output Messages (SLU P)

When defined with the system definition TERMINAL macro option statement FPACK or an ETO logon descriptor, Fast Path output messages are sent requesting an exception DR2 response when any of the following occurs:

- Queued output messages are available to be sent to a component defined for the workstation as PROGRAM1.
- The system definition TERMINAL macro option BID or an ETO logon descriptor option BID is defined, and queued output is available to be sent to a component defined for the workstation as PROGRAM2.
- The system definition TERMINAL macro option NOBID or an ETO logon descriptor option NOBID is defined, and queued output messages are available to be sent to a program. This component is not output protected due to a previous output or this Fast Path message. (That is, multiple components might be protected or reset selectively as defined in this chapter.)

Unlike non-Fast Path output, the output message remains outstanding (not dequeued) and the terminal remains in response mode until the terminal sends data, a ready-to-receive (RTR) command, or the DR2 response to cause the message to be dequeued. The RTR command should be considered when no input is to be generated for an abnormally long time (for example, when the terminal operator plans to leave the terminal). The input message from the terminal, the RTR

## Fast Path Messages with Finance and SLU P

command, or the DR2 response acknowledges that the preceding output has been received and is recoverable; therefore, IMS might dequeue the output message, process any input message, or send any available output.

The system definition `TERMINAL` macro option `NFPACK` or an `ETO` logon descriptor indicates that the Fast Path output exception `DR2` and next input acknowledgment protocol should not be used. In this case, Fast Path output messages are always sent requesting standard recoverable output message response protocols (`DR2` response).

When the `DR2` response is received, IMS dequeues the output message, removes the terminal from response mode, and sends any available queued output for `PROGRAM1` components or non-output protected `PROGRAM2` components defined for the workstation. Output protection is set at the component level during output for `PROGRAM2` components and can be selectively reset for one or more components based on subsequent input.

**Related Reading:** For more information on output protection for `PROGRAM2` components, see “Extended Output Component Protection (SLU P)” on page 422.

Also use the `NFPACK` option for all 4730 terminals. This allows any asynchronous output messages to be sent immediately following the acknowledgement of the Fast Path output reply.

## Fast Path Message Resynchronization

If Fast Path is defined, sequence number management and message resynchronization might require change.

**Related Reading:** For more information on message resynchronization, see “Message Resynchronization” on page 426.

## Fast Path Messages with Finance and SLU P

---

## Chapter 21. Network Operation for SLU P and Finance

This chapter describes how to start an IMS network, how to initiate sessions, the different transaction types, message switching, and IMS commands.

### In this Chapter:

- “Starting an IMS Network”
- “Making IMS Ready”
- “Session Initiation (Starting Workstations)”
- “Suspending Output from IMS” on page 438
- “Session Termination” on page 439
- “Shutting Down an IMS Network” on page 441
- “SLU P Messages” on page 441
- “Send/Receive and Bracket Protocol” on page 441

---

### Starting an IMS Network

Before a session with IMS can be established for any workstation:

- VTAM and NCP must be active.
- Controllers and logical units that are not activated automatically by VTAM must be activated (with a VARY command) online by the VTAM network operator.
- Controllers must be powered on, and appropriately configured, initialized, and activated for VTAM and NCP.

---

### Making IMS Ready

IMS must be made ready to receive VTAM logon (session initialization) requests. Issue the IMS /START command with the DC keyword to make IMS ready. The /START DC command tells VTAM to pass any queued VTAM logon requests (and any logon requests for workstations known to VTAM as belonging to IMS) to IMS.

The /START DC command activates the following processes:

- Initiates IMS Data Communications processing
- Opens the VTAM access method control block
- Enables the IMS VTAM logon exit

Any logon requests received by VTAM before the IMS /START DC command is issued, but after the IMS VTAM access method control block (ACB) has been opened, are queued in VTAM until the /START DC command processing is completed. If VTAM is active when IMS is initialized, the IMS VTAM ACB is opened; otherwise, the ACB is opened by the /START DC command.

---

### Session Initiation (Starting Workstations)

**Definition:** A *session* is the logical connection of a workstation to a VTAM application program, such as IMS or the system utilities. A session must be established before data can be transmitted between a workstation and IMS. Message resynchronization is performed during session initiation, unless IMS is cold started.

## Session Initiation (Starting Workstations)

Session initiation can be requested in one of the following ways:

- The workstation requests session initiation by sending the INITIATE-SELF command. VTAM verifies the command and passes the request to IMS. If the terminal is requesting a session with an IMS XRF complex, the IMS USERVAR name should be used for the application name.
- The MVS VTAM network operator requests session initiation on behalf of the workstation by using the VTAM VARY command with the LOGON option. VTAM processes the request and passes it to IMS.
- VTAM passes a logon request to IMS for each workstation that is defined to VTAM as belonging to IMS.
- The IMS master terminal operator requests session initiation for a workstation by entering the IMS /OPNDST command.

Regardless of how session initiation is requested, identical processing occurs when IMS receives the request.

## Session-Initiation Transmission Sequence

Figure 65 shows the sequence of transmissions that occurs when a workstation requests session initiation. The numbers relate to major events in the sequence. If another method of requesting session initiation is used, the sequence begins with step **2A**.

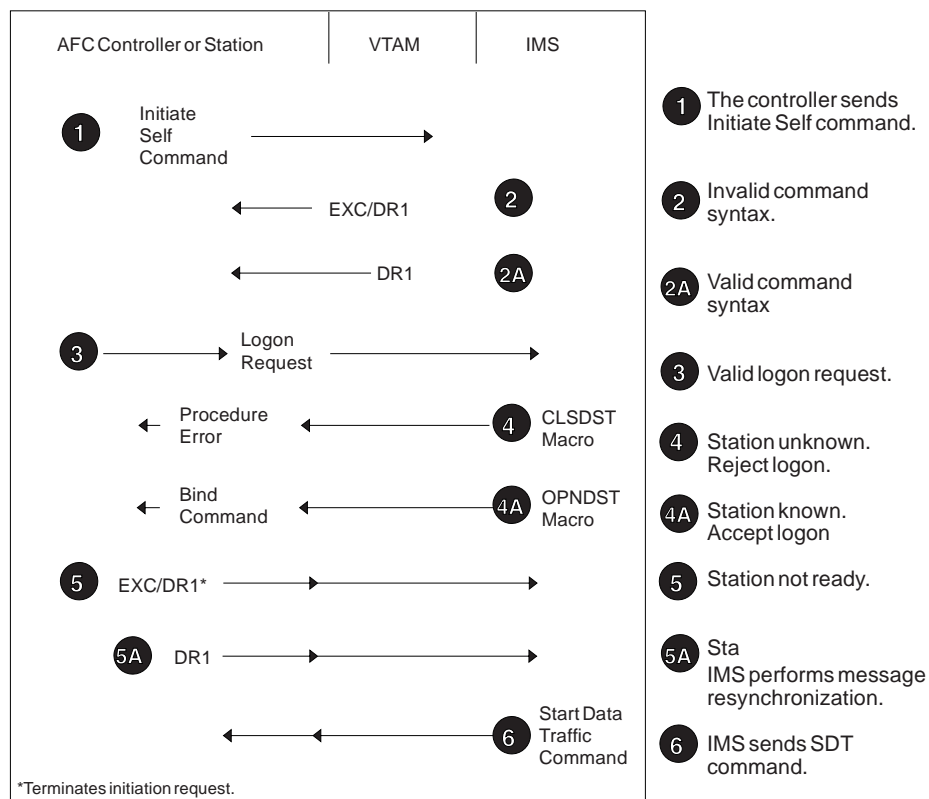


Figure 65. Session-Initiation Procedure

- 1** The controller sends the INITIATE-SELF command. VTAM checks the validity of the command, looking for syntax errors that might have been introduced by either the controller or NCP.



## Session Initiation (Starting Workstations)

**2** If the command is not valid or if VTAM does not find a match when verifying the initiation request, VTAM returns an EXC/DR1 response. EXC/DR1 terminates the initiation request. The SLU P system can retry the initiation request or notify the operator to begin corrective action.

**2A** If the command is valid, VTAM returns a DR1 response. VTAM then verifies the content of the resource field in the INITIATE-SELF command. This field must contain the name of the VTAM application program with which the workstation wants to enter a session. VTAM compares the resource field content to its list of active application programs (those programs with an open VTAM ACB).

**3** If a match is found, VTAM passes the logon request to the specified application program (IMS).

In an XRF environment, VTAM routes the request to the currently active system. When IMS receives the request, it compares the workstation node name supplied by VTAM to the node names defined during IMS system definition.

**4** If IMS does not recognize the workstation, it issues a VTAM CLSDST command, which causes VTAM to send a procedure error command. A procedure error terminates the session initiation request. The system notifies the workstation operator that the session is denied.

**4A** If IMS recognizes the workstation, it issues a VTAM OPNDST command, which causes VTAM to send a BIND command. The BIND command contains the name of the application program (IMS) that issued the OPNDST command.

In an XRF complex, the BIND command from VTAM contains the USERVAR segment in the user data field and the active APPLID in the PLUNAME field.

IMS supplies a set of BIND parameters, which define the communication rules and protocol that must be followed. IMS ignores the parameters of the mode table entry supplied by the workstation or network operator.

The workstation must respond to the BIND command.

**Related Reading:** For more information on the contents of the BIND data, see “Finance Communication System Bind Parameters” on page 471.

**5** If the workstation cannot begin a session, it returns an EXC/DR1. This can occur when either VTAM or IMS is requesting session initiation and the workstation is currently unable to communicate (for example, because it is involved in offline processing).

**5A** If the workstation can begin a session, it returns a DR1. When IMS receives the DR1, it performs message resynchronization, if necessary.

**6** If message resynchronization is not necessary, or when it is complete, IMS sends a VTAM start-data-traffic (SDT) command.

The workstation is in session and can transmit data to IMS.

## Controller Application Program Involvement in Message Resynchronization

The controller application program must participate in message resynchronization and, in order to do so, must have maintained copies of the sequence numbers of the last recoverable message that was successfully sent to IMS and the last recoverable message that was successfully received by the workstation.

## Design Considerations

The system application analyst must determine where to maintain the sequence numbers necessary for set-and-test-sequence-numbers (STSN) processing. Three options are available:

## Session Initiation (Starting Workstations)

- The controller disk or diskette.

The disk or diskette in the controller is the most reliable method and does not require involvement of the workstation operator if retransmission of a message is required. Either the permanent file or the transient files of the disk or diskette can be used. The REPLACE instruction should be used when writing to the disk or diskette file. REPLACE causes the data to be physically written to the disk or diskette and not just buffered in the controller's control storage. Disk or diskette storage assures sequence number retention across a power failure or a controller failure that can destroy the contents of controller's control storage. However, disk or diskette access and data transfer time can add significant overhead to each recoverable transaction.
- The controller's control storage.

If the controller's control storage is used to retain sequence numbers, a power loss or controller failure would destroy the contents of the controller's storage, making message resynchronization impossible. By not performing message resynchronization, recoverable input and output messages can be lost or duplicated.
- A workstation output device.

Message sequence numbers can be stored on a workstation output component such as the display or the journal printer. However, this method requires workstation-operator intervention to retrieve the numbers during STSN processing.

## Sequence Number Management

An understanding of how sequence numbers are handled during normal message transmission is prerequisite to understanding how and why the SLU P system performs the above functions.

The management of sequence numbers is shared between the controller, VTAM, and the VTAM application program (IMS). The controller assigns sequence numbers to messages originated by its workstations. VTAM assigns VTAM sequence numbers to messages originated by IMS. To ensure recoverability, IMS maintains a separate copy of the sequence numbers associated with the messages it sends.

Sequence numbers are assigned to each workstation. For each workstation, the controller tracks the last sequence number that it assigns (called the *last-assigned value*), and tracks the last sequence number that it receives (called the *last-received value*). Similarly, IMS tracks the last sequence number that VTAM assigns (called the *last-assigned value*), and tracks the last sequence number it receives (called the *last-received value*).

When the controller issues a request to transmit data, it updates its last assigned value for the requesting workstation, appends the new number to the data, and sends the message. When VTAM receives the message, it merely passes it on to IMS. IMS, in turn, adds 1 to its last received value for that workstation and compares this value with the sequence number of the message it just received. If the two numbers match, IMS processes the message as required. If the two numbers do not match, IMS issues a VTAM CLSDST macro instruction to terminate the session.

When IMS has a message to transmit, it updates its copy of VTAM's last assigned value and sends the message to VTAM. VTAM updates its last assigned value, appends the new number to the message, and sends the message. The controller

## Session Initiation (Starting Workstations)

removes the appended sequence number, updates its last received value, and compares this value with the sequence number that accompanied the message. If the two numbers match, the controller sends the message on to the application program. If the two numbers do not match, the controller returns an exception DR2 to indicate a sequence error. When IMS receives the sequence error indication, it issues a VTAM CLSDST macro instruction to terminate the session.

### Set-and-Test-Sequence-Numbers (STSN)

Message resynchronization is initiated by IMS when it sends the set-and-test-sequence-numbers (STSN) command to the workstation. The STSN command contains a 5-byte data field:

<b>Byte 0</b>	Action code
<b>Bytes 1, 2</b>	Controller sequence number of the last inbound sync-point message IMS received from the workstation
<b>Bytes 3, 4</b>	VTAM sequence number of the last outbound sync-point message IMS sent to the workstation

IMS uses the action code to ask the SLU P system to verify the controller and VTAM sequence numbers. The bits of the action code byte are:

<b>Bits 0, 1</b>	Refer to the controller sequence-number field
<b>Bits 2, 3</b>	Refer to the VTAM sequence-number field
<b>Bits 4, 5, 6, and 7</b>	Reserved

The following values are acceptable for bits 0, 1, 2, and 3 of the action code:

<b>00 IGNORE</b>	Not used by IMS.
<b>01 SET</b>	Set the appropriate sequence number to the value indicated in the sequence-number field.
<b>10 INVALID</b>	Not used by IMS. The SLU P system must return its version of the sequence number in the command response.
<b>11 SET AND TEST</b>	Set the appropriate sequence number to the value indicated in the sequence-number field. The SLU P system must indicate in the command response whether the sequence number values are acceptable.

IMS uses the SET option for the controller sequence number. For the VTAM sequence number, IMS uses either SET or SET AND TEST. SET is used when no acknowledgment to a recoverable output message is outstanding from the previous session. SET is also used when the IMS master terminal operator enters the /DEQUEUE or /ASSIGN commands that cause the last recoverable message to be removed from the queue. SET AND TEST is used if IMS sends a recoverable message to the workstation but does not receive the required acknowledgment prior to session termination. The response sent by the SLU P system to the STSN command indicates whether the workstation received the message.

When the SLU P system receives the STSN command, it must be able to:

## Session Initiation (Starting Workstations)

- Verify the controller sequence number and arrange to retransmit a message to IMS if required.
- Verify the VTAM sequence number and inform IMS whether the number is acceptable.
- Return a DR1 to IMS and, if required, return a 5-byte data response to the STSN command.

To verify the controller sequence number, the SLU P system compares the number provided by IMS with the number it maintained from the previous session. An equal compare indicates that IMS received all messages sent by the workstation. If the IMS-provided number is smaller, IMS did not receive the last inbound sync-point message sent by the workstation. These two numbers can differ by more than 1 if intervening irrecoverable or chained messages were sent by the workstation when the workstation was defined with the ACK option. A message that was not received by IMS should be retransmitted after message resynchronization is complete. If a copy of the message was maintained, that copy could be sent. Otherwise, the SLU P system can inform the workstation operator that the last inbound recoverable message was not received by IMS.

To verify the VTAM sequence number, the SLU P system compares the number provided by IMS with the number it maintained from the previous session. An equal compare indicates that the workstation has received all messages. If the IMS-provided number is unequal, the workstation did not receive the last outbound recoverable message.

The response to the STSN command indicates the results of the synchronization tests to IMS. The SLU P system must return a DR1 and, optionally, 5 bytes of data. If both the controller and VTAM sequence numbers are acceptable, only the DR1 is required. If one or both of the sequence numbers are not acceptable, a DR1 and the 5-byte data response are required. The format of the data response has the same format as the STSN command:

<b>Byte 0</b>	Action code
<b>Bytes 1, 2</b>	Controller sequence number of the last inbound recoverable message the workstation sent to IMS
<b>Bytes 2, 3</b>	VTAM sequence number of the last outbound recoverable message the workstation received from IMS

The SLU P system uses the action code to indicate the test results. The bits of the action code byte are:

<b>Bits 0, 1</b>	Refer to the controller sequence-number field
<b>Bits 2, 3</b>	Refer to the VTAM sequence-number field
<b>Bits 4, 5, 6, and 7</b>	Reserved

The following values are acceptable for bits 0, 1, 2, and 3 of the action code:

**00 RESET** The sequence number in the command is unacceptable and should be reestablished at its previous value. This code should never be returned to IMS.

**01 TEST POSITIVE** The sequence number in the command is acceptable. This code

## Session Initiation (Starting Workstations)

must be returned in response to the SET option. If the SET AND TEST option is specified and the SLU P system finds the sequence number acceptable, TEST POSITIVE should be returned.

**10 INVALID** The SLU P system has detected an error in the sequence number.

### 11 TEST NEGATIVE

The SLU P system does not agree with the sequence number presented with the SET AND TEST option.

TEST POSITIVE is the only acceptable response to the SET option. TEST POSITIVE and TEST NEGATIVE are acceptable responses to the SET AND TEST option. INVALID should be used if the SLU P system detects a “should-not-occur” condition when comparing the sequence numbers. An example of this condition is a controller sequence number specified in the command that is higher than the application program’s copy of the sequence number.

If IMS receives a TEST NEGATIVE response to the SET option, a RESET response, or an INVALID response, it terminates the session.

When returning any action code except TEST POSITIVE, the SLU P system should use the STSN response to return its version of the sequence numbers. The sequence numbers can be valuable information during problem determination and debugging. Any value can be returned, because IMS does not use the returned values. A TEST NEGATIVE response is all that is required to indicate that the sequence number is not acceptable.

When IMS receives the STSN response, it sends the start-data-traffic (SDT) command to the workstation. The SDT command allows normal message transmission to begin. If IMS receives a TEST NEGATIVE response to the SET AND TEST option, it sends the SDT command and retransmits the last recoverable message followed by any other available output. When IMS receives a TEST POSITIVE response to the SET AND TEST option, it sends the SDT command, dequeues the last recoverable message, and sends any other available output.

The contents of the STSN command as received from IMS and the resulting actions of the SLU P system and VTAM are summarized in Table 42.

Table 42. Set-and-Test-Sequence-Numbers (STSN) Summary

Action Code		Controller Sequence Number <sup>1</sup>	VTAM Sequence Number <sup>1</sup>	Remote System Action when STSN Received	VTAM Action when STSN Received
Remote System Number	VTAM Number	Last inbound sync point message that IMS received		Set controller sequence number field to value indicated in STSN controller field.	Set VTAM value to value indicated in STSN controller field.
01 set					

## Session Initiation (Starting Workstations)

Table 42. Set-and-Test-Sequence-Numbers (STSN) Summary (continued)

Action Code		Controller Sequence Number <sup>1</sup>	VTAM Sequence Number <sup>1</sup>	Remote System Action when STSN Received	VTAM Action when STSN Received
	01 set		Last outbound sync point message that IMS sends, and responded to by station.	Set host sequence number field to value indicated in VTAM field of STSN.	Set VTAM's last assigned value to value in VTAM field of STSN.
	11 set and test		Last recoverable message that IMS sends; no response from station received.		

**Note:**

1. The controller and VTAM sequence number fields can contain any value.

The valid SLU P system responses to the STSN command and the resulting IMS actions are summarized in Table 43.

Table 43. Set-and-Test-Sequence-Numbers (STSN) Response Summary

If Action Code Value Was:		Response Action Code Must Be: <sup>1</sup>		IMS Action when Response Received:
Remote System Number	VTAM Number	Remote System Number	VTAM Number	
01 set		01 test positive		Send SDT to station
	01 set		01 test positive	
	11 set and test		01 test positive	Dequeue last recoverable message and send SDT to station
			11 test negative	Send SDT to station and retransmit last recoverable message.

**Note:**

1. Any other response causes IMS to terminate the session.

## Suspending Output from IMS

If the controller application program does not want or cannot receive any more output from IMS, the program can send the VTAM quiesce-at-end-of-chain (QEC) command to IMS. IMS returns a DR1 and the VTAM quiesce-complete (QC) command. IMS does not send any more output to the workstation until it receives the VTAM release-quiesce (RELQ) command.

A workstation can also suspend output from IMS by sending the VTAM LUSTATUS or SIGNAL commands to IMS.

## Session Termination

**Definition:** *Session termination* releases the workstation from its current logical connection to the VTAM application program. It makes the workstation available for session with other VTAM applications, or communications can be terminated altogether.

Two types of session termination are orderly and immediate.

**Definitions:**

- In an *orderly termination*, the workstation is allowed to complete normal processing before the session is terminated.
- An *immediate termination* forces the workstation to terminate the session unconditionally.

Session termination can be invoked by any of the following:

- The IMS master terminal operator
- The VTAM network operator
- The workstation

Figure 66 summarizes the two types of session termination processing.

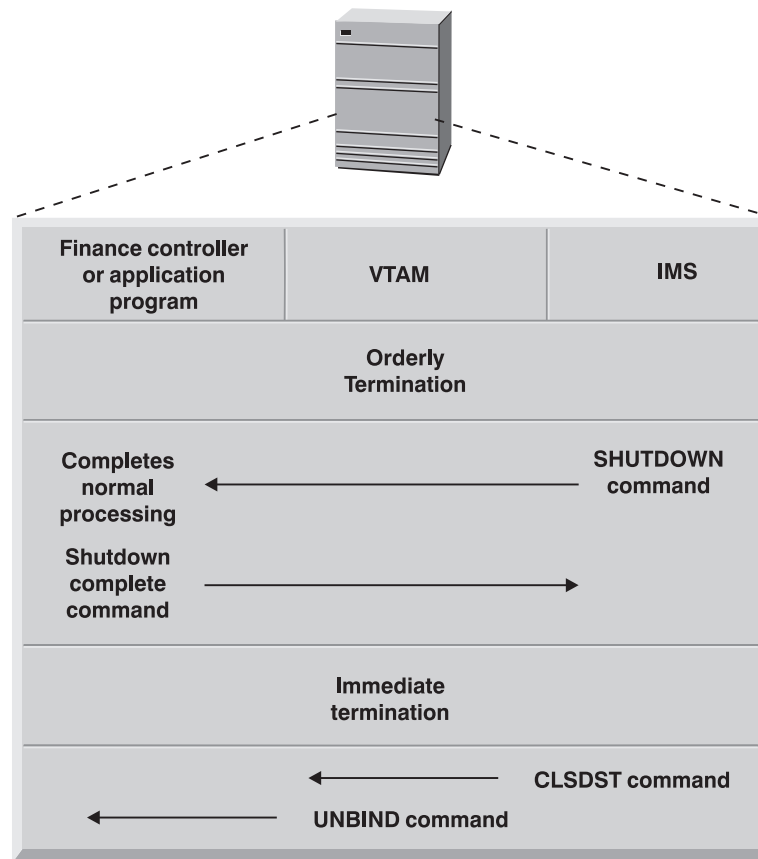


Figure 66. Termination Processing

Each installation must determine specific procedures for session termination. When developing the procedures, be aware of the requirements for session termination processing.

## Session Termination

### Orderly Termination

An orderly termination of the network is invoked by the IMS master terminal operator using the IMS /CHECKPOINT FREEZE, /CHECKPOINT PURGE, or /CHECKPOINT DUMPQ command with the QUIESCE parameter. QUIESCE is a /CHECKPOINT parameter that initiates shutdown processing for a network. When QUIESCE is specified, IMS sends the VTAM shutdown (SHUTD) command to all workstations and waits until all workstations have completed normal processing and returned the VTAM shutdown-complete (SHUTC) command.

When all workstations have indicated that shutdown is complete, IMS performs checkpoint processing and then issues the VTAM CLSDST macro instruction. CLSDST causes VTAM to send the UNBIND command to all workstations. This command releases the workstations from session with IMS. The controller prohibits any further data transmission from the workstations.

During shutdown processing (the time between the VTAM shutdown-complete and CLEAR commands), the system's workstations should be prepared for any IMS output that results from shutdown processing.

During the processing of an orderly termination, the IMS master terminal operator can terminate the network unconditionally rather than wait for the orderly termination processing to complete. This can be done by invoking an immediate termination.

### Immediate Termination

The IMS master terminal operator, the VTAM network operator, or the workstation user can invoke an immediate termination of a SLU P workstation.

The IMS master terminal operator uses the /CHECKPOINT command with the FREEZE, PURGE, or DUMPQ parameter, but without the QUIESCE parameter, to invoke immediate termination of the network. IMS issues the VTAM CLSDST macro instruction. CLSDST causes VTAM to issue the UNBIND command to all workstations. This command releases the workstation from session with IMS. The controller prohibits any further data transmission from the workstation.

To terminate workstations or portions of a network selectively, the IMS master terminal operator can use the IMS /CLSDST or /STOP command. /CLSDST and /STOP commands cause IMS to issue the VTAM CLSDST macro instruction for the specified workstations. The /STOP command also prevents further sessions from being established until a /START command is issued for the workstation.

The MVS VTAM network operator uses the VARY command to terminate a workstation immediately. MVS VTAM network operator intervention using the VARY command might be required to terminate sessions in which a workstation error prevents an I/O operation from completing.

The workstation uses VTAM session control commands to terminate a session with IMS. Under normal processing circumstances, when the workstation decides to terminate its session with IMS, that workstation should send the VTAM request-shutdown command. IMS completes any input or output currently in progress for that workstation and then issues the VTAM CLSDST macro instruction.



If the workstation detects an error condition from which it cannot recover and then wants to terminate the session, it can send the VTAM terminate-session command. VTAM releases the workstation from the session and notifies IMS accordingly.

---

## Shutting Down an IMS Network

Shutting down an IMS network may or may not include IMS.

The IMS /CHECKPOINT command is used to invoke termination of the network and a shutdown of IMS. The format of /CHECKPOINT used determines whether the network termination occurs immediately or waits for workstation processing to complete:

**/CHECKPOINT FREEZE|DUMPQ|PURGE QUIESCE**

Allows all workstations to complete normal processing before shutting down IMS.

**/CHECKPOINT FREEZE|DUMPQ|PURGE**

Causes immediate session termination for all workstations.

**Related Reading:** See “Immediate Termination” on page 440 if the network, or a portion of a network, requires shutdown without shutting down IMS.

---

## SLU P Messages

The primary function of IMS support for the SLU P system is to receive and transmit data for a workstation and to ensure that the data is processed properly. Input/output messages can consist of the following data types:

- IMS transactions
- IMS message switches
- IMS commands
- VTAM commands and indicators
- Message Format Service (MFS) control requests

IMS can process nongraphic characters when sending and receiving any of these messages.

**Related Reading:** For more information on how IMS processes nongraphic message data, see “IMS Sensitivity to Nongraphic Message Data” in *IMS/ESA Administration Guide: System* .

---

## Send/Receive and Bracket Protocol

The change-direction indicator and bracket protocol, as defined by VTAM, are used to specify the beginning and end of synchronous transmissions and to control the flow of such transmissions. Because of its queued input and output processing, IMS does not support multiple related inputs or multiple related outputs. While in an in-brackets state, the following can occur:

- One input message (coded BB/EB)
- One output message (coded BB/EB)
- One input message and an output reply (input coded BB/CD and output coded EB)

The input message and output reply are related only for:

- Fast Path transactions

## Send/Receive and Bracket Protocol

- Terminal-response mode transactions
- Conversational transactions

For other transaction types, the input message and output reply might or might not be related.

**Related Reading:** For more information on how IMS sends and receives the bracket and change-direction indicators, see “Input Bracketing Protocol” on page 446 and “Output Bracketing Protocol” on page 453.

---

## Chapter 22. SLU P Message Protocols

### In this Chapter:

- “General Format of Input Function Management Headers (Finance)”
- “Input Message Descriptor Byte (Finance)” on page 444
- “General Format of Input Function Management Headers (SLU P)” on page 444
- “Output Messages” on page 447
- “MFS Distributed Presentation Management Output (SLU P)” on page 449
- “General Format of Output Function Management Headers (Finance)” on page 450
- “General Format of Output Function Management Headers (SLU P)” on page 451
- “Input Response Requirements” on page 455
- “Output Response Requirements” on page 456
- “IMS Transaction Types” on page 456
- “Error Handling” on page 459

A single transmission must be used for VTAM commands and indicators and MFS control requests. Single or multiple transmissions can be used to send IMS transactions, commands, and message switches. Multiple transmissions for one message are required if the workstation’s transmission buffer is not large enough to hold all of the data to be sent. Multiple transmissions can also be used if the workstation user wants to segment the data. Multiple transmissions are logically related through the concept of chained messages. Each transmission of a multiple transmission message is identified by its position in the chain—that is, first-in-chain, middle-in-chain, last-in-chain (see Figure 67 on page 449).

In a SLU P system using chained input messages, the chaining indicator is used to specify the chain position of the transmission; it should not be set for an unchained message. The chaining indicator must be turned on for the first-in-chain transmission, turned off for each middle-in-chain transmission, and turned on again for the last-in-chain transmission.

When MFS is defined, input messages can be processed by the MFS. For input formatting, MFS does not distinguish between possible input components but assumes that all input comes from the 4701/4702 application program.

---

### General Format of Input Function Management Headers (Finance)

IMS transactions, message switches, and commands are in the standard IMS format (transaction code, logical terminal name, or command verb, followed by a blank and text) unless MFS or a Physical Terminal Input edit routine is used to edit the message into the standard format. The first or only transmission of IMS transactions can have a variable-length FM header. If MFS is not defined for the workstation, the header is ignored. If MFS is defined, the header can contain either an MFS control request or a MID name. For MFS control requests, only the first 2 bytes of the FM header are required.

- |               |  |
|---------------|--|
| <b>Byte 0</b> | Header length including the length byte (binary). Bytes 0 and 1 constitute an MFS control request. |
| <b>Byte 1</b> | Message description (binary).  |
| <b>Byte 2</b> | MID name length (binary).  |

## General Format of Input Function Management Headers (Finance)

**Bytes 3-10** MID name (1 to 8 bytes) (EBCDIC).

IMS uses the header, but removes it before sending the message to the MPP or MFS. When IMS receives a message with a header, it interrogates the length of the message. If the message length is 2 bytes, IMS assumes that the message is an MFS control request. The MFS control request is indicated in the message description (byte 1).

The PAGEREQ function is not an MFS control request for paging requests.

**Related Reading:** For more information on using the PAGEREQ function, see *IMS/ESA Application Programming: Transaction Manager*.

---

## Input Message Descriptor Byte (Finance)

If the message length is more than 2 bytes, IMS interrogates the message description to determine whether an MFS MID name is present. The format description specified by the MID name is used to format the message.

The format of the message descriptor byte is:

<b>Bit 0</b>	Reserved
<b>Bit 1</b>	Page advance requested (NEXTTPP)
<b>Bit 2</b>	Message advance requested (NEXTMSG)
<b>Bit 3</b>	Message advance protect requested (NEXTMSGP)
<b>Bit 4</b>	Next logical page requested (NEXTLP)
<b>Bit 5</b>	Reserved
<b>Bit 6</b>	Format description that is indicated by the MID name length and MID name fields should be used to format this input message.
<b>Bit 7</b>	Reserved

Bits 1, 2, 3, and 4 (MFS control requests) are mutually exclusive and are examined only if the message length is 2 bytes. Bit 6 must be set on if the input message is to be formatted by an MFS MID name. If bit 6 indicates an MFS MID name is present, bits 0 through 5 must be zero.

---

## General Format of Input Function Management Headers (SLU P)

IMS transactions, message switches, and commands are in the standard IMS format (transaction code, logical terminal name, or command verb, followed by a blank and text). If a message is in nonstandard format, MFS or a Physical Terminal Input edit routine can be used to edit the message into the standard format. The first or only transmission of IMS transactions can have a variable-length FM header. If MFS is not defined for the terminal, the header is ignored. If MFS is defined, the header can contain either an MFS control request or optional MFS fields used to invoke MFS formatting. IMS uses the header in these cases, but removes it before passing the message on to the MPP or MFS. Incorrect header specifications or formats can cause the session to terminate. SLU P uses header type X'42'.

The format of the input FM header is:

**Byte 0** Header length including the length byte in binary (first byte)

## General Format of Input Function Management Headers (SLU P)

<b>Byte 1</b>	Header type (must be X'42')
<b>Byte 2</b>	Message descriptor 1 (flag byte is binary)
<b>Byte 3</b>	Message descriptor 2 (flag byte is binary)
<b>Byte 4</b>	Input component identification (binary)

Optional MFS fields:

### For DPM:

**Bytes 5 and 6** Version ID (in binary) if bit 0 of byte 3 is on.

**Bytes 7-15** MID name length, including length byte (1 byte), followed by the MID name (1 to 8 bytes). If bit 0 of byte 3 is off (version ID not included in the FM header), the MID name length and the MID name are in bytes 5 through 13.

### For SCS:

**Bytes 3-13** MID name length (in binary), including length byte (1 byte), followed by the MID name (1 to 8 bytes).

**Related Reading:** For more information on the Version ID field, see *IMS/ESA Application Programming: Transaction Manager*.

When IMS receives a message with a header, it determines the length of the message. If the message length is 5 bytes, IMS assumes that the message is an MFS control request. The specific MFS control request is indicated in message descriptor 1 (byte 2). If the message length is more than 5 bytes, IMS examines the message description bytes to determine whether an MFS version ID and a MID name are present. When both are present, the version ID must immediately precede the MID name length. (Version ID does not exist for SCS1.) The format description specified by the MID name is used to format the accompanying input message, and the version ID is used to validate the format description level. If no FM header is provided, no version ID is provided, or a version ID of zero is provided, MFS bypasses the validity check on the format description level.

## Input Message Descriptor Bytes (SLU P)

Message descriptor 1 (byte 2 of the FM header) has the following format:

<b>Bit 0</b>	Reserved
<b>Bit 1</b>	Page advance requested (NEXTTPP)
<b>Bit 2</b>	Message advance requested (NEXTMSG)
<b>Bit 3</b>	Message advance protect requested (NEXTMSGP)
<b>Bit 4</b>	Next logical page requested (NEXTLP)
<b>Bit 5</b>	Reserved
<b>Bit 6</b>	Format description indicated by the MID name length, and MID name fields should be used to format this input message
<b>Bit 7</b>	Reserved

Bits 1, 2, 3, and 4 (MFS control requests) are mutually exclusive. Bit 6 must be set on if the input message is to be formatted by the MFS MID name provided. When bit 6 indicates a MID name is present, bits 0 through 5 must be zero.

## General Format of Input Function Management Headers (SLU P)

Message descriptor 2 (byte 3 of the FM header) has the following format:

- |                |  |
|----------------|--|
| <b>Bit 0</b>   | Version identifier for MFS DPM. Bit 0 must be set on if bytes 5 and 6 of the FM header contain the version ID. |
| <b>Bit 1-7</b> | Reserved   |

## Input Component Identification (SLU P)

The input component identification is used for the input component selected, as previously described in “Component Definition” on page 411. The component identification can be a value between 0 and 4, matching the ICOMP specified on the NAME macro during IMS system definition or on an ETO user descriptor.

**Related Reading:** For more information on the ETO feature, see “Chapter 7. Administering the Extended Terminal Option” on page 123.

If a component identification is zero, or if no FM header is sent, the input is associated with component 1. Further, IMS uses the input component identification to reset component output protection. If a value of 0 (zero) is sent, output protection is reset on all components of the terminal. If a value 1 through 4 is sent, output protection is reset on only the specified component.

## Input Bracketing Protocol

A workstation in session with IMS must indicate begin-bracket (BB) or both begin-bracket and end-bracket (EB) on each of following:

- The first transmission of a chained message (multiple transmission)
- Each transmission of unchained messages

The end-bracket indicator places both IMS and the workstation between brackets and in a contention state. If only the begin-bracket indicator is sent, the workstation should send a change-direction (CD) indicator on each of the following:

- The last transmission of a chained message (multiple transmission)
- Each transmission of unchained messages

The CD indicator is necessary, because IMS cannot correlate multiple input messages. The CD indicator is implied if not specified as described above.

If BB and CD are indicated, IMS indicates EB on its next output to the workstation. If IMS detects an error while receiving chained input on which only BB is specified, it returns an EB and an IMS error message.

**Related Reading:**

- For more information on output messages, see “Output Messages” on page 447.
- For more information on error handling, see “Error Handling” on page 459.

**Restriction:** For SLU P terminals, a CD or EB indicator must accompany each input chain; otherwise, the session terminates.

## Activating MFS Input Formatting for Finance Workstations

When MFS is used, input messages can be processed by the message and format descriptors. When IMS receives an input message, IMS Basic Edit is performed unless a MID name accompanies the message. The MID name can be supplied by including it either in the input function management header or in the beginning of

## General Format of Input Function Management Headers (SLU P)

the message text with the required MFS escape characters (/). When the MID name is present, MFS edits the message using the specified MID and its associated device input format (DIF). The MID name is provided by either the operator or the remote application program.

---

## Output Messages

Output messages from IMS can be one of the following types:

- Data replies to recoverable or irrecoverable input transactions
- Data replies to IMS commands
- Message switches
- VTAM indicators
- IMS system messages
- Broadcast messages
- A null (length=0) message containing end-bracket to return the workstation to between-brackets and to a contention state

All messages from IMS to a workstation are sent in a single transmission unless:

- The MPP, a message switch, a command, or MFS provides a multisegment output message.
- The workstation's read buffer is too small to hold the single-segment output message provided by the MPP or MFS.
- A broadcast message is multisegment.

Each segment of a message is sent in a single transmission whenever possible, that is, when only-in-chain is indicated. However, if a message segment exceeds the size of the receiving workstation read buffer, the segment is divided into as many transmissions as required until the complete segment is sent. Multisegment output messages are handled like chained input messages: Each segment is identified appropriately as first-in-chain, middle-in-chain, or last-in-chain.

Exercise care when defining the sizes for IMS message queue data sets and workstation output buffers. Incorrect specification can result in multisegment output chains from IMS. Output buffer sizes that are too small can result in multiple transmissions for large IMS segments. Message queue data set sizes that are too small can result in multiple IMS segments being created for large message processing program (MPP) inserts.

**Related Reading:** For information on the MSGQUEUE and TERMINAL macros, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

**Definition:** *MFS-paged output* is each physical page of a message destined for a display device and sent as if it were a complete message.

Figure 67 shows the relationship between transmissions sent to IMS, segments produced by the MPP, and segments transmitted by IMS.

After successfully receiving a message, the SLU P system must interrogate the read-type field (SMSCRT), the read-flags field (SMSCRF), and, optionally, the read-flags field extension (SMSCRE) to determine the type and characteristics of the message received.

## Output Messages

**Related Reading:** For more information on these fields, see *IBM 4700 Finance Communication System: Controller Programming Library, Volume 3: Communication Program*.

If a non-MFS-formatted output segment is sent in multiple transmissions and the SCAN option is defined for that workstation, IMS does not allow a 4701/4702 device's select control, position-control, or write-control sequence to be split across transmissions. When transparent write-control sequences are split into two transmissions, the data length on the first transmission is modified, and the proper write control characters are inserted at the beginning of the second transmission.

Editing for device control sequences does not occur for non-MFS-formatted output if the NOSCAN option is specified during IMS system definition or on an ETO logon descriptor. If MFS formats the output segment, the device-control sequence (select and position) editing always occurs. Each transmission of the segment, except the last, ends in a three-character, null-horizontal-position sequence to suppress the automatic new-line function of the physical SLU P system output device.



## MFS Distributed Presentation Management Output (SLU P)

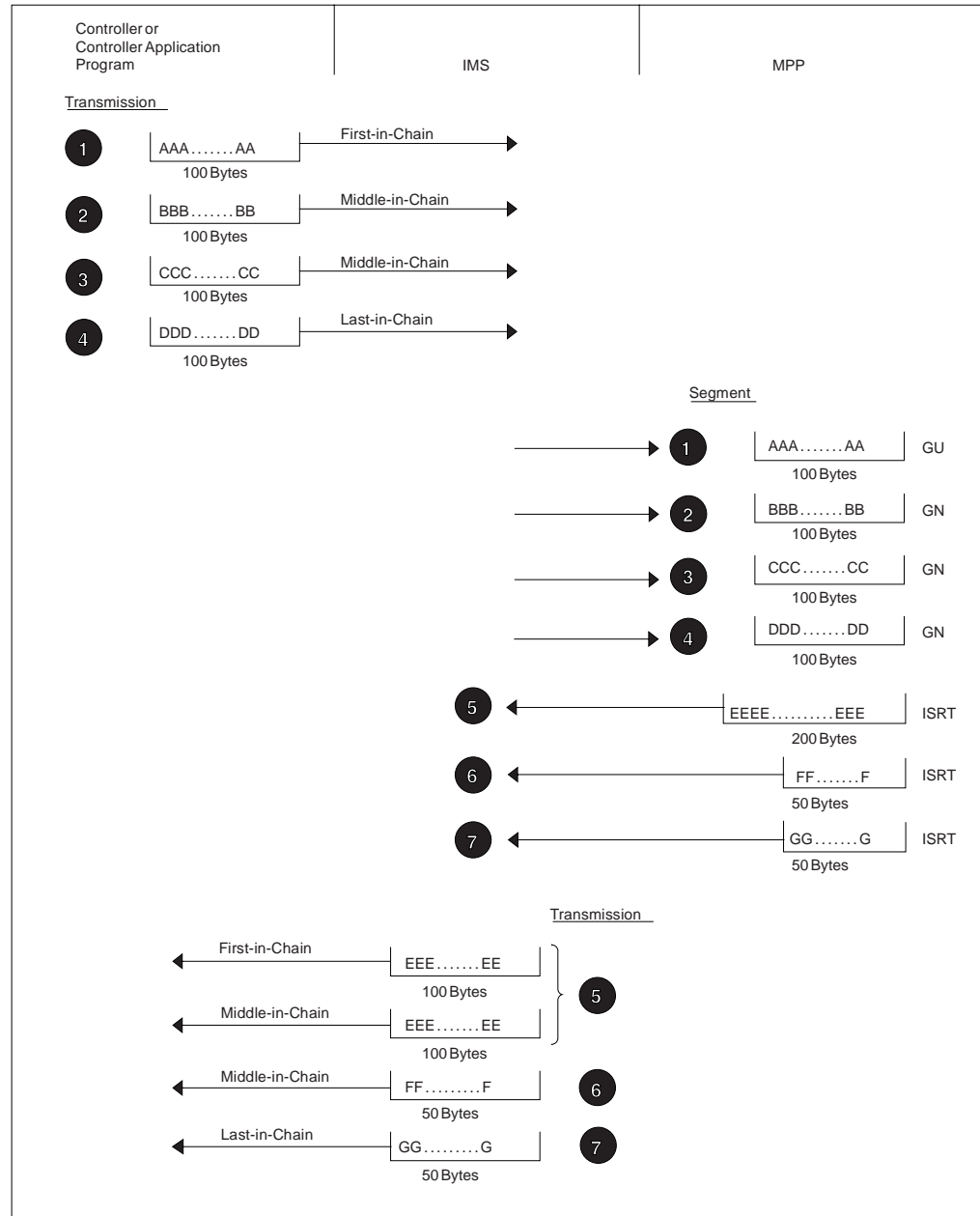


Figure 67. Chained Message Interaction between a Finance Communication System and an IMS MPP

## MFS Distributed Presentation Management Output (SLU P)

IMS sends the entire output message as a single chain of one or more related transmissions. For MFS DPM output using a device format defined with paging `OPTIONS=DPAGE` or `OPTIONS=PPAGE`, IMS sends each logical or presentation page in a single chain of one or more related transmissions as if it were a complete message.

For MFS DPM output using a device format defined with paging `OPTIONS=DPAGE` or `OPTIONS=PPAGE`, IMS sends an output function management header as part of the first or only transmission of each chain (logical or presentation page) of the message. The remainder of the transmission contains data fields (DFLDs) up to the

## MFS Distributed Presentation Management Output (SLU P)

defined record length. This function management header contains the DPM version ID, DPAGE or PPAGE name, and a MID name if specified.

When a forms literal is defined in the DPM format specification for paging `OPTIONS=DPAGE` or `OPTIONS=PPAGE` (`FORMS=` parameter on the `DEV` statement), IMS precedes the first logical or presentation page with an only-in-chain transmission consisting of a function management header containing only the forms literal. No version ID, DPAGE name, or PPAGE name is provided. Output components are protected at the end of each output chain; this prevents IMS from sending subsequent output, or logical or presentation pages (chains), until an appropriate input message or control request is received.

For MFS DPM output using a device format defined with paging `OPTIONS=MSG`, IMS sends the entire message as a single chain of one or more related transmissions. The function management header is sent as the first or only transmission and contains the DPM version ID, format name, and any MID name or forms literal defined in the MFS format specification. In this case, as for non-DPM output, component protection is provided only if the destination component is defined as `PROGRAM2`.

For DPM, the device type symbolic name is specified to MFS as `DPM-Xn`, where `X` can be `A` or `B`. Message formatting is specified on the `TERMINAL` macro or on an `ETO` logon descriptor for logical units defined as `SLU P` by using device type symbolic names of the form `DPM-An`. `DPM-Bn` device type symbolic names refer to logical units defined for Intersystem Communication.

---

## General Format of Output Function Management Headers (Finance)

IMS appends a header to the first or only-in-chain transmission of all messages sent, except the IMS null message and VTAM commands and indicators. This header describes the type of message that follows and, if appropriate, indicates the output component that is to receive the message. The header can contain optional MFS information.

The length of the output function management header varies from 3 to 29 bytes. The format is:

<b>Byte 0</b>	Header length including length byte (binary)
<b>Byte 1</b>	Message description (binary)
<b>Byte 2</b>	Output component identification (binary)
<b>Bytes 3-28</b>	MFS data (length byte: binary; names: EBCDIC)

## Output Message Descriptor Byte (Finance)

The bits of the message descriptor byte (byte 1), when set on, have the following meanings:

<b>Bit 0</b>	This message is an IMS system message or a broadcast message.
<b>Bit 1</b>	This message is formatted by MFS.
<b>Bit 2</b>	An MFS system control area (SCA) has requested device alarm for this message. (The controller application program can take any appropriate action, such as turning on a terminal's light.)
<b>Bit 3</b>	This message is a nonqueued message (no specific destination). Bit 0 is also set.

## General Format of Output Function Management Headers (Finance)

<b>Bit 4</b>	Reserved.
<b>Bit 5</b>	This message is Fast Path-recoverable output.
<b>Bit 6</b>	A MID name is present in the MFS data field.
<b>Bit 7</b>	A FORMS name is present in the MFS data field.

## Output Component ID Byte (Finance)

The output component identification (byte 2) contains a value from X'01' to X'04'. This value identifies the terminal component to which this message is directed. Output component identifiers are assigned during IMS system definition, on an ETO user descriptor, or by using the Signon (DFSSGNX0) and the Output Creation (DFSINSX0) exit routines. These identifiers are specified by the sender of the message (either IMS or the message processing program).

**Related Reading:** For more information on the output component identification, see "Output Component Selection" on page 412.

## MFS Data Bytes (Finance)

Bytes 3 through the end of the header can contain MFS data, and one or two fields can be present. Bits 6 and 7 of the message descriptor byte indicate whether these fields are included. The first field, if present, is 2 to 9 bytes long and contains a 1-byte length indicator including length byte and a 1- to 8-byte MFS MID name. This MID name field is present if the message output description contains the name of the MID to be used for the next input message, and should be returned to IMS by the controller program. The second field, if present, is 2 to 17 bytes long and contains a 1-byte length indicator including length byte and a 1- to 16-byte FORMS name. FORMS name identifies the special form required for this message. Before printing the message, the application program should ensure that the form is in place and that page size and forms alignment are established.

---

## General Format of Output Function Management Headers (SLU P)

IMS appends a header to the first or only-in-chain transmission of all messages sent, except the IMS null message and VTAM commands and indicators. This header describes the type of message that follows and, if appropriate, indicates the output component that is to receive the message. The header can also contain optional MFS information. SLU P uses header type X'42'.

The length of the output function management header varies from 5 to 42 bytes. The format is:

<b>Byte 0</b>	Message header length including length byte (binary)
<b>Byte 1</b>	Header type (X'42')
<b>Byte 2</b>	Message descriptor 1 (flag byte is binary)
<b>Byte 3</b>	Message descriptor 2 (flag byte is binary)
<b>Byte 4</b>	Output component identification (binary)
<b>Bytes 5-41</b>	MFS data (length byte: binary; names: EBCDIC)

**Related Reading:** For information on MFS data, see "MFS Data Bytes (SLU P)" on page 452.

## General Format of Output Function Management Headers (SLU P)

### Output Message Descriptor Bytes (SLU P)

The bits of the message descriptor 1 (byte 2) have the following meaning when set on:

<b>Bit 0</b>	This message is an IMS system message or a broadcast message.
<b>Bit 1</b>	This message is formatted by MFS.
<b>Bit 2</b>	Reserved.
<b>Bit 3</b>	This message is a nonqueued message (no specific destination). Bit 0 is also set.
<b>Bit 4</b>	Reserved.
<b>Bit 5</b>	Fast Path-recoverable output.
<b>Bit 6</b>	A MID name is present in the MFS data field.
<b>Bit 7</b>	A forms name is present in the MFS data field.

The bits of message descriptor 2 (byte 3), when set on, have the following meaning:

<b>Bit 0</b>	This output is formatted by MFS DPM. The version ID might or might not be present, and one of the following is present in the MFS data field: <ul style="list-style-type: none"><li>• DPM format name</li><li>• Logical page name (DPAGE statement label)</li><li>• Presentation name (PPAGE statement label)</li></ul>
<b>Bit 1</b>	Reserved.
<b>Bit 2</b>	At the end of this chain, the component indicated in byte 4 of the output function management header is protected.
<b>Bits 3-7</b>	Reserved.

If message descriptor 2, bit 2, is set on, IMS does not send another output chain until an appropriate input occurs. This input can be an input message, an MFS control request, or a READY-TO-RECEIVE (RTR) command.

**Related Reading:** For more information on extended output component protection, see “Extended Output Component Protection (SLU P)” on page 422.

### MFS Data Bytes (SLU P)

Bytes 5 through the end of the header can contain up to four MFS fields, depending on settings in message descriptor 1, bits 6 and 7, and message descriptor 2, bit 0.

If bit 0 is on, the first field, if present, is a 2-byte MFS DPM version identification. This field is present for all MFS DPM formatted output. The field can contain a version ID, or the value 0 (zero). In either case, one of the following fields (second, third, or fourth field) is present.

The second field, if present, is 2 to 9 bytes long and contains a 1-byte length indicator, including length byte followed by a 1- to 8-byte MFS MID name. This MID name field is present if the message output description contains the name of the MID to be used for the next input message, and should be returned to IMS by the controller program.

## General Format of Output Function Management Headers (SLU P)

The third field, if present, is 2 to 9 bytes of data name, as defined by the paging option for the MFS device format.

The fourth field, if present, is 2 to 17 bytes long and contains a 1-byte length indicator, including length byte followed by a 1- to 16-byte user-specified MFS forms literal. The forms literal identifies the special setup or forms required for this message. The user must define the procedure to be followed at the terminal upon receipt of the forms literal.

The message format name (FMT statement label) is present if OPTIONS=MSG is specified; the logical page name (DPAGE statement label) is present if OPTIONS=DPAGE is specified; the presentation page name (PPAGE statement label) is present if OPTIONS=PPAGE is specified. This field appears in all output (message descriptor 2, bit 0) from DPM.

The content and format of the function management header for DPM formatted output can be influenced when defining the MFS device format through the HDRCTL=FIXED or HDRCTL=VARIABLE option. If HDRCTL=VARIABLE is specified, each MFS header field is variable in size. If HDRCTL=FIXED is specified, the following MFS header fields are sent and padded to their maximum size.

- The MID name field in the output header is padded with trailing blanks for the maximum length of 8 bytes. Eight blanks are sent if no MID name is specified to format the next input through the user-supplied message output description.
- The message format name is padded to a maximum length of 6 bytes. The logical page name or the presentation page name is padded to a maximum length of 8 bytes.

## Output Bracketing Protocol

For output messages, IMS specifies begin-bracket and end-bracket on single-segment output messages and on the first transmission of a multisegment message. If the input message specified begin-bracket and change-direction, the next output IMS sends specifies only end-bracket. If IMS detects an error when receiving chained input on which only begin-bracket is specified, it returns end-bracket and an IMS error message.

**Related Reading:** For more information on error handling, see “Error Handling” on page 459.

In the case of Fast Path, conversational, or response-mode transactions, the output message is the reply to the previous input message. In other cases, the output message might or might not be related to the previous input message.

IMS sends an only-in-chain output message (no FM header and data length=0), requesting exception DR2, and indicating FM header end-bracket when no other output is immediately available. This occurs in the following situations:

- An input message from a workstation defined to IMS as “negated terminal-response mode” specifies begin-bracket and change-direction.
- An input message from a workstation defined to IMS as “forced terminal-response mode” specifies begin-bracket and change-direction and is not an IMS transaction.
- An input message from a workstation defined to IMS as “transaction-dependent terminal-response mode” specifies begin-bracket and change-direction and is not a response-mode transaction.

## General Format of Output Function Management Headers (SLU P)

After receiving exception-response sense codes 0811 and 0812, IMS also sends an only-in-chain output message (no FM header and data length=0), requesting exception DR2, and indicating FM header begin-bracket/end-bracket when no other output is immediately available.

## Activating MFS Output Formatting

MFS output formatting occurs when an output message has an associated message output descriptor (MOD). This occurs when any of the following occurs:

- The MPP supplies the name of a MOD (MOD name) with the output message.
- The input message was processed by a message input descriptor (MID) whose definition specified a MOD name for output formatting.
- The output message is a message switch from a device using MFS editing.

If no MOD is associated with the output message, standard IMS output editing occurs.

Input formatting for a workstation only occurs when a MID name and the message itself are sent to IMS. Specifying the next MID name is allowed in the message output description. Controller involvement is required to assure the proper MID is used, because IMS cannot directly control the receipt of input. Therefore, while IMS is formatting an output message, the workstation can be sending an input message. Input sent while IMS is processing output is queued by VTAM until IMS completes output processing. If the next MID name is specified, the input queued by VTAM appears as if it were the result of the current output. Unpredictable results occur if the internal MID name is used to format the message. IMS avoids this by sending the MID name as part of the input message.

IMS informs the controller that the name of the next MID is specified, and sends the MID name requested and the output message to the workstation. The controller then saves the MID name, displays the output message, reads the next operator input, adds the saved MID name to the transaction, and sends the transaction to IMS.

## Response Requests (Finance)

Output messages from IMS require specific responses. IMS commands, broadcast output, message switches, and non-Fast Path transactions defined to IMS as update-inquiry or recoverable-inquiry require a DR2, except replies to all but the last physical or logical page of non-operator, logical-paged MFS output.

**Related Reading:** For information on response requirements, see “Input Response Requirements” on page 455.

Transactions defined to IMS as irrecoverable-inquiry or replies to operator logical-paged MFS output require an exception DR2. Except for the last physical page of a message whose MOD does not specify PAGE=YES, MFS-paged output requests an exception DR2 reply (regardless of how the transaction is defined), because an explicit MFS control or data input must follow each page before another page of output is allowed; otherwise, the message is dequeued.

Fast Path output messages are sent requesting an exception DR2 response regardless of the transaction type. The next input message from the workstation or an RTR command provides acknowledgment that the preceding output has been received and that IMS might dequeue the message.

## Response Requests (SLU P)

Except for replies to all but the last physical or logical page of non-operator, logical-paged MFS output, output messages from IMS require specific responses. IMS commands, broadcast output, message switches, and non-Fast Path transactions defined to IMS as update-inquiry or recoverable-inquiry require a DR2.

**Related Reading:** For information on response requirements, see “Input Response Requirements”.

Because an explicit MFS control or data input must follow each page before another page of output is allowed (or the message is dequeued), transactions defined to IMS as irrecoverable-inquiry and replies to operator logical-paged MFS output require an exception DR2. MFS-paged output, except the last physical page of a message whose MOD does not specify PAGE=YES, requests an exception DR2 reply regardless of how the transaction is defined.

---

## Input Response Requirements

Table 44 summarizes input requirements for requesting responses for all types of data transmission between a workstation and IMS.

Table 44. Input Response Requirements by Message Type

Data Type	Responses Accepted (ACK Option)	Responses Accepted (OPTACK Option)
Update transaction	DR2 <sup>1</sup> Exception DR1 or exception DR2 <sup>3</sup>	DR1 or DR2 Exception DR1 or exception DR2 <sup>1, 2</sup>
Recoverable-inquiry transaction	DR2 <sup>1</sup> Exception DR1 or exception DR2 <sup>3</sup>	DR1 or DR2 Exception DR1 or exception DR2 <sup>1, 2</sup>
Irrecoverable-inquiry transaction	DR1 or DR2 Exception DR1 or DR2 <sup>1</sup>	DR1 or DR2 Exception DR1 or exception DR2 <sup>1, 2</sup>
Fast Path transaction <sup>4</sup>	N/A	DR1 or DR2 Exception DR1 or exception DR2 <sup>1, 2</sup>
IMS message switch	DR2 <sup>1</sup>	DR1 or DR2 Exception DR1 or exception DR2 <sup>1, 2</sup>
IMS command	DR1 or DR2 Exception DR1 or DR2 <sup>1</sup>	DR1 or DR2 Exception DR1 or exception DR2 <sup>1, 5</sup>
VTAM command or indicator	DR1 <sup>1</sup>	DR1 <sup>1</sup>
SLU P system MFS control request	DR1 or DR2 Exception DR1 or DR2 <sup>1</sup>	DR1 or DR2 Exception DR1 or DR2 <sup>1</sup>

**Notes:**

1. Recommended or required response.
  2. Fast Path users should always use OPTACK option with exception DRx.
  3. With CD only for the commands /DIS, /RDIS, and /FOR.
  4. ETO users should always use OPTACK option.
  5. With Change Direction (CD) only.
-

---

### Output Response Requirements

Table 45 summarizes the response to be requested by IMS for all types of output messages.

*Table 45. Output Response Requested by Message Type*

<b>Output Data Type</b>	<b>Response Requested</b>
Update, Recovery	DR2
Inquiry, Recovery	DR2
Inquiry, Irrecovery	Exception DR1 and DR1 <sup>3</sup>
Fast Path (Recovery)	Exception DR2 <sup>1</sup> and DR2 <sup>2</sup>
Last MFS Page (when PAGE=YES is not specified on the MOD)	Refer to above data types for response request
Nonlast MFS Page (or all pages when PAGE=YES is specified on the MOD)	Exception DR2 and DR2 <sup>2</sup>
IMS Command Replies: /FORMAT, /DISPLAY, /RDISPLAY	DR2
Other Commands	Exception DR2 and DR2 <sup>2</sup>
Test Mode Output	Exception DR1 and DR1 <sup>3</sup>
Broadcast or Message Switch Output	DR2

**Notes:**

1. Next input message or RTR provides acknowledgment if exception DR2 is requested.
  2. DR2 is requested when one of the following occurs:
    - The message is sent with BB (bidding with data).
    - Messages on the IMS message queues are waiting to be sent to the workstation.
    - The NFPACK option is defined on the system definition TERMINAL macro or on an ETO logon descriptor.
  3. DR1 is requested when the message is sent with BB (bidding with data).
- 

---

### IMS Transaction Types

Transactions are the most common data type sent from a workstation to IMS. IMS supports two kinds of transactions—update and inquiry. An update transaction can modify a database. An inquiry transaction can look at data in a database but cannot change or update it. Transactions are defined as update or inquiry during IMS system definition.

An additional attribute is defined for inquiry transactions—recoverable or irrecoverable. Recoverable-inquiry transactions are always recoverable no matter which element in the network fails. Irrecoverable-inquiry transactions are not recovered following an I/O error condition or IMS system restart.

All update transactions are recoverable.

All Fast Path transactions must be defined as recoverable, but can be either inquiry or update.

A workstation can be defined to handle one or both types of transactions. Other decisions and processing might be required of the controller or controller application program when both recoverable and irrecoverable transactions are handled. The



decision to define a transaction as recoverable or irrecoverable requires an evaluation of the advantages and disadvantages that each transaction type offers to the individual operating environment.

The following sections describe the inquiry message types the responses required for each.

### Recoverable-Inquiry Transactions

To ensure that a recoverable transaction can be recovered, the workstation must:

- Request a DR1 or DR2 on input to IMS, or, if the OPTACK option is defined, optionally request an exception DR1 or exception DR2 and specify begin-bracket and change-direction on input to IMS.
- Maintain the input message sequence number, and, optionally, a copy of the input message until the DR1 or DR2 is returned, or, if the OPTACK option is specified, until a reply message containing an end-bracket is returned.
- Return any DR2 requested by IMS, or, if Fast Path, ensure that another input message or RTR command is sent after having accepted responsibility for an IMS output message by either logging or writing out the data.

To ensure that a recoverable transaction can be recovered, IMS:

- Requests a DR2 response, or, if Fast Path output and no output messages are waiting to be sent, requests exception DR2 and waits for subsequent data or RTR as acknowledgment of the output.
- Maintains the message sequence number and a copy of the message until a DR2, input data, or RTR, if Fast Path, is returned.
- Returns any requested DR1 or DR2 after having accepted responsibility for the message by either logging or writing out the data.

If a failure occurs between the sending of a recoverable message and receipt of the acknowledgment or reply message, the sender cannot determine whether the message reached its destination. During the restart procedure, IMS uses the VTAM STSN command to inform the controller of the sequence numbers of the last inbound sync-point message IMS received and of the last outbound sync-point message IMS sent. Any messages that have not been received can then be retransmitted.

When recoverable messages are being sent, only one recoverable message can be outstanding at a time. This means that the sender should send one message and wait for the response or reply before sending another. IMS reads a message, places the message on the input queue, and returns the DR1 or DR2 response or reply before it accepts another message.

### Irrecoverable-Inquiry Transactions

IMS treats an irrecoverable transaction in the same manner as a recoverable transaction, except that all processing required to achieve recoverability is eliminated. As a result, irrecoverable transactions require less processing time but can be lost in the event of a failure (for example, line failure, processor failure, queue failure) in the network.

A irrecoverable transaction need not request a DR1 or DR2. Because recoverability is not guaranteed, the receiver of the message is not required to acknowledge receipt. A irrecoverable transaction should, however, request an exception DR1 or

## IMS Transaction Types

DR2. The exception DR1 or DR2 requires fewer line transmissions than DR1 or DR2, because under normal circumstances no response is returned.

## Verifying IMS Receipt of Irrecoverable Messages

The SLU P system does not request that IMS acknowledge receipt of irrecoverable messages. However, if the system is transmitting multiple irrecoverable messages, indicating both begin-bracket and end-bracket, and wants to verify their receipt, it can send the VTAM CHASE command. VTAM indicators must request a DR1. When the SLU P system receives the normal definite response 1, all messages preceding the CHASE command have been received by IMS.

## IMS Message Switches

Because IMS treats a message switch that is sent or received just like a recoverable transaction:

- A message switch must request a DR1 or DR2 if the workstation is defined with the ACK option. If the OPTACK option is specified, a message switch optionally requests an exception DR1 or DR2 and should specify begin-bracket and change-direction on input to IMS.
- A message switch is recovered by IMS.
- The sequence number of a message switch is used in message resynchronization.

## IMS Commands

The network system analyst decides whether a workstation is allowed to enter IMS commands and, if so, which commands are allowed. IMS does not require that the controller request a specific response when sending an IMS command. DR1, DR2, and exceptions DR1 and DR2 are allowed; exception DR1 or DR2 is recommended.

IMS processes its commands and returns an IMS message when it finishes processing the command. Therefore, when a DR1 or DR2 is requested, IMS returns the DR1 or DR2, followed immediately by the message confirming command completion.

The additional line transmission required to send either the DR1 or DR2 and the command completion message can be eliminated if the controller requests exception DR1 or DR2.

If exception DR1 or DR2 is requested, the command completion message acknowledges receipt of the command and indicates that command processing is complete. If the ACK option is defined for the workstation, the VTAM sequence number of the command does not participate in message resynchronization as it does when only DR1 or DR2 is requested. If the OPTACK option is defined, all input participates in message resynchronization.

## VTAM Commands and Indicators

When a VTAM command is sent to IMS, a DR1 must be requested.

When IMS sends a VTAM command, it requests a DR1. If the BIND or BID command is sent, the response must be either a DR1 or exception DR1. If the STSN command is sent, a DR1 is required.

The controller responds to all other VTAM commands and indicators for the workstation.

## MFS Control Requests

When Message Format Service (MFS) is used, MFS control requests can be used to display paged messages and to control the display component screen. (The control request is specified in the input function management header.) DR1, DR2, or exception DR1 and DR2 can be requested. Using exception DR1 or DR2 is recommended, because the additional line transmission required to send DR1 or DR2 and the completion message can be eliminated if the controller requests exception DR1 or DR2.

---

## Error Handling

This section describes the procedures used by IMS and required of the controller or the controller application program to handle failures resulting from transmission or protocol errors.

## IMS-Detected Errors

Whenever IMS detects an abnormal send or receive condition from VTAM, NCP, or the controller, it terminates the session. When an error on a received message is detected, IMS protects any display component and returns an exception DR1 or DR2 that includes 4 bytes of sense information:

**Bytes 0, 1**      System sense field

**Bytes 2, 3**      User sense field

### System Sense Field (SSENSE)

Bytes 0 and 1 contain one of the following values:

**X'0800'**          The user sense field contains the user message from the user message table.

**X'0819'**          The user sense field contains the IMS message number 290 (in binary). No output is available. This value is set only for a non-MFS response to a previously received RTR command.

**X'0826'**          The user sense field contains the IMS message number.

### User Sense Field (USENSE)

IMS uses the user sense field to pass on the number (converted to binary) of the appropriate error message. For example, an invalid transaction code generates the message:

```
DFS064 DESTINATION CANNOT BE FOUND OR CREATED
```

If IMS receives an invalid transaction code, it returns an exception DR1 or DR2 response with X'0040' in the user sense field.

In the XRF complex, the alternate system sends SSENSE=X'0826', USENSE X'0F15', and a DFS3861I SYSTEM TAKEOVER OCCURRED message if an input transaction is lost across the XRF session takeover. You must then retransmit the last input record.

## Error Handling

**Related Reading:** For more information on responding to error messages transmitted during a system takeover, see “Writing the Controller Application Program for XRF Systems” on page 403.

IMS binds each session with unconditional bracket termination. When IMS sends an exception DR1 or DR2 to input, it must remain in a send state (regardless of the current bracket state), and the workstation must go to a receive state. Both IMS and the workstation must go to a between-brackets state if the input message in error specified end-bracket. IMS then places the workstation between-brackets and into a contention state by sending the entire IMS error message as a single chain with the appropriate bracket indicators.

An IMS error message can have multiple segments, with a maximum message length of 32,000 bytes. However, all current IMS error messages that are sent to the workstation are single-segment and less than 132 bytes. These messages adhere to the FM header and transmission rules for all output messages.

The generation of the user sense data is dependent on the standard IMS error message format. The session can be terminated if any user Output edit routine modifies this format.

## Controller or Station-Detected Errors

Whenever the controller detects an error on a message from IMS, or simply cannot accept the message at that time, an exception DR2 that includes 4 bytes of sense data is returned:

**Bytes 0, 1**      System sense field

**Bytes 2, 3**      User sense field

### System Sense Field

Bytes 0 and 1 should contain one of the following values:

**X'0802'**      RECOVERABLE ERROR—If the message is recoverable, IMS returns it to the output queue and retransmits it following the next input from that workstation or whenever additional output for that destination is queued. Retransmission of chained messages begins from the first-in-chain transmission. Because any open bracket is closed upon receipt of the exception DR2, the retransmitted message contains end-bracket only or both begin-bracket and end-bracket indicators (depending upon whether input occurred or whether the bracket indicators were specified). Irrecoverable messages have been dequeued and cannot be recovered and retransmitted.

**X'0811'**      BREAK—Cancel output. The output message is terminated. This message is dequeued from IMS and is not retransmitted; it is not subject to recovery. Any remaining output messages that are queued for the workstation are sent. If additional output is not available after receiving this sense code, IMS sends a null output message.

**Related Reading:** For more information on output messages, see “Output Messages” on page 447.

**X'0812'**      TEMPORARY ERROR—Resource not available. IMS action same as for X'0802'.

<b>X'0813'</b>	TEMPORARY ERROR, BID OR BRACKET REJECTED—NOT READY TO RECEIVE follows. IMS action is the same as for X'0802'.
<b>X'0814'</b>	TEMPORARY ERROR, BID OR BRACKET REJECTED—READY TO RECEIVE follows. IMS action is the same as for X'0802'.
<b>X'081C'</b>	STATION/COMPONENT DOWN—If byte 3 indicates a valid component (X'01' through X'04'), IMS marks the indicated component as inoperable. If byte 3 indicates X'00' or an invalid component (other than X'01' through X'04'), IMS interprets this as a session termination request and terminates the session. If the message in progress is recoverable, IMS returns it to the output queue and retransmits it when the workstation or component is again operable. Irrecoverable messages cannot be recovered and retransmitted.

Any other value for bytes 0 and 1 are handled as a request to terminate the session.

If the BID option is defined, the controller application program should use either X'0813' or X'0814' in an exception DR1 to the BID command; otherwise, the session is terminated because of an invalid response.

The X'0813' sense code results in the message being returned to the IMS output queue if it is recoverable, or dequeued if it is irrecoverable. Any recoverable data returned to the queue causes the bid to be sent again following the next input from the workstation, or whenever additional output for that destination is queued.

The X'0814' sense code causes a recoverable message to be returned to the IMS output queue, or a irrecoverable message to be dequeued. IMS then waits for an RTR command. The workstation can now send input to IMS. Until an RTR is received, IMS does not initiate any output when in a between-brackets state. If IMS is left in an in-bracket send state (the last input carried BB/CD), IMS sends output with EB to terminate the current bracket.

### User Sense Field

This field is used when the system sense field is set to X'081C'. Byte 2 is not inspected by IMS and, therefore, can contain any value.

Byte 3 contains the hexadecimal identification (X'nn') of the component to which the output message is directed. IMS marks that component as inoperable and continues to send output to other operable components of the workstation. If X'00' or an invalid component is specified, the session is terminated.

IMS binds each session with unconditional bracket termination. This means that both IMS and the workstation go to between-brackets and contention states for all supported sense codes except X'0813' and X'0814'. For these two sense codes, IMS goes to between-brackets and receive states; the workstation goes to between-brackets and send states.

## VTAM Logical Unit Status (LUSTATUS) Command

If the SLU P system detects a component-down or workstation-down condition, and the SLU P system is not currently receiving a message from IMS, it can send the VTAM LUSTATUS command to notify IMS of the condition. Four bytes of data, identical in format to the exception DR1 or DR2 for STATION/COMPONENT DOWN, must

## Error Handling

accompany the LUSTATUS. When IMS receives the LUSTATUS, it marks the indicated component inoperable or terminates the session.

The workstation can also notify IMS when the component is operable by sending the VTAM LUSTATUS. The system sense field must equal X'0001', and the user sense field byte 3 must equal a value from X'01' through X'04'. IMS resets the inoperable condition for the indicated component and sends any output queued for it.

All other values for LUSTATUS are considered requests to terminate the session.

## VTAM Ready-to-Receive (RTR) Command

The following is a summary of previously defined IMS functions performed upon receipt of an RTR command:

- Any outstanding Fast Path output message is dequeued and IMS removes the workstation from terminal response mode.
- Any output-protected components are marked unprotected and available for output.
- Output that is suspended because of a X'0814' sense code on a previous exception response is now allowed.
- If IMS output is available to be sent, IMS returns a DR1 response, immediately followed by the output message.
- If no IMS output is available, or if the workstation is in a status where no output can be sent (output stopped or quiesced), IMS returns an exception DR1 followed by an error message indicating no output is available.
- If an RTR is received during the transmission of MFS-paged output, IMS returns an exception DR1 followed by an error message indicating invalid paging request.
- Any exception response returned by IMS causes any display component defined for the logical unit to be protected and unavailable for output.

## VTAM CANCEL Command

The VTAM CANCEL command is used by IMS, and should be used by the SLU P system to terminate a chained message in progress that has not completed. IMS sends the CANCEL command to a workstation in the following situations:

- The IMS master terminal operator issues the IMS /DEQUEUE command and IMS has not yet sent the last-in-chain transmission of a message.
- IMS receives an exception DR1 or DR2 to a transmission of a message and has not yet sent the last-in-chain transmission.

The controller or controller application program should use the CANCEL command in the following situations:

- When an exception DR1 or DR2 is received from IMS prior to sending the last-in-chain transmission of a message.
- When multi-RU chained input to IMS must be cancelled prior to sending the last-in-chain transmission of a message. In this case, if the message to be cancelled contains only the begin-bracket indicator, the VTAM CANCEL command can include the end-bracket indicator, the change-direction indicator, or no additional indicators.

## VTAM Request-Recovery Command

If, during its normal operation, a workstation detects an error condition that does not terminate the session with IMS but does cause the workstation to resynchronize with IMS (for example, diskette failure in the controller or a program check), the application program can initiate message resynchronization. This is done by sending the request-recovery (RQR) command. IMS responds with a CLEAR command followed by the set-and-test-sequence-numbers (STSN) command. Message resynchronization follows and is the same as that described under "Message Recovery" on page 425.

**Restriction:** Session initiation and resynchronization caused by an SNA Request-Recovery (RQR) command is not allowed when the node is in response mode and the response reply message is not yet available for output; that is, the input response mode transaction is still queued or in the process of execution. Response-mode transactions are not recoverable or restartable prior to the application sync point; therefore, session input acknowledgement does not occur until input processing is complete.

Four bytes of data can accompany a SIGNAL command that is sent to IMS. The last two bytes are user data that is ignored by IMS. The first two bytes are system-signal data, which IMS handles as follows:

**X'0000'**

No specific action is taken. This data can, however, cause IMS to send any available output for operative, unprotected components if the workstation is idle due to a previous exception DR1 or DR2 sent to IMS. This input does not reset the output component protection or display protection.

**X'0001'**

This data is treated as an attention signal that causes IMS to stop sending and to wait for input at the end of the current output message. This type of signal can be used before sending either input or required DR1 or DR2 or exception DR1 or DR2 responses to IMS.

All other values are reserved.

## Error Handling



---

## Part 7. Appendixes



---

## Appendix A. Using Programmed Symbols for IBM 3270

**Definition:** *Programmed Symbols (PS)*, an optional feature on the IBM 3270 terminal, lets you store and access up to six character sets of 190 characters each that are user-definable and program-loadable. After the coded graphic character sets have been loaded into the PS buffers, they can be used on a field-by-field basis using MFS.

If the PS buffers are desired and have not been loaded, they must be loaded by a VTAM application or by some other means.

**Related Reading:** For information on programmed symbols, their use, and loading, see *IBM 3270 Information Display System: 3274 Control Unit Description and Programmer's Guide*.

### **In this Appendix:**

- “Determining Whether the Buffers Are Loaded”
- “Loading the PS Buffer”
- “Solving Load Problems” on page 468
- “Reloading the PS Buffer” on page 469
- “Application Design Considerations” on page 469

---

### Determining Whether the Buffers Are Loaded

If you know that the buffers have already been loaded with the desired programmed symbols, you do not need to resend the entire load programmed symbols data stream. To determine if the buffers are loaded with the desired programmed symbols, use one of the following methods:

- Maintain a handwritten log at the device and record the current status of the PS buffers.
- Run a test transaction that attempts to use the desired programmed symbols. Either the desired transaction's output is successfully displayed or an error message (DFS2078) is sent to the terminal and the message is returned to the IMS message queue, with the terminal left in protected mode. The message must be dequeued and a transaction started to load the programmed symbols buffer.

---

### Loading the PS Buffer

If the operator knows the PS buffers need to be loaded (because the device was just powered on by one of the methods described above, or by some other method), enter a response-mode transaction that loads programmed symbols. An installation-written application program that loads PS buffers should be made available for this purpose. The load PS data stream should be the first part of the message sent by the application program and a notification, such as THE PROGRAMMED SYMBOLS LOAD FOR `psname` COMPLETE should be the remainder. The notification informs the terminal operator when the programmed symbols have been loaded. This transaction requires the MFS bypass option.

**Related Reading:** For information on the MFS bypass option, see *IMS/ESA Application Programming: Transaction Manager*.

When the output device for which the PS buffers are to be loaded includes a printer or a different display, another technique must be used.

## Loading the PS Buffer

**Example:** The following events illustrate one example of using an automated operator application program:

1. The master terminal operator at `Display_A` enters a transaction (response or conversational) requesting that programmed symbols be loaded for `Display_A`, `Printer_B`, and `Display_C`.
2. In processing the transaction, an AOI program assigns LTERMs for `Printer_B` and `Display_C`, temporarily, to a special PTERM reserved for loading the PS buffers. The AOI program assigns dummy LTERMs to `Printer_B` and `Display_C`.
3. The AOI program inserts the message containing the load programmed symbols messages to the dummy LTERMs of `Printer_B` and `Display_C`.
4. The AOI program sends a programmed symbols message to `Display_A`.
5. The master terminal operator visually verifies messages on both displays and the printer and confirms to the transaction that the load was successful.
6. The transaction reassigns the LTERMs back to their original status.

---

## Solving Load Problems

If a hardware error occurs while loading a programmed symbols buffer, the following actions occur:

1. The message used to load the programmed symbols is returned to the IMS message queue.
2. The terminal is taken out of service  
**Exception:** SLU 2 devices are not taken out of service.
3. The error is logged on the IMS log.
4. Message DFS2078 is sent to the IMS master terminal. For AOI purposes, provide the message number, if possible.

After the hardware error is corrected and the terminal is in service, the message to load the programmed symbols is re-sent.

If the loading of the programmed symbols fails because of an error in the message to load the programmed symbols, the operator must:

- Dequeue the message. The master terminal operator might need to issue the dequeue (/DEQ) command.
- Correct the error.
- Reenter the transaction to resend the programmed symbols load message.

If a method is available for informing the next user of the programmed symbols buffer status, the terminals with loaded PS buffers should not be powered off. When a power failure occurs or a terminal is powered off, the contents of the PS buffers might be lost.

When a terminal is powered on, if no IMS messages are waiting to be sent to the display, an IMS response mode transaction should be entered to load all required PS buffers or some non-IMS method should be used to load the PS buffers. However, if IMS messages are waiting to be sent, and these messages require the use of one or more PS buffers, sending the queued messages must be delayed until the PS buffers can be reloaded.

For SLU 2 devices, if the PS buffers are not loaded and a message requiring the use of a programmed symbols buffer is sent to the terminal, the message is rejected and IMS:

- Returns the invalid message to the IMS queue
- Logs the error on the IMS log
- Sends message DFS2078 to the IMS master terminal and to the node, indicating an output failure.

The terminal is left in protected mode after receipt of message DFS2078.

A user application program can be designed to queue to an LTERM an unsolicited message requiring a particular programmed symbols buffer to be loaded. The first part of the message can include a load programmed symbols data stream; however, this message cannot be processed by MFS.

---

## Reloading the PS Buffer

When a message is waiting on the IMS queue for a terminal and requires programmed symbols that are not loaded, take one of the following actions:

- If the terminal is attached by VTAM, load the PS buffers using a VTAM application.
- If a queued message requires a programmed symbols buffer, and it is normal user output (for example, the output is not response mode or conversational), then use a response-mode transaction to load the programmed symbols buffer, which loads the programmed symbols buffer so that the queued message is properly displayed.
- Dequeue (/DEQ) the message (or have the master terminal operator dequeue the message) requiring use of a programmed symbols buffer; enter a transaction to load the programmed symbols buffer required; and then reenter the transaction that originally generated the queued message.

---

## Application Design Considerations

A message that uses programmed symbols is different from other IMS output messages; it relies on the correct PS buffer content for its presentation. One application might require a PS buffer content that is incompatible with a different application. Further, the PS buffer can be erased by external action (such as powering off the terminal). IMS helps in managing this PS buffer by keeping messages that cannot be displayed on the IMS message queue; this allows the PS buffer to be loaded and the message re-sent.

Multiple output messages can be queued for the same device. IMS uses a set algorithm to determine which message to send to the terminal next. For example, response-mode output is always sent before any nonresponse-mode output messages are sent; this is very useful for programmed symbols, because it allows a transaction to be entered to load the PS buffer before other queued messages are sent to the terminal.

Provide a response-mode transaction to load the PS buffer. This transaction allows its output to overtake any nonresponse-mode output that requires the PS buffer to be loaded. A database keyed by LTERM name can be used to record the program symbol set name that is to be loaded into the buffer. All IMS applications using program symbols need to update this value whenever the PS buffer is loaded. In this way the terminal operator never needs to know the program symbol name.

Output messages using PS should be sent as nonresponse-mode, allowing the PS buffer to be reloaded using a special response-mode transaction (described above). An alternative is to use a non-IMS program to load the PS buffer.

## Application Design Considerations

An Automated Operator Interface (AOI) routine can be triggered by the DFS2078 message, indicating a possible need to reload the PS buffer. This avoids the need for any human intervention. If you choose this technique, provide for invalid or corrupted 3270 data streams that produce the DFS2078 message, rather than an actual PS buffer problem. Be careful to design the AOI routine to detect and avoid looping if it automatically reactivates the terminal.

---

## Appendix B. Bind Parameters for SLU P and LU 6.1

This appendix lists the session parameters that IMS specifies when establishing connection using the VTAM OPNDST macro instruction. These parameters define the rules that a logical unit must follow when communicating with IMS.

**Restriction:** A VTAM restriction exists on the OUTBUF and RECANY buffer sizes for logical units requiring a bind from IMS.

**Related Reading:** For more information, see *IMS/ESA Installation Volume 2: System Definition and Tailoring and VTAM V4R2 for MVS/ESA Resource Definition Reference*.

### **In this Appendix:**

- “Finance Communication System Bind Parameters”
- “LU Type 6.1 Bind Parameters” on page 473

The following outbound (from IMS to the SLU) BIND formats apply to the device types indicated in this appendix.

**Related Reading:** For information on the inbound (initial logon) user data format, which applies to non-ISC device types, see “Appendix D. Format for CINIT User Data Parameters” on page 485.

---

## Finance Communication System Bind Parameters

The session parameters for a Finance Communication System, defined as UNITYPE=FINANCE or UNITYPE=SLUTYPEP on the TYPE macro or an Extended Terminal Option (ETO) logon descriptor, are set as follows:

Table 46. Finance Communication System Bind Parameters

BIND FORMAT Description	Byte	Bit	Content
	0		X'31'
Format=0; BINDTYPE=COLD <sup>1</sup>	1		X'01'
FM Profile 4	2		X'04'
TS Profile 4	3		X'04'
Primary NAU protocol	4		
Multiple RU chains		0	B'1'
Immediate request		1	B'0'
Any response		2-3	B'11'
Reserved		4-5	B'00'
No compression		6	B'0'
Primary can send EB		7	B'1'
Secondary NAU protocol	5		B'1'
Multiple RU chains		0	B'1'
Immediate request		1	B'0'
Any response		2-3	B'11'
Reserved		4-5	B'00'
No compression		6	B'0'
Secondary can send EB		7	B'1'

## Finance Communication System Bind Parameters

Table 46. Finance Communication System Bind Parameters (continued)

BIND FORMAT Description	Byte	Bit	Content
Common NAU protocol	6		X'60'
Reserved		0	B'0'
Message headers are allowed		1	B'1'
Brackets to be used		2	B'1'
Bracket termination rule 2 (unconditional)		3	B'0'
No alternate code		4	B'0'
Reserved		5-7	B'000'
Common NAU protocol	7		X'80'
FM transmission mode - HDX flip/flop		0-1	B'10'
Primary ERP responsibility		2	B'0'
Secondary station is the first speaker		3	B'0'
Reserved		4-5	B'00'
No related chains		6	B'0'
Contention resolution		7	B'0'
Reserved: X'00' should be specified	8		
SLU receive pacing count: Not changed by IMS <sup>2</sup>	9		
SLU max. RU send size: Set to maximum receive any buffer size from IMS system definition <sup>1, 3, 4</sup>	10		
PLU max. RU send size: Set from output buffer size specified on IMS system definition <sup>1, 4</sup>	11		
Reserved: X'0000' should be specified	12-13		
LU TYPE: X'00' should be specified	14		
Reserved: XL11'00' should be specified	15-26		
PLU name length <sup>2</sup>	27		
PLU name: IMS ACB name <sup>2</sup>	28-35		
User data length	36		X'0B'
User data <sup>5</sup>	37-47		
Structured data indicator	37		X'00'
Length of USERVAR segment	38		X'09'
USERVAR segment indicator	39		X'03'
USERVAR	40-47		XRF USERVAR

### Notes:

- Bytes 1 through 7, 10, and 11 are set to the indicated values set by IMS and cannot be changed by the user. The pacing parameter is defined in the VTAM list LU definition or through the mode table entry for the LU (VTAM DLOGMOD parameter).
- Bytes 0, 9, and 27 through 35 are set by VTAM.
- The receive-any buffer size is determined by the user-supplied value for size on the RECAN keyword parameter of the COMM macro statement, less 28 bytes. The resulting value is input to the algorithm described in *VTAM V4R3 for MVS/ESA Programming*.
- IMS does not set the buffer size for the 4701/4702.
- When the BIND data issued is an XRF system, the following structured user segment is included:
  - Length of USERVAR segment - X'09'
  - USERVAR segment indicator - X'03'
  - USERVAR (8 bytes) - USERVAR of the XRF system



## LU Type 6.1 Bind Parameters

This section provides information on logical unit type 6.1 bind parameters.

### IMS as Primary Half Session

The following bind format is sent by IMS during an IMS-to-other session initiation. If the mode table entry indicates negotiated bind, IMS overrides the mode table primary NAU protocol field with the indicated values prior to sending the bind.

IMS allows some parameters to be optionally set by a VTAM mode table entry or negotiated bind response. IMS then operates within the indicated constraints. For a non-negotiated bind, IMS checks the parameters for validity before sending the bind. For negotiated bind, IMS checks the parameters for validity prior to sending the BIND request and upon receipt of the bind response, because the secondary half session can modify parameters within the constraints indicated in the following chart.

Table 47. Logical Unit Type 6.1 Bind Parameters

BIND FORMAT Description <sup>1</sup>	Byte	Bit	Content
Bind request code	0		X'31'
FORMAT: TYPE 0	1	0-3	B'0000'
BIND TYPE: 0000 - negotiated 0001 - non-negotiated		4-7	B'0000', B'0001'
FM Profile 18	2	0-7	X'12'
TS Profile 4	3	0-7	X'04'
<b>Primary NAU protocol</b>			
Multiple RU chains	4	0	B'1'
Immediate request		1	B'0'
10-Definite response chains 11-Exception/Definite response chains		2-3	B'10', B'11' (B'11' set for negotiated BIND)
Cannot send prepare		4	B'0'
Reserved		5	B'0'
No compression		6	B'0'
0-Primary cannot end bracket 1-Primary can send end bracket		7	B'1'
<b>Secondary NAU protocol</b>			
0-Single RU chains, 1-Multiple RU chains	5	0	B'1'
Immediate request		1	B'0'
01-Exception response chains			
10-Definite response chains			
11-Exception/Definite response chains		2-3	B'10', B'11'
Cannot send prepare		4	B'0'
Reserved		5	B'0'
No compression		6	B'0'
0-Secondary cannot end bracket 1-Secondary can send end bracket		7	B'0', B'1'
<b>Common NAU protocol</b>			
Reserved	6	0	B'0'

## LU Type 6-1 Bind Parameters

Table 47. Logical Unit Type 6.1 Bind Parameters (continued)

BIND FORMAT Description <sup>1</sup>	Byte	Bit	Content
Function management headers allowed		1	B'1'
Bracket state			
0 - Bracket state mgr reset to in-bracket state			
1 - Bracket state mgr reset to between-bracket state <sup>2</sup>		2	B'0',B'1'
Conditional bracket termination		3	B'1'
No alternate code		4	B'0'
STSN request flag		5	B'0' Sequence numbers not available B'1' Sequence numbers available
BIS sent flag		6	B'0' BIS not sent B'1' BIS sent
Reserved		7	B'0'
FM transaction mode, HDX-FF	7	0-1	B'10'
Sender ERP		2	B'1'
Secondary is first speaker		3	B'0'
Reserved		4-6	B'0000'
If byte 6 bit 2 is 0, then			
0-Secondary speaks first after data traffic active state			
1-Primary speaks first after data traffic active state <sup>2</sup>		7	B'0', B'1'
<b>TS Usage</b>			
Secondary send pacing count	8	0-7	unchanged
Secondary receive pacing count	9	0-7	unchanged
SLU max send RU size <sup>3</sup>	10	0-7	set from user define receive-any
PLU max send RU size	11	0-7	set from user define outbuffer size
Primary send pacing count	12	0-7	unchanged
Primary receive pacing count	13	0-7	unchanged
<b>Presentation Services</b>			
LU profile (LUTYPE6)	14	0-7	X'06'
Function management header subset	15	0-7	X'00' reserved
Primary half-session flags	16	0	Reserved
		1	Reserved
		2	1 - FMH6: Receive SYMSG supported
		3	1 - Receive SCHEDULER model supported
		4	1 - Receive QMODEL supported
		5	0 - Linear file model ignored
		6	0 - DL/I model ignored
		7	reserved
	17		reserved
	18-19		reserved
Secondary half-session flags	20	0	reserved

## LU Type 6-1 Bind Parameters

Table 47. Logical Unit Type 6.1 Bind Parameters (continued)

BIND FORMAT Description <sup>1</sup>	Byte	Bit	Content
		1	reserved
		2	1 - Receive SYSMMSG supported
		3	0 - Receive SCHEDULER not supported <sup>4</sup> 1 - Receive SCHEDULER model supported
		4	0 - Receive QMODEL ignored 1 - Receive QMODEL supported
		5	0 - linear file model ignored
		6	0 - DL/I model ignored
		7	reserved
	21		reserved
	22-26		reserved
<b>Variable-length fields</b>			
Length of PLU name	27	0-7	
PLU name	28-M		
Length of user data	M+1	0-7	X'00' no user data present
User data	M+2-N		X'00' structured fields follow ~X'00' first byte of unstructured user data <sup>5</sup>
For unstructured user data	M+3-N		remainder of unstructured user data
For structured user data <sup>6</sup>	M+3-N		structured fields
Length of URC <sup>7</sup>	N+1	0-7	structured fields, request correlation (URC) field X'00' = no URC present
URC <sup>4</sup>	N+2-P		end user-defined identifier
Length of secondary LU name <sup>4</sup>	P+1	0-7	X'00'=no secondary name
Secondary LU name <sup>4</sup>	P+2-R	0-7	secondary LU name

## LU Type 6-1 Bind Parameters

Table 47. Logical Unit Type 6.1 Bind Parameters (continued)

BIND FORMAT	Description <sup>1</sup>	Byte	Bit	Content
<b>Notes:</b>				
1. The length of the BIND RU cannot exceed 256 bytes; otherwise, a negative response is returned.				
2. Set to indicate possible in-bracket or process restart. Set by IMS on bind when response mode output remains on the queue or when IMS is in conversational mode. Can also be sent on a negotiated bind response by the other half session.				
3. The receive-any buffer size is determined by the user-supplied value for size on the RECANY keyword parameter of the COMM macro statement, less 28 bytes. The resulting value is input to the algorithm described in <i>VTAM V4R3 for MVS/ESA Programming</i> .				
4. When the bind indicates that the other half session does not support the SCHEDULER process, IMS sends all unsolicited and asynchronous output using ATTACH.				
5. Unstructured user data is ignored and not provided by IMS.				
6. Structured user data formats. A structured field contains architected or subsystem-defined information and provides a means for subsystems to communicate data. Each structured field contains a field identifier (subfield number) and length. A structured data field can contain unstructured data. If structured field (M+3-N) is X'00', it contains unstructured data as follows:				
1		Length of unstructured data field (including subfield key field). If zero, this field can be omitted entirely.		
2		Subfield key: X'00'		
3-N		Unstructured data. If the structured subfield number is X'03', an 8-byte USERVAR name follows the subfield.		
If the structured field (M+3-N) is X'01', it contains a session qualifier as follows:				
1		Length of session qualifier field (including subfield key field). If zero, this field can be omitted entirely.		
2		Subfield key: X'01'		
3		Length of primary resource qualifier (X'00' means no primary source qualifier is present). Values 0 to 8 are valid.		
4-N		Primary resource qualifier		
N+1		Length of secondary resource qualifier (X'00' means no secondary resource qualifier is present). Values 0 to 8 are valid.		
N+2-M		Secondary resource qualifier		
M+1		Length of password (X'00' means no password is present). Values 0 to 8 are valid.		
M+2-P		Password. Ignored on bind or bind reply from IMS and is not sent on bind or bind reply. IMS indexes structured fields to find field X'01', a session-qualifier field, when these parameters are required for session initiation using parallel sessions.		
M+3-N		If the structured field has a subfield of X'02', IMS interprets the field as an MSC partner ID. If bind is issued in an XRF environment, an additional structured segment is included in the user data. The format of this segment is:		
1		Length of USERVAR segment, X'09'		
2		Subfield key, X'03'		
3-10		8-byte field containing USERVAR of the XRF complex		
7. The URC and secondary LU name are not used by IMS but are shown for compatibility purposes.				

## IMS as Secondary Half Session

The following bind format can be received by IMS during an IMS-to-other session initiation. If the bind parameters received indicated a negotiated BIND request, IMS overrides the secondary NAU protocol field with the indicated values before sending the bind response.

## LU Type 6-1 Bind Parameters

IMS allows some parameters to be optionally set using the bind and operates within the indicated constraints.

Table 48. IMS-To-Other Secondary Half Session

<b>BIND FORMAT Description<sup>1</sup></b>	<b>Byte</b>	<b>Bit</b>	<b>Content</b>
Bind Request Code	0		X'31'
Format: TYPE 0	1	0-3	B'0000'
Bind type: 0000 - negotiated 0001 - non-negotiated		4-7	B'0000', B'0001'
FM profile 18	2	0-7	X'12'
TS profile 4	3	0-7	X'04'
<b>Primary NAU Protocol</b>			
0-Single RU chains 1-Multiple RU chains	4	0	B'0', B'1'
Immediate request	1	B'0'	01-Exception response chains 10-Definite response chains
11-Exception/Definite response chains		2-3	B'01', B'10', B'11'
Cannot send prepare		4	B'0'
Reserved		5	B'0'
No compression		6	B'0'
0-Primary cannot end bracket 1-Primary can send end bracket		7	B'0', B'1'
<b>Secondary NAU Protocol</b>			
Multiple RU chains	5	0	B'1'
Immediate request		1	B'0'
10-Definite response chains 11-Exception/Definite response chains		2-3	B'10', B'11' (B'11' set for negotiated bind)
Cannot send prepare		4	B'0'
Reserved		5	B'0'
No compression		6	B'0'
0-Secondary cannot send end bracket 1-Secondary can send end bracket		7	B'1'
<b>Common NAU Protocol</b>			
Reserved	6	0	B'0'
Function Management headers allowed		1	B'1'
Bracket state 0 - Bracket state mgr reset to in-bracket state; 1 - Bracket state mgr reset to between-bracket state <sup>2</sup>		2	B'0', B'1'
Conditional bracket termination		3	B'1'
No alternate code		4	B'0'
STSN required flag		5	

## LU Type 6-1 Bind Parameters

Table 48. IMS-To-Other Secondary Half Session (continued)

BIND FORMAT Description <sup>1</sup>	Byte	Bit	Content
			B'0' Sequence numbers not available B'1' Sequence numbers available
BIS sent flag		6	
			B'0' BIS not sent B'1' BIS sent
Reserved		7	B'000'
FM transaction mode, HDX-FF	7	0-1	B'10'
Sender ERP		2	B'1'
Reserved		3-6	B'0000'
IF byte 6 bit 2 is 0: 0 - Secondary speaks first after data traffic active state 1 - Primary speaks first after data traffic active state <sup>2</sup>		7	B'0', B'1'
<b>TS Usage</b>			
Secondary send pacing count	8	0-7	unchanged
Secondary receive pacing count	9	0-7	unchanged
SLU max send RU size	10	0-7	must ≥ defined outbuffer size
PLU max send RU size <sup>3</sup>	11	0-7	must be ≤ define receive-any size
Primary send pacing count	12	0-7	unchanged
Primary receive pacing count	13	0-7	unchanged
Presentation services			
LU profile (LUTYPE6)	14	0-7	X'06'
Function Management header subset	15	0-7	X'00' Reserved
Primary half-session flags	16	0	Reserved
		1	Reserved
		2	1 - Receive SYSMMSG supported
		3	0 - Receive SCHEDULER model not supported <sup>4</sup>
			1 - Receive SCHEDULER model supported
		4	0 - Receive QMODEL not supported
			1 - Receive QMODEL supported
		5	0 - linear file model not supported
		6	0 - DL/I model not supported
		7	reserved
	17		reserved
	18-19		reserved
Secondary half-session flags	20	0	reserved
		1	reserved
		2	1 - FMH6: Receive SYSMMSG supported

## LU Type 6-1 Bind Parameters

Table 48. IMS-To-Other Secondary Half Session (continued)

BIND FORMAT Description <sup>1</sup>	Byte	Bit	Content
		3	1 - Receive schedule model supported
		4	1 - Receive QMODEL supported
		5	0 - linear file model not supported
		6	0 - DL/I model not supported
		7	reserved
	21		reserved
	22-26		reserved
Variable-length fields			
Length of PLU name	27	0-7	
PLU name	28-M		
Length of user data	M+1	0-7	X'00' no user data present
User data	M+2-N		X'00' structured fields follow ~X'00' first byte of unstructured user data <sup>5</sup>
For unstructured user data	M+3-N		remainder of unstructured user data
For structured user data	M+3-N		structured fields <sup>6</sup>
Length of URC <sup>7</sup>	N+1	0-7	length of user request correlation (URC) field X'00' = no URC present
URC <sup>7</sup>	N+2-P		URC: end-user defined identifier
Length of secondary LU name <sup>7</sup>	P+1	0-7	X'00'=no secondary name
Secondary LU name <sup>7</sup>	P+2-R	0-7	secondary LU name

## LU Type 6-1 Bind Parameters

Table 48. IMS-To-Other Secondary Half Session (continued)

BIND FORMAT	Description <sup>1</sup>	Byte	Bit	Content
<b>Notes:</b>				
1.	The length of the BIND RU cannot exceed 256 bytes; otherwise, a negative response is returned.			
2.	Set to indicate possible in-bracket or process restart. Set by IMS on bind when response mode output remains on the queue or when IMS is in conversational mode. Can also be sent on a negotiated bind response by the other half session.			
3.	The receive-any buffer size is determined by the user-supplied value for size on the RECANY keyword parameter of the COMM macro statement, less 28 bytes. The resulting value is input to the algorithm described in <i>VTAM V4R3 for MVS/ESA Programming</i> .			
4.	When the bind indicates that the other half session does not support the SCHEDULER process, IMS sends all unsolicited or asynchronous output using ATTACH.			
5.	Unstructured user data is ignored and not provided by IMS.			
6.	Structured user data formats. A structured field contains architected or subsystem-defined information and provides a means for subsystems to communicate data. Each structured field contains a field identifier (subfield number) and length. A structured data field can contain unstructured data. If structured field (M+3-N) is X'00', it contains unstructured data as follows:			
1	Length of unstructured data field (including subfield key field). If zero, this field can be omitted entirely.			
2	Subfield key: X'00'			
3-N	Unstructured data If the structured subfield number is X'03', an 8-byte USERVAR name follows the subfield.			
If the structured field (M+3-N) is X'01', it contains a session qualifier as follows:				
1	Length of session qualifier field (including subfield key field). If zero, this field can be omitted entirely			
2	Subfield key: X'01'			
3	Length of primary resource qualifier (X'00' means no primary source qualifier is present). Values 0 to 8 are valid.			
4-N	Primary resource qualifier			
N+1	Length of secondary resource qualifier (X'00' means no secondary resource qualifier is present). Values 0 to 8 are valid.			
N+2-M	Secondary resource qualifier			
M+1	Length of password (X'00' means no password is present). Values 0 to 8 are valid.			
M+2-P	Password (ignored on bind or bind reply received by IMS and not sent on bind or bind reply. IMS indexes structured fields to find field X'01', a session-qualifier field, when these parameters are required for session initiation using parallel sessions.  When the BIND data issued is an XRF system, the following structured user segment is included: Length of USERVAR segment - X'09' USERVAR segment indicator - X'03' USERVAR (8 bytes) - USERVAR of the XRF system			
7.	The URC and secondary LU name are not used by IMS but are shown for compatibility purposes.			



## Appendix C. Bind Parameters for SLU 1 and SLU 2

This appendix lists the session parameters that IMS specifies when establishing connection with SLU 1 and SLU 2. These parameters define the rules that a logical unit must follow when communicating with IMS.

### **Related Reading:**

- For more information on using these parameters, see “Specifying Logon Modes When Establishing Connection” on page 233.
- For information on the BIND parameters for SLU P and LU 6, see “Appendix B. Bind Parameters for SLU P and LU 6.1” on page 471.

### **In this Appendix:**

- “SLU 1 Bind Parameters”
- “SLU 2 Bind Parameters” on page 483

The following outbound (from IMS to the SLU) BIND formats apply to the device types indicated in this appendix.

**Related Reading:** For the inbound (initial logon) user data format, which applies to non-ISC device types, see “Appendix D. Format for CINIT User Data Parameters” on page 485.

## SLU 1 Bind Parameters

*Table 49. SLU 1 Bind Parameters*

<b>Description</b>	<b>Byte:</b>	<b>Bit:</b>	<b>Content</b>
Format=0; BINDTYPE=COLD	0		X'01'
FM Profile 3	1		X'03'
TS Profile 3	2		X'03'
<b>PRIMARY LU PROTOCOLS</b>	<b>3</b>		
Multiple RU chains		0	B'1'
Immediate Request Mode		1	B'0'
Chain Response Protocol: Any		2-3	B'11'
Reserved		4-5	B'00'
No Compression		6	B'0'
Primary can send End Bracket		7	B'1'
<b>SECONDARY LU PROTOCOLS</b>	<b>4</b>		
Multiple RU Chains		0	B'1'
Immediate Request Mode		1	B'0'
Chain Response Protocol		2-3	Note 1
Reserved		4-5	B'00'
No Compression		6	B'0'
Secondary does not send End Bracket		7	B'0'
<b>COMMON LU PROTOCOLS (FIRST BYTE)</b>	<b>5</b>		
Reserved		0	B'0'
FM Headers		1	Note 2
Brackets can be used		2	B'1'
Bracket Termination Rule 1 (Conditional)		3	B'1'

## SLU 1 Bind Parameters

Table 49. SLU 1 Bind Parameters (continued)

Description	Byte:	Bit:	Content
EBCDIC (No Alternate Code)		4	B'0'
Reserved		5-7	B'000'
<b>COMMON LU PROTOCOLS (SECOND BYTE)</b>	<b>6</b>		
Half-Duplex Flip-Flop		0-1	B'10'
Primary ERP Responsibility		2	B'0'
Secondary is First Speaker		3	B'0'
Reserved		4-6	B'000'
Secondary is Contention Winner		7	B'0'
Reserved	7	0-1	B'00'
SLU Send Pacing Count (Set by VTAM) <sup>3</sup>		2-7	xx
Reserved	8	0-1	B'00'
SLU Receive Pacing Count (Set by VTAM) <sup>3</sup>		2-7	xx
SLU to PLU RU Size	9	0-7	set from user-defined receive any
PLU to SLU RU Size	10	0-7	set from user-defined outbuf size
Reserved	11	0-1	B'00'
PLU CPMGR Send Pacing Count (Set by VTAM) <sup>3</sup>		2-7	xx
Reserved	12	0-1	B'00'
PLU CPMGR Receive Pacing Count (Set by VTAM) <sup>3</sup>		2-7	xx
LU Profile (LUTYPE1)	13	0-7	X'01'
Remainder of Bind Area	14-35		See Notes

Table 49. SLU 1 Bind Parameters (continued)

Description	Byte:	Bit:	Content
<b>Notes:</b>			
1. IMS forces flip-flop mode.			
2. The preceding bind (bytes 0-6 and 13) overrides anything that might be in a logmode entry.			
3. You can do this by coding the VPACING parameter on the VTAM list LU definition or by specifying the appropriate mode table entry on the LU definition by using the VTAM DLOGMOD parameter.			
4. The remainder of the bind (bytes 14-35) can be specified if it is required by the device. It is taken from the logmode entry.			
5. Unattended operation must be specified in the logmode entry using the following:			
<b>BYTE 18</b>			
<b>Bit 0</b>			
	0		Initiates attended
	1		Initiates unattended
<b>Bit 1</b>			
	0		Does not alternate from attended/unattended during session
	1		Alternates from attended/unattended during session
IMS forces attended mode if the node is defined as the master terminal.			
6. IMS users should make the logmode entry according to the IMS definition.			
7. If the terminal sends SCS2 transparent fields (identified by a X'35' followed by one byte containing the field's length, followed by the transparent field), the bind image bit BINPDSB1=BINTRNDS (offset 17=01) must be set. IMS processes these fields by deleting the X'35' and length byte, and by passing the unaltered transparent field to the editing routine.			
8. IMS accepts a setting of B'10' (definite response), B'01' (exception response), or B'11' (either response). If a setting of B'00' is found, IMS sets B'01' if only the first component is defined, and B'10' if more than one component is defined.			
9. IMS leaves this bit on (FM headers allowed). If off, IMS leaves it off (FM headers not allowed) if only one component is defined, and sets it on if more than one component are defined.			

## SLU 2 Bind Parameters

The bind parameters for SLU 2 devices are set as follows:

Table 50. SLU 2 Bind Parameters

Description	Byte:	Bit:	Content
Format=0; BINDTYPE=COLD	0		X'01'
FM Profile 3	1		X'03'
TS Profile 3	2		X'03'
Primary NAU protocol	3		
Multiple RU chains		0	B'1'
Immediate request		1	B'0'
Any response		2-3	B'11'
Reserved		4-5	XX
No compression		6	B'0'
Primary can send EB		7	B'1'
Secondary NAU protocol	4		
Multiple RU chains		0	B'1'
Immediate request		1	B'0'
Exception response		2-3	B'01'

## SLU 2 Bind Parameters

Table 50. SLU 2 Bind Parameters (continued)

Description	Byte:	Bit:	Content
Reserved		4-5	XX
No compression		6	B'0'
Secondary cannot send EB		7	B'0'
Common NAU protocol	5		
Reserved		0	X
Message headers are not allowed		1	B'0'
Brackets to be used		2	B'1'
Bracket termination rule 1 (conditional)		3	B'1'
No alternate code		4	B'0'
Reserved		5-7	xxx
Common NAU protocol	6		
FM transmission mode: HDX flip/flop		0-1	B'10'
Primary ERP responsibility		2	B'0'
Secondary station is the first speaker		3	B'0'
Reserved		4-6	xxx
For HDX-FF mode, secondary sends first when leaving data traffic reset state		7	B'0'
SLU Send Pacing count: Not changed by IMS	7		
SLU Receive Pacing count: Not changed by IMS	8		
SLU MAX RU send size: Set to Max receive any buffer size from IMS system definition	9		
PLU MAX RU send size: Set from output buffer size specified at IMS system definition	10		
Reserved: X'0000' should be specified	11-12		
LU TYPE: Set to X'02' by IMS	13		
Reserved: X'00' should be specified	14-18		
User specified screen size if 3274/3276 device. Otherwise not changed.	19-20		
Alternate screen size, X'00' should be specified.	21-22		
Set to X'7E' if 3274/3276 NDS device or X'02' if non-NDS 3270 master terminal. Otherwise should be X'00'.	23		
User specified: should be X'00'	24-25		
PLU Name length	26		
PLU name: IMS ACB name	27-34		
User data length: Not supported, X'00' must be specified	35		

**Notes:**

1. Bytes 0-6, 9, and 10 are set to the indicated values by IMS and cannot be changed by the user.
2. Byte 8 and bytes 26 through 34 are set by VTAM. You should set the remaining bytes to 0 (zero), but it is not mandatory.

## Appendix D. Format for CINIT User Data Parameters

**In this Appendix:** This appendix describes the CINIT user data parameters, and the syntax rules for them.

For all non-MSA VTAM terminal types, IMS can receive user data parameters from the following sources:

- IMS (/OPNDST command)
- IMS autologon request
- User logon (such as SNA INITSELF command)
- Installation Logon exit routine (DFSLGNX0)
- Signon exit routine (DFSSGNX0)
- Output Creation exit routine (DFSINSX0)

User data parameters are optional for migration purposes.

**Exception:** When a match must be made between the terminal and user for session initiation, user data parameters are required for:

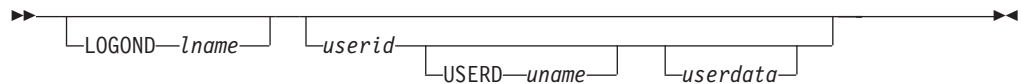
- ISC parallel session (LU 6.1 architecture)
- Finance (3601) and SLU P terminals, when used with ETO

All parameters, optional and required, appear in the CINIT user data field and are available to IMS when the VTAM Logon exit routine is scheduled.

During logon and signon processing, IMS performs minimal processing on CINIT user data parameters before calling the Logon exit routine (DFSLGNX0) and the Signon exit routine (DFSSGNX0). If the Logon or Signon exit routines are not supplied, IMS assumes a default user data format, as defined below.

**Related Reading:** For the ISC LU 6.1 architected format, see *Systems Network Architecture Format and Protocol Reference: Architectural Logic*.

The format for the user data is:



**Recommendation:** For these parameters, blanks are required and are the only recognized delimiters. You should not use more than one blank to delimit parameters.

### **Restrictions:**

- The LOGOND and USERD parameters are valid only for ETO.
- For ISC parallel sessions, DFSLGNX0 receives only the user ID that translates to the ISC subpool name (SNA PHA/SHA qualifier).
- userid is required for Finance, SLU P, ISC, and output-only devices (such as printers).

The parameters specify the following:

Parameter	Description
-----------	-------------

## Format for CINIT User Data Parameters

<i>lname</i>	Specifies logon descriptor name to be used for the terminal attempting to log on to IMS. <i>name</i> is one to eight bytes in length.
<i>userid</i>	Specifies the user ID of the user logging on to IMS. Supplying <i>userid</i> indicates that the user associated with this ID will also sign on to IMS. <i>userid</i> is one to eight bytes in length.  <b>Restriction:</b> ISC is restricted to only the user ID that translates to the ISC SUBPOOL name (SNA PHS/SHS qualifier).
<i>uname</i>	Specifies the user descriptor name to be used for creating the user control block structure at sign-on. <i>uname</i> is one to eight bytes in length.
<i>userdata</i>	Specifies additional data for the Signon exit routine (DFSSGNX0) or the Sign On/Off Security exit routine (DFSCSGN0) and security products, such as RACF. For the exit routines, your installation defines the format of the <i>userdata</i> fields. For RACF, the format of the <i>userdata</i> is:

Parameter	Description
-----------	-------------

<i>userpw</i>	Identifies the user password associated with the previously entered <i>userid</i> . No keyword precedes the user password. The password is one to eight bytes in length.
---------------	--

<b>GROUP</b> <i>groupname</i>	Identifies <i>groupname</i> as the group name associated with the <i>userid</i> . The GROUP keyword and associated parameter are optional. The group name is one to eight bytes in length.
-------------------------------	--

<b>NEWPW</b> <i>nuserpw</i>	Identifies <i>nuserpw</i> as a new user password that replaces the current password, <i>userpw</i> .
-----------------------------	--

The ETO logon descriptor name, *lname*, applies to IMS logon processing. All remaining optional parameters, however supplied, are passed to ETO user allocation, signon processing, and the security product, such as RACF. If a security product is used, all parameters not applicable to that product must be deleted before it is called. The parameters can be deleted in DFSLGNX0, DFSSGNX0, and DFSCSGN0.

---

## Appendix E. SNA Character String Controls

This appendix provides information on SNA character string controls (SCS), which describe specific functions for the EBCDIC control codes. The primary functions defined are those that format a printed page or an alphanumeric display screen. Functions are also defined for codes that set modes of device operation, define data to be used in a unique fashion, or are used for communication between a device operator and an application program (where the specific function associated with the code is defined in a protocol established between a program and an operator).

An SCS data stream consists of a sequential string of control and data characters. Control function characters in the form of SCS-defined control codes can be intermixed with graphic data characters. Other data types (such as binary and packed decimal) are permitted only in conjunction with the transparent (TRN) control.

SCS control codes appear within the data portion of the request unit (RU). A function management (FM) header can precede SCS data within an RU. Functions such as component selection are performed by FM header functions and are not included as SCS functions.

SCS functions do not include data flow control functions, even when these functions are available to a keyboard operator through keys on the keyboard. For example, CANCEL is a data flow control request that can be initiated by a key on the keyboard.

### **In this Appendix:**

- “Format Controls”
- “Control Function Code Assignments” on page 488

---

## Format Controls

The formatting control functions format the output media at the device on a line and page basis. In addition to these controls, a device using SCS also automatically formats the character string to fit the line length of the device. Automatic new line generation eliminates device line length dependencies from those applications in which a specific output format is not required. Therefore, the same character string is sent to devices with varying line length capabilities without the requirement to reformat the character string.

Where specific line and page formats are required, the formatting control functions are used. The automatic new line feature is always active; however, a character string formatted for a given line length can be presented on a device with a shorter line length without loss of data, but the format is lost. When the situation is reversed, where a character string constructed for a maximum presentation position is less than the line length of the device to which it is directed, use of the smaller maximum presentation position allows the string to be presented without loss of format.

### Control Function Code Assignments

SCS control functions are assigned to EBCDIC codes as shown in Table 51.

Table 51. Control Function Code Assignments

EBCDIC Code	Function	Function Abbreviation
04	Vertical Channel Select	VCS <sup>1</sup>
05	Horizontal Tab	HT
0B	Vertical Tab	VT
0C	Form Feed	FF
0D	Carrier Return	CR
14	Enable Presentation	ENP
15	New Line	NL
16	Back Space	BS
17	Program Operator	POC <sup>1</sup>
24	Inhibit Presentation	INP
25	Line Feed	LF
2B	Format	FMT <sup>1</sup>
34	Presentation Position	PP <sup>1</sup>
35	Transparent	TRN <sup>1</sup>

**Note:**

1. Functions with parameters

**Definitions:** Parameters in SCS can be one of two types:

- *Function parameters* extend the function defined by the function code. For example, the PP control function has a function parameter to explicitly define the positioning function performed. The form for a function parameter is a single EBCDIC character.
- *Value parameters* specify a numeric value for the function. For example, the PP function also has a value parameter for it. If the move is relative to the current position, the value parameter specifies the number of columns or lines the presentation position is to be moved from its current position. If the move is absolute, the value parameter specifies the absolute column or line number to which the presentation position is to move. The form for a value parameter is a 1-byte binary number.



---

## Appendix F. Examples Using ISC Edit ATTACH Parameters

This appendix provides examples of using ISC edit ATTACH parameters.

### In this Appendix:

- “ATTACH and SCHEDULER Parameters with ISC Edit”
- “ATTACH Parameters with the IMS SYSMMSG Process” on page 490
- “ATTACH and SCHEDULER Parameters with IMS MFS” on page 492

---

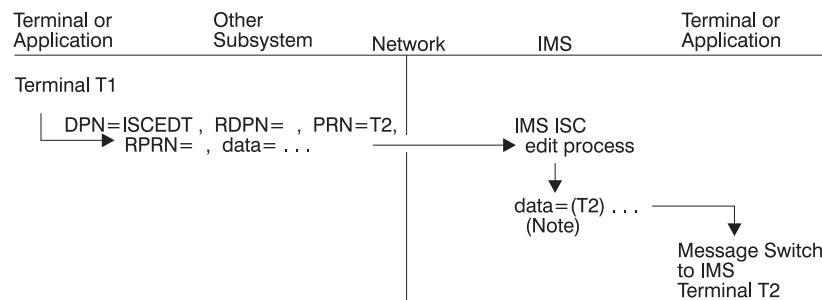
### ATTACH and SCHEDULER Parameters with ISC Edit

The following examples illustrate the use of the ATTACH parameters when using ISC edit for IMS input and output messages.

ISC edit is used for normal transactions, commands, and message switches between LTERMs when one of the following occurs:

- An ATTACH FM header is received while in a between-brackets state that does not specify a destination process name (DPN).
- The DPN names IMS ISC edit as defined during IMS system definition.
- No ATTACH FM header is received either when the attach manager is in the between-brackets reset state or when the active process is ISC edit.
- The DPN=ISCEDT.

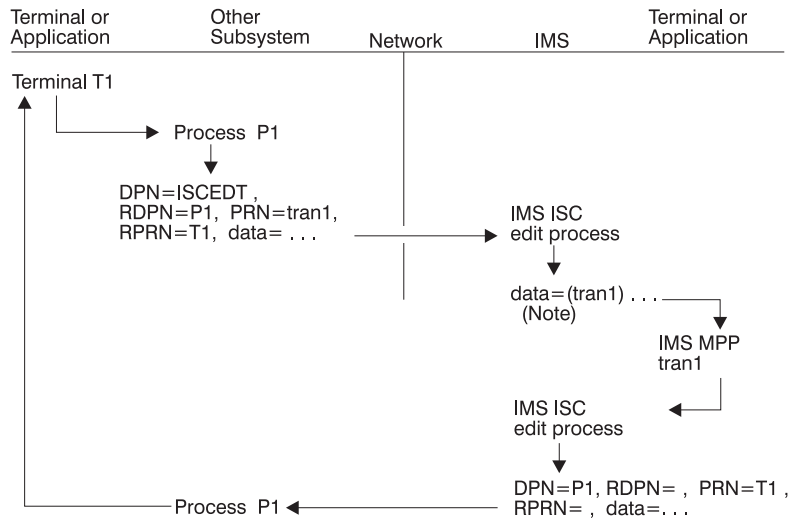
#### Example 1: Message Switch from Other Subsystem Terminal to IMS Terminal



**Note:** The IMS LTERM name for terminal T2 can be supplied either as the input primary resource name (PRN) or as the first field in the data.

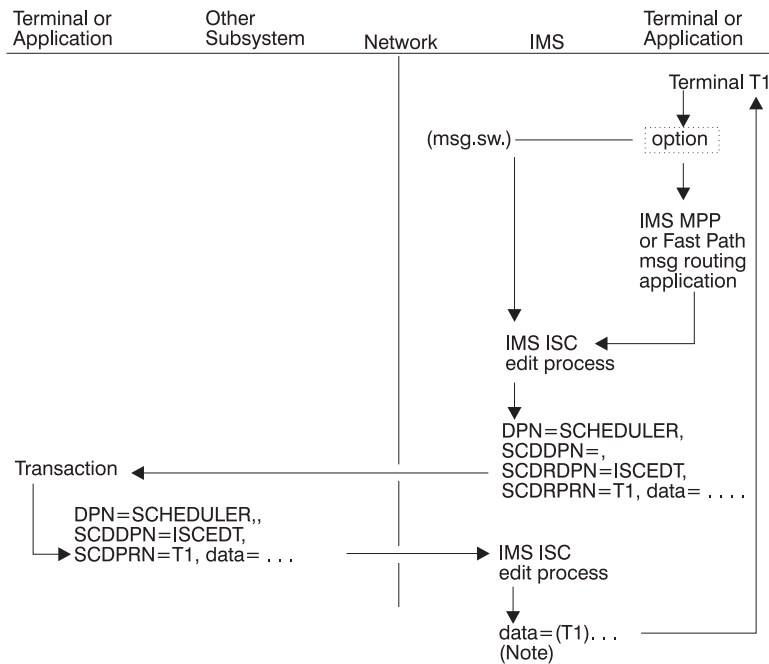
#### Example 2: Other Subsystem Accesses IMS Application Program with Reply Back to Other Subsystem Terminal

## ATTACH and SCHEDULER Parameters with ISC Edit



**Note:** The IMS transaction code can be supplied either as the input PRN or within the data. The input RDPN and RPRN parameters become the DPN and PRN parameters respectively for any resulting reply from the application program when not processed by MFS.

### Example 3: Message Routing



**Note:** The IMS LTERM name can be supplied either as the input SCDPRN or the first data field.

## ATTACH Parameters with the IMS SYMSG Process

The system message process (SYMSG) is indicated through a special SNA process name in the ATTACH header.

## ATTACH Parameters with the IMS SYSMSG Process

Input system messages for IMS ISC sessions are logged and routed directly to the IMS master terminal operator when no PRN is supplied using the input ATTACH parameters. If a PRN value is supplied for IMS ISC sessions, it becomes the IMS input message destination transaction code or LTERM name (message switch), and the input system message is passed to ISC edit. SYSMSG does not provide access to the IMS command processor.

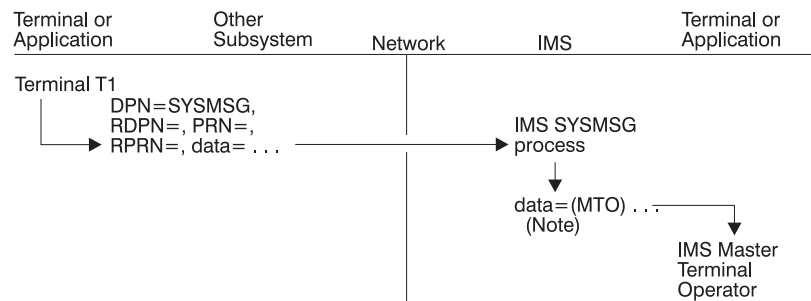
A reply can result from the input SYSMSG if a PRN value supplied is for an IMS transaction. This reply is sent just as is any other (non-SYSMSG) asynchronous output message.

IMS requests attachment of the SYSMSG process to send IMS broadcast and other system messages that are not directly solicited by an input IMS command.

**Exception:** When IMS detects an error condition during an input IMS response mode or conversational transaction, an exception response (selective receiver ERP) to the input transaction occurs and the system message is sent as an ERP message.

The following examples illustrate the use of the ATTACH parameters when attaching the SYSMSG process for IMS input and output messages.

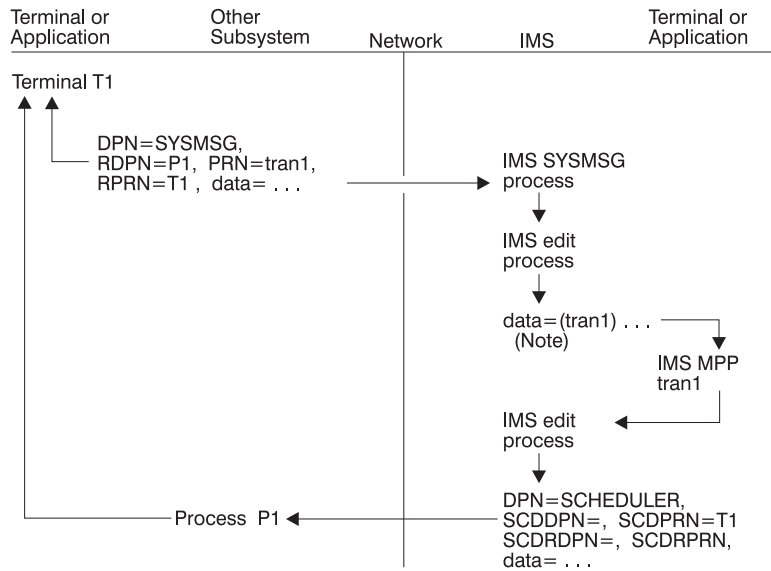
### Example 1: SYSMSG without PRN from Other Subsystem Terminal to IMS A



**Note:** The IMS LTERM name for the master terminal operator is assumed when no input ATTACH PRN parameter is supplied.

### Example 2: SYSMSG with PRN from another Subsystem Terminal to IMS

## ATTACH Parameters with the IMS SYSMSG Process

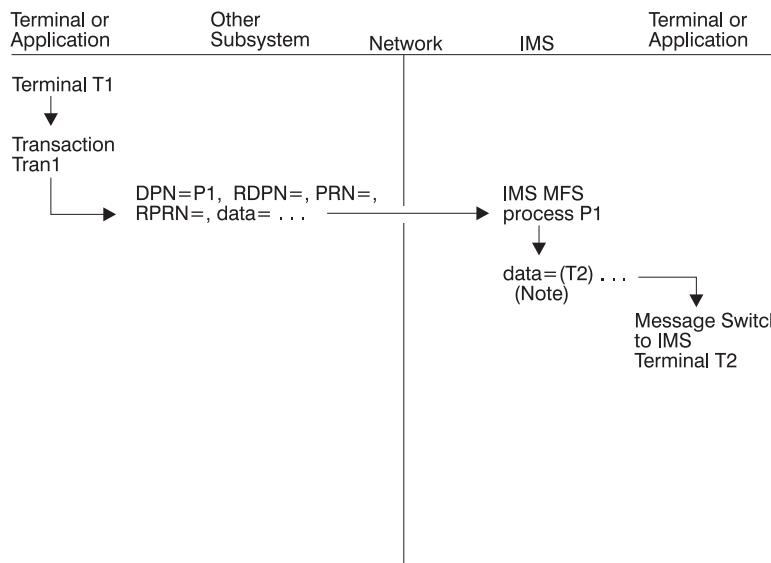


**Note:** The PRN parameter becomes the IMS destination. The input RDPN and RPRN parameters become the output DPN and PRN parameters respectively for any resulting reply from the MPP.

## ATTACH and SCHEDULER Parameters with IMS MFS

The following examples illustrate the use of the ATTACH and SCHEDULER parameters when using IMS MFS for IMS input and output messages.

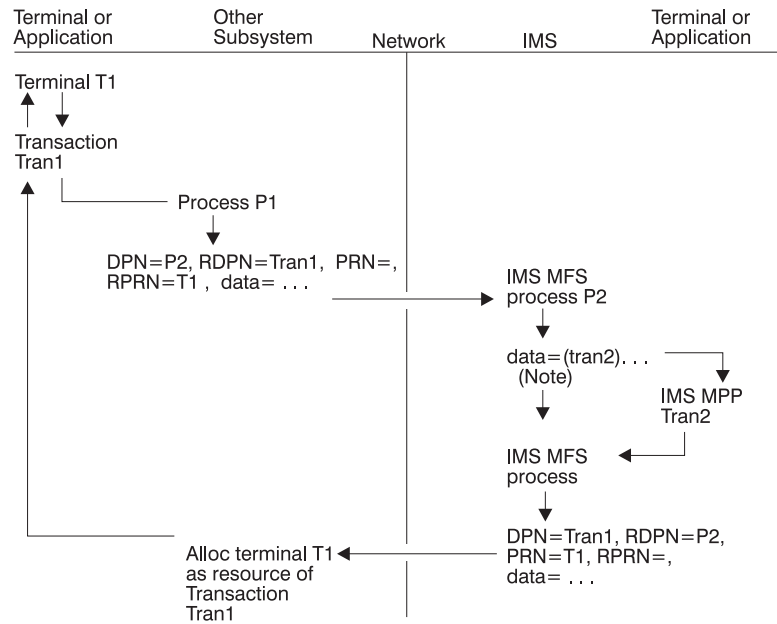
### Example 1: Message Switch from Other Subsystem Terminal to IMS Terminal



**Note:** The IMS LTERM name for terminal T2 can be supplied as the input primary resource name (PRN), as a data field, or as an MFS-defined literal. RPRN and RDPN, if supplied, can be optionally included in the data presented to the IMS terminal by MFS. The DPN parameter must be supplied in the input ATTACH FM header to invoke MFS.

## ATTACH and SCHEDULER Parameters with IMS MFS

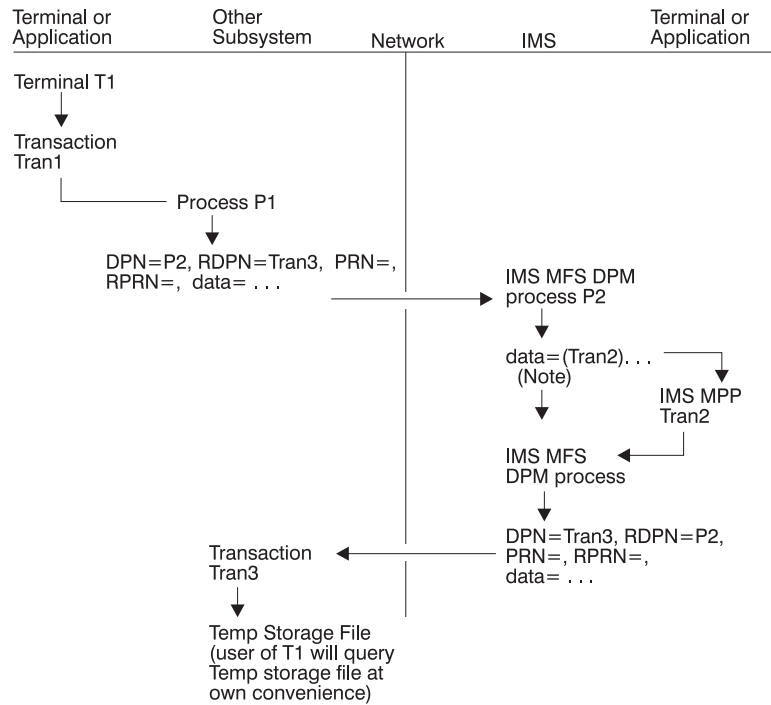
### Example 2: Other Subsystem Terminal Accesses IMS Application Program with Reply



**Note:** The IMS transaction code can be supplied as the input PRN, as a data field (first field for basic edit), or as an MFS-defined literal. The DPN parameter must be supplied to invoke IMS MFS DPM. If a reply inserted by the MPP is processed by MFS, the DPM process might create or override any of the output ATTACH parameters. The input RDPN and RPRN parameters become the output DPN and PRN parameters respectively for any resulting reply from the application program if they are not overridden by the DPM process.

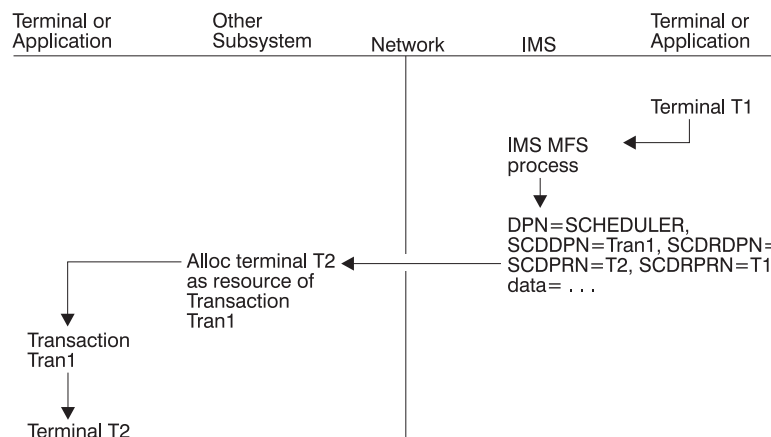
### Example 3: IMS MPP Is Accessed from Other Subsystem Terminal with Reply to a Temporary Storage File of the Other Subsystem

## ATTACH and SCHEDULER Parameters with IMS MFS



**Note:** The IMS transaction code can be supplied as the input PRN, as a data field, or as an MFS-defined literal. The DPN parameter must be supplied to invoke IMS MFS DPM. If a reply is inserted by the MPP, the DPM process might override or create any of the output ATTACH parameters. The input RDPN and RPRN parameters become the output DPN and PRN parameters respectively for any resulting reply from the application program if they are not overridden by the DPM process.

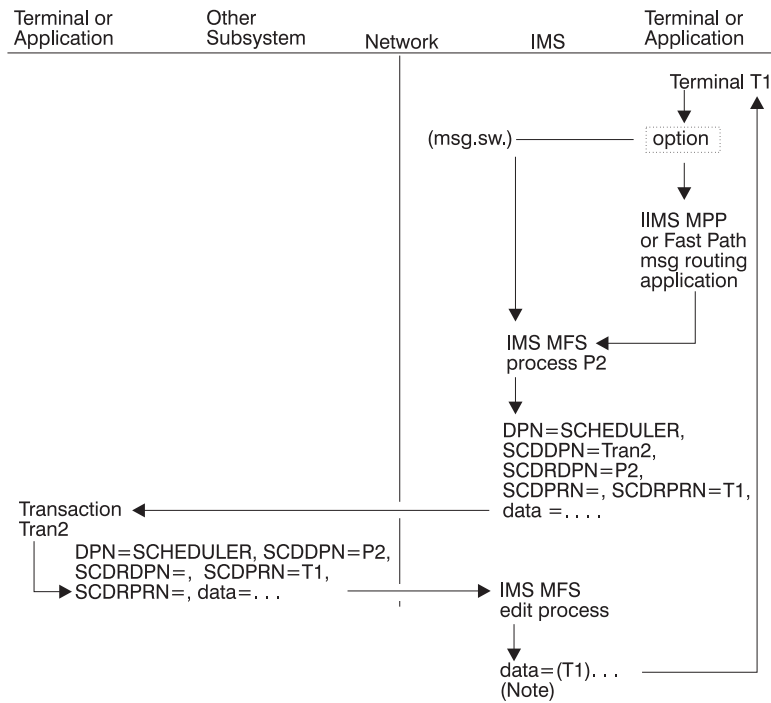
### Example 4: Message Switch from an IMS Terminal to a Terminal on another Subsystem



MFS DPM is necessary to create the required output SCHEDULER parameters. The SCDDPN and SCDPRN can be entered as data from the terminal or application program. If a reply is returned to another IMS terminal or application program, the default SCDRPRN parameter can be overridden within the IMS output message's SCHEDULER FM header to specify the suggested IMS terminal or application to receive that reply. The reply is returned with this value in the SCDPRN field.

## ATTACH and SCHEDULER Parameters with IMS MFS

### Example 5: Message Routing



**Note:** The IMS LTERM name can be supplied as the input SCDPRN, as the first data field, or as an MFS-defined literal. The SCDDPN parameter must be supplied to invoke IMS MFS. MFS can override or create any of the output SCHEDULER parameters.

## ATTACH and SCHEDULER Parameters with IMS MFS



---

## Appendix G. How IMS and CICS Use the ISC Interface

This appendix describes some of the issues involved when designing and implementing an ISC network that contains both IMS and CICS nodes. Before reading this chapter, you should be familiar with the basic system design concepts of both IMS and CICS running in a VTAM environment. The CICS information provided here is for planning purposes.

**Related Reading:** For more information on the design and coding of a CICS application that communicates with IMS via LU 6.1 ISC, see *CICS/ESA 4.1 Intercommunication Guide*.

### **In this Appendix:**

- “Functions Available to the ISC Session”
- “General Flow of CICS EXEC Commands within a CICS Application” on page 500
- “Coding Asynchronous Messages” on page 504
- “Commands That Should Not Be Used on an ISC Session” on page 507
- “Selecting Appropriate CICS Installation Options for ISC” on page 507
- “Coding CICS System Generation Options” on page 508
- “Preparing CICS Resource Definition” on page 508
- “Defining IMS-CICS LU 6.1 Links” on page 508
- “Defining Compatible IMS and CICS Nodes” on page 509
- “Defining Multiple Links to an IMS System” on page 513
- “Defining CICS Transactions for IMS-CICS ISC” on page 515
- “Understanding Application-Related Concepts” on page 517
- “Coding Function Management Headers for CICS” on page 525
- “Recovery and Restart Concepts” on page 531
- “Handling Transaction Abends” on page 536
- “Coding CICS Applications for Restart” on page 537

---

## Functions Available to the ISC Session

### **Definitions:**

- In *Synchronous mode* in an ISC session, the session is held between the moment a request is entered and the moment a reply is returned. Thus, a direct correlation can be made between the input request and the output reply.
- In *asynchronous mode* in an ISC session, no correlation can be made between an input request and an output reply. No assumptions can be made as to the timing of the output reply or the availability of any other output for that session.

**Related Reading:** For more information on these terms in the context of an ISC session, see “Relationship of ISC and IMS Execution Modes” on page 238.

The special support for front-end/back-end system utilization provided by the Front-End Switch exit routine uses ISC in asynchronous mode.

**Related Reading:** For information on the Front-End Switch exit routine (DFSFEBJ0), see *IMS/ESA Customization Guide*.

## Functions Available to the ISC Session

CICS implements ISC using its Command Level (EXEC) application programming interfaces (APIs). CICS has two VTAM APIs: synchronous and asynchronous. The synchronous API uses the SEND/RECEIVE EXEC commands; the asynchronous API uses the START/RETRIEVE EXEC commands. When used within the context of a CICS application program, the EXEC commands create the SNA data flow control protocols for synchronous or asynchronous processing that are associated with messages on the session.

The CICS implementation is quite different from IMS, in which the data flow control protocols sent with a message are created by the system and based on:

- Whether the attributes (such as recoverability and segmentation) are defined for the message on the TRANSACT, TERMINAL, and SUBPOOL macro statements during IMS system definition or on an ETO logon descriptor
- Whether the characteristics are defined in the bind parameters
- Whether the message is a reply that is returned on the same session as an associated request or on a different session than the associated request
- Whether the message is unsolicited asynchronous output

**Related Reading:** For more information on these factors, see “Relationship of ISC and IMS Execution Modes” on page 238.

This section identifies synchronous and asynchronous command sequences available to the CICS application.

CICS SEND/RECEIVE commands create the predefined SNA protocols associated with synchronous processing when SEND is used with the INVITE option. SEND INVITE causes an SNA ATTACH function management header and change-direction to be issued with the message. The message is sent from CICS to the remote IMS subsystem for processing. When the message reply is returned, it is processed by a RECEIVE command issued within the same CICS transaction that issued the first message.

**Exception:** Restarted transactions are handled differently; for information, see “Recovery and Restart Concepts” on page 531.

When the SEND command is used with the LAST option, it causes the message to be sent on the session with an ATTACH function management header and carries both begin-bracket and end-bracket (BB/EB) status. IMS treats this special case of synchronous processing in the same way as an IMS asynchronous transaction is treated. If IMS generates an output reply as a result of an input generated with SEND LAST, the returned reply can be processed by a newly initiated CICS transaction. The IMS output reply is also sent with ATTACH BB/EB. CICS uses RECEIVE to process that reply.

The START/RETRIEVE commands create the SNA protocols associated with asynchronous processing. A START command causes an SNA ATTACH function management header and a concatenated SCHEDULER function management header to be issued with the message. Messages sent to IMS with the START command can be only one chain in length and are sent with BB/EB. CICS uses the RETRIEVE command to obtain asynchronous messages from the session for CICS transaction processing. The other SNA data flow control protocols issued with an asynchronous message depend upon the type of message issued. When an asynchronous message is sent from CICS to IMS, the sending transaction might terminate. The returned reply can be processed by a newly initiated CICS transaction according to parameters in the function management headers returned

## Functions Available to the ISC Session

on the message by IMS. Figure 68 illustrates the concept of SEND/RECEIVE processing.

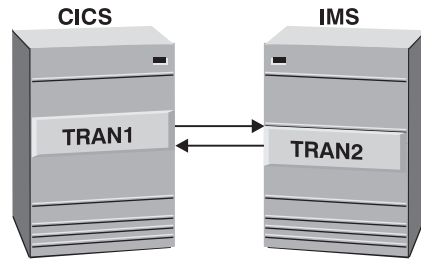


Figure 68. SEND/RECEIVE (Synchronous) Processing

Figure 69 illustrates the concept of START/RETRIEVE processing.

Table 52 summarizes the functions available to a session between IMS and CICS.

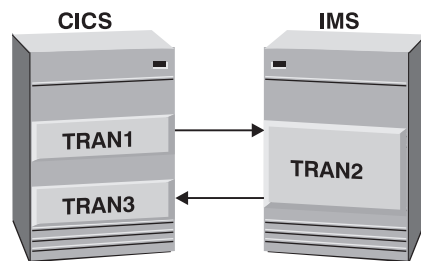


Figure 69. START/RETRIEVE (Asynchronous) Processing

You must know whether the flow on a session is to be synchronous or asynchronous in order to determine what functions are available to that session.

Table 52 does not take into consideration the recovery aspects of any of the listed combinations. A “Yes” indicates only that the approach is possible. However, a system designer should take into account the need for ease of recovery and restart.

Table 52. Functions Available to an IMS-CICS Session for CICS EXEC Commands

	SEND(INVITE)/ RECEIVE (Synchronous)	SEND(LAST) (Synchronous)	START/ RETRIEVE <sup>1</sup> (Asynchronous)
<b>CICS Front End</b>			
IMS Transaction Type:			
Nonresp./nonconvers. mode:			
Input	No	Yes	Yes
Output	Yes <sup>2</sup>	N/A	Yes
Response mode (including Fast Path)			
	Yes <sup>3</sup>	No	No
Conversation mode:			
Nonlast input	Yes	No <sup>4</sup>	No
Last conversation input	No	No <sup>4</sup>	No
IMS message switch	No	Yes	Yes
IMS operator command	Yes <sup>5</sup>	Yes <sup>6</sup>	Yes <sup>6</sup>
IMS transaction attributes:			
Recoverable <sup>7</sup>	N/A	N/A	N/A
Nonrecoverable <sup>7</sup>	N/A	N/A	N/A
Single-segment input/output	Yes	Yes	Yes
Multisegment input/output	Yes	Yes	No
<b>IMS Front End</b>			

## Functions Available to the ISC Session

Table 52. Functions Available to an IMS-CICS Session for CICS EXEC Commands (continued)

	SEND(INVITE)/ RECEIVE (Synchronous)	SEND(LAST) (Synchronous)	START/ RETRIEVE <sup>1</sup> (Asynchronous)
CICS transaction	No	No	Yes
CICS operator command (system transaction)	No	No	No
<b>MFS/BMS Support<sup>8</sup></b>			
Without IMS MFS	Yes	Yes	Yes
MFS without paging	Yes	Yes	Yes
MFS autopaged input	Yes <sup>9</sup>	Yes <sup>9</sup>	No
MFS autopaged output	Yes <sup>10</sup> .	N/A	No
MFS demand-paged output	2	N/A	No <sup>11</sup>
BMS support <sup>8</sup>	Yes	N/A	N/A
	N/A		

### Notes:

1. Single-chain messages only.
2. The nonresponse-mode reply to a transaction sent with SEND LAST is sent ATTACH BB/EB. CICS uses RECEIVE (without SEND) to obtain the message from the session. (See "CICS to IMS Using the SEND LAST EXEC Command" on page 502.)
3. IMS forces response mode regardless of system definition if a transaction is specified externally (using the function management header) as synchronous. See "Relationship of ISC and IMS Execution Modes" on page 238.
4. IMS conversational mode requires that IMS terminate the conversation by sending the last reply with EB. The exception to this is explained in "CICS versus IMS Conversation Mode" on page 520.
5. The replies that result from the IMS /DISPLAY, /FORMAT, and /RDISPLAY commands are sent asynchronously after IMS replies synchronously to the input command.
6. Supported only for /DISPLAY, /RDISPLAY, and /FORMAT.
7. A transaction is defined as recoverable or nonrecoverable during IMS system definition and is acceptable to IMS only when the sync point protocols and transaction definitions are consistent. See "Relationship of ISC and IMS Execution Modes" on page 238 and "Keeping Half Sessions Synchronized" on page 287 for more information.
8. MFS maps data between the session and the IMS application program. BMS maps data at the request of the CICS application program. BMS does not interact with the session, but can be used to handle mapping of data to and from IMS. IMS's MFS and CICS's BMS do not communicate.
9. Both single- and multiple-message chains can be used, but only one chain can occur per SEND.
10. To avoid tying up the session, when processing synchronous autopaged output from IMS, CICS should read all pages of the IMS output before processing it.
11. Although demand paging is possible between IMS and CICS, it is not recommended, because it requires complex CICS application coding and a complex terminal user interface. A preferable approach is to create autopaged output from IMS to CICS and use CICS page retrieval in the local system for paging.

## General Flow of CICS EXEC Commands within a CICS Application

The design of a CICS application program using the CICS EXEC command level application programming interfaces is determined by whether the transaction being sent on the ISC session is to be processed using SEND/RECEIVE, SEND LAST, or START/RETRIEVE. The sequence of EXEC commands issued is determined by whether the transaction is defined as recoverable or nonrecoverable. Program design can also depend on whether CICS is the front-end system (initiating a transaction) or back-end system (replying to an IMS transaction).

This section presents an overview of synchronous and asynchronous transaction processing flow within CICS. As such, it supplements the overview information

## Flow of CICS EXEC Commands within a CICS Application

provided in “Functions Available to the ISC Session” on page 497. Understanding the content and functions of the ATTACH and SCHEDULER FM headers is helpful in understanding this section.

**Related Reading:** For information on the ATTACH and SCHEDULER FM headers, see “Coding Function Management Headers for CICS” on page 525.

## CICS to IMS Using SEND/RECEIVE EXEC Commands

In a system in which CICS is the front-end subsystem and IMS the back-end subsystem, the SEND/RECEIVE commands are used to process IMS response mode (including Fast Path) and conversational transactions, and IMS commands.

The general CICS application program flow for a synchronous message from CICS to IMS is shown in Figure 70.

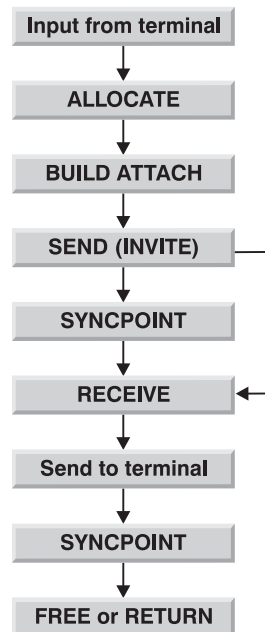


Figure 70. Application Program Flow Using SEND/RECEIVE from CICS to IMS

In this example, the CICS application reads the input message from the terminal and establishes a CICS-to-IMS session using the ALLOCATE command.

The CICS application program builds the required ATTACH function management header (the only header required, because this is a synchronous application) by using the BUILD ATTACH EXEC command. The RPROCESS and RRESOURCE fields should be specified in the BUILD ATTACH to provide for restart. Issuing SEND causes the output message to be constructed. SEND INVITE causes change-direction (CD) to be appended to the output message.

**Related Reading:** For information on using the BUILD ATTACH command and its parameters, see “Coding Function Management Headers for CICS” on page 525 and “Recovery and Restart Concepts” on page 531.

The output message is not sent across the session until the next command is executed. The next EXEC command that is issued depends upon the definition of the transaction within IMS, because this command must both append the sync-point

## Flow of CICS EXEC Commands within a CICS Application

request to the message as required and send the message across the ISC session. If this transaction is defined to IMS as recoverable, the next command must be SYNCPOINT. SYNCPOINT causes the outbound message to be sent with an RQD2. If the transaction is defined as nonrecoverable, a SYNCPOINT or a RECEIVE command can be issued next. A RECEIVE command immediately following a SEND command (without an intervening SYNCPOINT command) causes the outbound message to be issued with RQE1.

RECEIVE issued after either the SYNCPOINT or SEND commands reads the IMS reply. The reply from IMS is returned synchronously on the session (with the ATTACH FM header). If the transaction reply is to a response mode (including Fast Path) transaction, a command (except /DISPLAY, /RDISPLAY, and /FORMAT), or the last conversational reply, it is returned to CICS carrying RQD2 EB if recoverable, or RQD1 EB if nonrecoverable. If the transaction is a nonlast conversational reply, it is returned to CICS carrying RQE2 and CD (because the conversation continues).

If the transaction is defined within IMS as a nonrecoverable-inquiry, the CONVERSE command can be used. The CONVERSE command acts as if a SEND command were issued, immediately followed by a RECEIVE command. Transactions sent with a CONVERSE command carry BB/CD and a request for an exception response (RQE1). If transactions other than those defined as nonrecoverable are sent using CONVERSE, an error results.

After issuing RECEIVE, the application program must save and check the values in the EXEC interface block (EIB) before performing any additional processing. If the transaction reply is received successfully as determined by checking the EIB, the application now issues a sync point (SYNCPOINT). Issuing this sync point causes a DR2 response to be returned to the IMS transaction reply. If CICS wants to perform any application processing based upon the contents of the IMS reply message, including sending an output message to the terminal, that processing is performed before the sync point is issued. This ensures that CICS resources and IMS resources are committed in synchronism.

If the transaction completes successfully, the application program must free the session using either a FREE or RETURN EXEC command.

## CICS to IMS Using the SEND LAST EXEC Command

In a system in which CICS is the front-end subsystem and IMS the back-end subsystem, IMS supports the use of the CICS synchronous API to allow access to IMS asynchronous transactions. SEND LAST is used to process IMS nonresponse mode and nonconversational transactions, message switches, and the /DISPLAY, /RDISPLAY and /FORMAT commands.

The general CICS application program flow for CICS to IMS using SEND LAST is shown in Figure 71.

## Flow of CICS EXEC Commands within a CICS Application

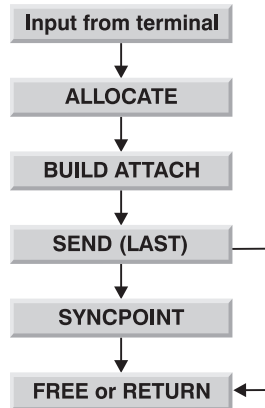


Figure 71. Application Program Flow Using SEND LAST from CICS to IMS

In this example, the CICS application reads the input message from the terminal and establishes a CICS-to-IMS session using the ALLOCATE command.

The CICS application program builds the required ATTACH function management header (the only header required, because this is a synchronous application) by using the BUILD ATTACH EXEC command. The RPROCESS and RRESOURCE fields should be specified in the BUILD ATTACH to identify the terminal and transaction to be used to process the reply. Issuing a SEND command causes the output message to be constructed. A SEND LAST command causes end-bracket to be appended to the message.

**Related Reading:** For information on using the BUILD ATTACH command and its parameters, see “Coding Function Management Headers for CICS” on page 525.

The output message is not sent on the session until the next command is executed. The next EXEC command that is issued depends upon the definition of the transaction within IMS, because this command both appends the sync-point request to the message and issues the message on the ISC session. If this transaction is defined to IMS as recoverable, the next command must be SYNCPOINT. SYNCPOINT causes the outbound message to be issued with an RQD2. If the transaction is defined as nonrecoverable, either SYNCPOINT, FREE, or RETURN can be issued next. FREE or RETURN issued immediately after the SEND and without an intervening SYNCPOINT causes the outbound message to be issued with RQE1.

If IMS generates a reply message as a result of receiving this transaction, that reply from IMS is returned with the ATTACH FM header. In order to get the IMS reply message from the session, CICS must issue a RECEIVE (because the reply message carries only ATTACH).

## IMS to CICS Using the RECEIVE EXEC Command

When IMS has a reply to send to CICS that results from a message sent by CICS with SEND LAST (ATTACH BB/EB), IMS sends that message using the ATTACH function management header and BB/EB. In order to receive the message from the session, CICS must initiate a new transaction (or a new instance of the original transaction) that uses the RECEIVE EXEC command for this purpose. The general CICS application program flow for this is shown in Figure 72 on page 504.

## Flow of CICS EXEC Commands within a CICS Application

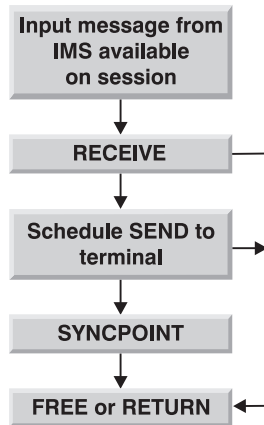


Figure 72. Application Program Flow Using RECEIVE from IMS to CICS

The reply from IMS is returned synchronously on the session (with the ATTACH FM header) and carries BB/EB. A recoverable reply carries RQD2; a nonrecoverable reply carries RQD1. The RPROCESS (RDPN) and RRESOURCE (RPRN) fields sent to IMS are automatically wrapped by IMS into the DPN and PRN fields of the outbound ATTACH FMH.

CICS examines the returned reply and uses the DPN, or the first four characters of the data, to determine the transaction that can process the returned reply. RECEIVE is used to obtain the IMS reply from the session. The PRN field is passed to the initiated transaction, indicating the terminal to which the returned reply is to be sent.

After issuing RECEIVE, the application program must save and check the values in the EXEC interface block (EIB) before performing any additional processing.

CICS must schedule an asynchronous transaction to send the returned reply to the terminal by issuing a START command. The transaction that is initiated is brought up, and owns the terminal to which the reply is to be sent. This asynchronous transaction uses the START interface to communicate with the terminal, and all processing within this transaction occurs asynchronously to the transaction that initiated it. If the transaction that sends output to the terminal is scheduled successfully, the originating application now issues a sync point (SYNCPOINT), if required. Issuing this sync point causes an appropriate response to be returned to the IMS transaction reply. If the transaction is not scheduled successfully, issuing the sync point causes an appropriate exception response to be returned to the IMS transaction reply. If CICS wants to perform any application processing based upon the contents of the IMS reply message, that processing is performed before the sync point is issued. This ensures that CICS resources and IMS resources are committed in synchronism.

If the processing completes successfully, the application program must free the session using either a FREE or RETURN EXEC command.

---

## Coding Asynchronous Messages

When receiving asynchronous messages (sent with ATTACH and SCHEDULER function management headers) from IMS, CICS invokes a CICS-supplied “mirror transaction” to obtain these incoming messages from the session. The mirror transaction is CICS’s name for the SCHEDULER process. This mirror transaction examines the incoming data stream and schedules (using START) the transaction



## Coding Asynchronous Messages

that is to be initiated as a result of the incoming DPN or the first four characters of the data. The receiving CICS transaction is assumed to be one of the following:

- ISC edit (ISCE), the default DPN set by IMS
- The transaction whose identifier was placed in the DPN field of the ATTACH FM header by IMS's wrapping of the incoming RDPN (the RTRANSID specified on the CICS START)
- A transaction whose identifier has been supplied by an MFS MOD

CICS uses the PRN in the incoming message as the TERMID parameter of the START command so that the transaction is initiated and owns the CICS terminal (if applicable) to which this message should be written.

A SYNCPOINT command is now issued to return the appropriate response to IMS and to complete the scheduling of the asynchronous transaction. Following the sync point, a RETURN command is issued to terminate the transaction and free the session. The response returned to IMS causes the output asynchronous message to be dequeued from the IMS output queue.

## CICS to IMS Using the START/RETRIEVE EXEC Commands

CICS uses START/RETRIEVE commands to process IMS nonresponse mode and nonconversational transactions, message switches, and the /DISPLAY, /RDISPLAY, and /FORMAT commands. The general CICS application program flow for an asynchronous message from CICS to IMS is shown in Figure 73.

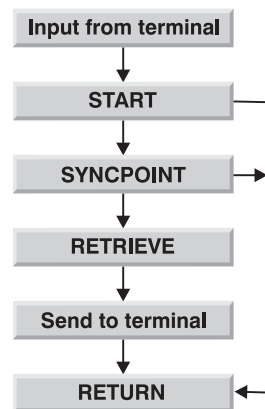


Figure 73. Application Program Flow Using START/RETRIEVE from CICS to IMS

After the input message is received from the terminal, the transaction is assembled using the START command. The values to be contained in the ATTACH and SCHEDULER function management header DPN, PRN, RDPN, and RPRN fields are specified as parameters on the START command. The DPN and PRN are used by IMS to determine the editing process to be initiated and the destination of the incoming message. The RDPN and RPRN are used to identify the CICS terminal and transaction to which the reply is to be returned. The START command must also specify NOCHECK if this is an IMS nonrecoverable transaction, and NOCHECK PROTECT if this is an IMS recoverable transaction. (PROTECT can optionally be used for nonrecoverable transactions.)

NOCHECK is a mandatory parameter for an ISC session with IMS. In a CICS-CICS ISC session, the use of START causes a return code reply to be sent to the initiating CICS from the receiving CICS. This return code indicates that the receiving CICS

## Coding Asynchronous Messages

has scheduled the transaction to be executed as a result of the message it has received. This return code does not occur on an IMS-CICS session. NOCHECK informs the sending CICS that no such response is to be expected. START NOCHECK creates a message carrying BB/EB RQE1.

PROTECT, specified for recoverable transactions, delays the sending of the transaction on the session until CICS successfully takes a SYNCPOINT. Specifying START causes begin- and end-bracket (BB/EB) and the ATTACH and SCHEDULER function management headers to be appended to the outbound message. NOCHECK PROTECT requests a definite response (RQD2).

A SYNCPOINT command is now issued to append the appropriate sync point protocols and to send the message on the session. If this transaction elects to wait for a reply, it issues a RETRIEVE command. If this transaction does not choose to wait for a reply, it can terminate by issuing RETURN following either the START command or the SYNCPOINT command. The CICS mirror transaction schedules a new instance of this transaction when a reply is received.

If the transaction chooses to wait for the reply, it can issue RETRIEVE with or without the WAIT parameter. Issuing RETRIEVE without the WAIT parameter causes CICS to check for any queued asynchronous input for this transaction and terminal. If such input is immediately available, it satisfies the RETRIEVE command. If no such input is immediately available, an appropriate indication is returned to the RETRIEVE command, and the application can then perform other processing or issue RETURN to terminate.

If RETRIEVE WAIT is issued, CICS does not return control to the application program until an asynchronous message is sent from IMS destined for this transaction and this terminal. Using WAIT causes the terminal be held until an IMS reply is received for it.

In both cases (RETRIEVE with or without WAIT), when a message is sent from CICS to IMS requesting asynchronous processing, no assumptions can be made by the originating application as to the timing of the output reply or the availability of any other output. That is, if unsolicited asynchronous output is pending for CICS, this output can be sent to CICS before the reply to this asynchronous transaction is returned. Therefore, requests and replies are not correlated within CICS in an asynchronous environment. Issuing RETRIEVE obtains any reply from IMS on any session that has the indicated transaction and terminal identification.

## IMS to CICS Using the RETRIEVE EXEC Command

Figure 74 illustrates the general CICS application program flow in asynchronous mode from IMS to CICS.

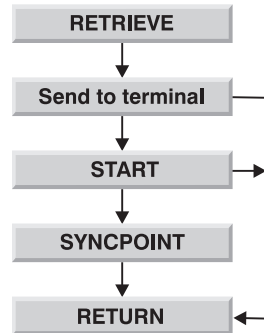


Figure 74. Application Program Flow Using RETRIEVE from IMS to CICS

Replies resulting from previous asynchronous input transactions and unsolicited asynchronous output from an IMS front-end to a CICS back-end are sent to CICS asynchronously (that is, with the ATTACH and SCHEDULER FM headers). When IMS is the front-end subsystem, asynchronous mode is the only flow supported. The CICS mirror transaction obtains the input, analyzes the incoming message, and schedules (using START) the transaction that is to be initiated as a result of the incoming DPN or the first four characters of the data. If this transaction is sending output to a terminal, the mirror transaction issues the START with the TERMID parameter (from the PRN parameter of the incoming FMH), so that the asynchronous transaction that is scheduled owns the terminal to which output is to be sent.

If a reply to IMS is required, the CICS application must wrap the incoming RTERMID and RTRANSID into the TERMID and TRANSID fields to be used in a subsequent START. These fields indicate, respectively, the destination LTERM or transaction within IMS and the IMS editor or MFS MID that is to receive the message. The START command to send this reply to IMS is processed.

#### **Related Reading:**

- For an explanation of the content of the TERMID and TRANSID fields, see “Coding Function Management Headers for CICS” on page 525.
- For information on the processing of the START command to send the reply to IMS, see “CICS to IMS Using the START/RETRIEVE EXEC Commands” on page 505.

If no reply to IMS is required, RETURN can be issued to terminate the session.

---

## **Commands That Should Not Be Used on an ISC Session**

Do not use the following commands on an ISC session:

- WAIT SIGNAL  
The WAIT SIGNAL command cannot be used in a CICS-IMS session.
- WAIT TERMINAL  
The WAIT TERMINAL command is not normally used in a CICS-IMS ISC session.

---

## **Selecting Appropriate CICS Installation Options for ISC**

This section describes only those system generation options and resource definition options that have unique considerations for an IMS-CICS ISC environment.

## Selecting CICS Installation Options for ISC

Some of the parameters coded during the CICS installation process must be compatible with some of the IMS system definition parameters coded on the IMS system definition macros. The section “Defining Compatible IMS and CICS Nodes” on page 509 brings together the definitions provided here with the IMS system definition information provided in “Statically Defining an ISC Node to IMS” on page 257.

---

## Coding CICS System Generation Options

The CICS standard pregenerated system as supplied on the distribution tape includes a pregenerated version for each of the CICS management modules required to support an Intersystem Communication environment that includes all of the required system generation options.

### **Related Reading:**

- For information on system generation, see *CICS/ESA 4.1 Intercommunication Guide* for the specifications necessary to generate a CICS subsystem that participates in IMS-CICS Intersystem Communication.
- For a checklist of recovery options, see *CICS/ESA 4.1 Recovery and Restart Guide*.

---

## Preparing CICS Resource Definition

CICS user-created tables and Resource Definition Online (RDO) describe the database and data communications environment, and the treatment of the elements in that environment. They contain information about terminals, files, databases, programs, transactions, transient data destinations, and temporary storage data identifiers. They are created independently of system generation, but must be present for the system to be operational. The CICS System Log and Dynamic Log, which are optional in some CICS environments, are required for IMS-CICS ISC.

### **Related Reading:**

- For more information on table preparation and RDO options, see *CICS/ESA 4.1 Resource Definition Guide*.
- For a summary of the parameters required for IMS-CICS Intersystem Communication, see *CICS/ESA 4.1 Intercommunication Guide*.
- For a checklist of recovery-related options, see *CICS/ESA 4.1 Recovery and Restart Guide*.

---

## Defining IMS-CICS LU 6.1 Links

A CICS link to an IMS system requires a definition of the connection (or system) and a separate definition of each of the sessions. Resource Definition Online or Macro-Level Resource Definition can be used to define an ISC link to IMS. IMS-CICS ISC links are implemented with LU 6.1 protocol.

Table 53 on page 509 shows the RDO form of definition for individual LU 6.1 sessions, and shows the macro form for how the operands are related.

The TRMTYPE, TRMIDNT, SYSIDNT, NETNAMQ, and SESTYPE operands must be coded for each session that you define. The remaining operands of TYPE=TERMINAL can optionally be coded on the TYPE=SYSTEM macro to provide defaults for all the defined sessions. Also, the CONNECT, DATASTR, and

## Defining IMS-CICS LU 6.1 Links

RECFM operands of TYPE=SYSTEM can be coded for individual sessions, if required.

Table 53. Defining an LU 6.1 Link with Individual Sessions

RDO Definition	Macro-Level Definition
DEFINE CONNECTION(sysidnt) GROUP(groupname) NETNAME(name) ACCESSMETHOD(VTAM) PROTOCOL(LU61) DATASTREAM(USER 3270  SCS STRFIELD  LMS) RECORDFORMAT(U VB) AUTOCONNECT(NO YES) SECURITYNAME(name) INSERVICE(YES)	DFHTCT TYPE=SYSTEM ,SYSIDNT=sysidnt) ,NETNAME=name ,ACCMETH=VTAM) ,DATASTR=({USER 3270  SCS STRFIELD  LMS}) ,RECFM={U VB} [,CONNECT=AUTO ALL] ,XSNAME=name

Each individual session is then defined as follows:

DEFINE SESSIONS(csdname) GROUP(groupname) SESSNAME(name) CONNECTION(sysidnt) NETNAMEQ(name) PROTOCOL(LU61) SENDCOUNT(0 1) RECEIVECOUNT(1 0) SENDSIZE(size) RECEIVESIZE(size) BUILDCHAIN(N Y) OPERID(operator-id) OPERPRIORITY(number) OPERRSL(number) OPERSECURITY(number) IOAREALEN(value) SESSPRIORITY(number)	DFHTCT TYPE=TERMINAL ,TERMIDNT=name ,SYSIDNT=sysidnt ,NETNAMQ=name ,TRMTYPE=LUTYPE6 ,SESTYPE= SEND RECEIVE ,BUFFER=size ,RUSIZE=size ,CHNASSY={NO YES} ,OPERID=operator-id ,OPERPRI=number ,OPERRSL=number ,OPERSEC=number ,TIOAL=value ,TRMPRTY=number ,TRMSTAT=TRANSCIVE
---	---

## Defining Compatible IMS and CICS Nodes

To define IMS-CICS ISC links, you should understand the relationship between the way remote systems and sessions are defined in CICS and the way they are defined in IMS.

The relationships between CICS and IMS definitions are summarized in Figure 75 on page 512 (RDO) and in Figure 76 on page 513 (macro-level definition).

RDO terms are used in the following explanation of the compatibility requirements. See Table 53 for the equivalent macro-level operands.

## System Names

The network name of the CICS system is specified in the APPLID operand of the DFHSIT macro. (It can be provided as an override during CICS startup or in the APPLID operand of the DFHTCT TYPE=INITIAL macro.) This name must be specified in the NAME operand of the TERMINAL macro or on the ETO logon descriptor that defines the CICS system. This name must be known to the IMS

## Defining Compatible IMS and CICS Nodes

master terminal operator as it is required to be specified within those IMS master terminal operator commands (for example, /OPNDST) that reference sessions between IMS and CICS.

The network name of the IMS system is specified in the APPLID operand of the IMS COMM macro. You must specify this name in the NETNAME operand of the DEFINE CONNECTION command that defines the IMS system.

## Number of Sessions

For statically defined terminals in IMS, the number of parallel sessions that are required between the CICS and IMS system must be specified in the SESSION operand of the IMS TERMINAL macro. Each session is then represented by a SUBPOOL entry in the IMS VTAMPOOL. In CICS, each of these sessions is represented by an individual session definition.

## Session Names

**Definition:** Each CICS-IMS session is uniquely identified by a *session-qualifier pair*, formed from both the CICS and IMS session names.

The CICS session name is specified in the SESSNAME operand of the DEFINE SESSIONS command. For sessions that are to be initiated by IMS, this name must correspond to the ID parameter of the IMS /OPNDST command for the session. For sessions initiated by CICS, the name is supplied on the CICS /OPNDST command and is saved by IMS.

The IMS session name is specified in the NAME operand of the IMS SUBPOOL macro. You must make the relationship between the session names explicit by coding this name in the NETNAMEQ operand of the corresponding DEFINE SESSIONS command.

**Recommendation:** For operational convenience, use the same CICS and IMS names for a session.

## Other Session Parameters

This section lists the remaining operands of the DEFINE CONNECTION and DEFINE SESSIONS commands that are of significance for CICS-IMS sessions.

### SENDSIZE

Specifies the maximum request unit (RU) size that the remote IMS system can receive. The equivalent IMS value is specified in the RECANY parameter of the IMS COMM macro. You must specify a size that is:

- Not less than 256 bytes
- At least 22 bytes less than the value in the RECANY parameter.

### BUILDCHAIN(N|Y)

Specifies whether multiple RU chains are to be assembled before being passed to the application program. If Y is specified, a complete chain is passed to the application program in response to each RECEIVE command, and the application must perform any required deblocking. If N is specified, a single RU is passed to the application program in response to each RECEIVE command.

**Recommendation:** BUILDCHAIN(Y) is recommended, even when IMS record mode (VLVB) is used, because the logical records produced as IMS output might not coincide with RU boundaries.

### DATASTREAM(USER)

Specifies a data stream profile when CICS is communicating with IMS using the START command (asynchronous processing). CICS messages generated by the START command always cause IMS to interpret the data stream profile as input for component 1. This parameter is required.

**Related Reading:** For more information on component determination, see “LTERM Users (Subpools) and Components” on page 240.

The data stream profile for distributed transaction processing can be specified by the application program using the DATASTR option of the BUILD ATTACH command.

### RECORDFORMAT(U|VB)

Specifies the type of chaining that CICS is to use for transmissions that are initiated by START commands (asynchronous processing) on a particular session.

Two types of data handling algorithms are supported between CICS and IMS:

- Chained

Messages are sent as SNA chains. The user can use private blocking and deblocking algorithms. This format corresponds to RECORDFORMAT(U). When IMS is the receiver, a chain of RUs is interpreted as the complete single-segment message.

- Variable length variable blocked records (VLVB)

Messages are sent in variable-length variable-blocked format with a halfword length field before each record. This format corresponds to RECORDFORMAT(VB). When IMS is the receiver, the input deblocked element is treated as the input segment.

The data stream format for distributed transaction processing can be specified by the application program using the RECFM option of the BUILD ATTACH command.

### SENDCOUNT and RECEIVECOUNT

These operands are used to specify whether the session is a SEND session or a RECEIVE session. (In macro-level definition, this is specified in the SESTYPE=SEND|RECEIVE operand.)

A SEND session is one in which the local CICS is secondary and is the contention winner. It is specified by:

```
SENDCOUNT(1)  
RECEIVECOUNT(0)
```

A RECEIVE session is one in which the local CICS is primary and is the contention loser. It is specified by:

```
SENDCOUNT(0)  
RECEIVECOUNT(1)
```

**Recommendation:** SEND sessions are recommended for all CICS-IMS sessions.

You need not specify a SENDPFX or a RECEIVEPFX; the name of the session is taken from the SESSNAME operand.

Figure 75 on page 512 shows the relationship between the CICS and IMS definitions of an intersystem link. Related operands are shown by the numbered

## Defining Compatible IMS and CICS Nodes

paths, all of which pass through the central connecting line. If CICS is communicating with an XRF IMS, NETNAME(SYSIMS) should be the USERVAR associated with the XRF IMS.

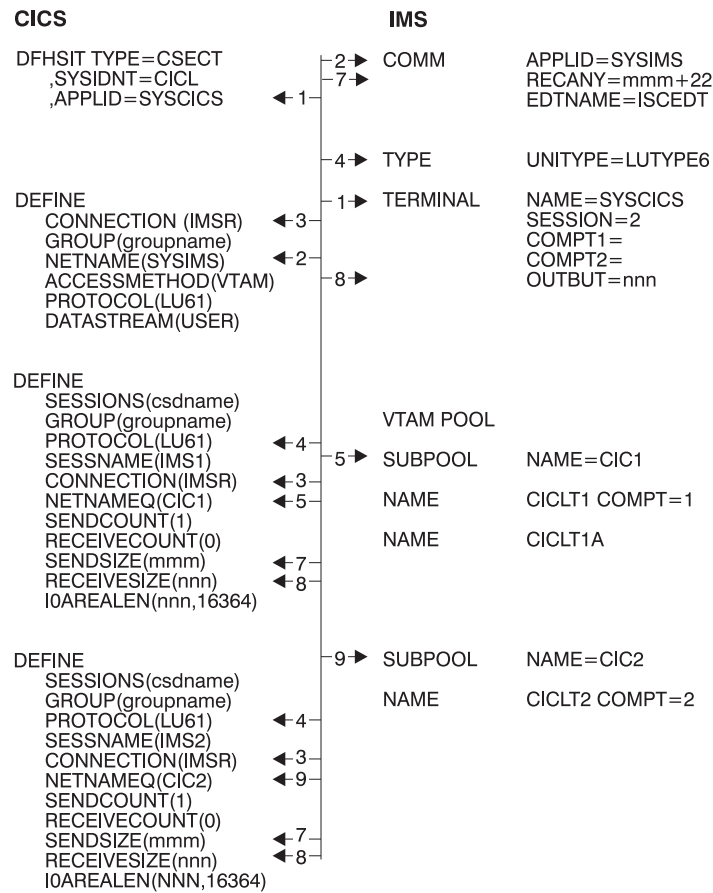


Figure 75. Defining Compatible CICS and IMS Nodes - RDO

Figure 76 on page 513 shows the relationship between the CICS and IMS definitions of an intersystem link. Related operands are shown by the numbered paths, all of which pass through the central connecting line. NETNAME=SYSIMS should be the USERVAR associated with the IMS system if you are defining a CICS to XRF IMS session.



## Defining Multiple Links to an IMS System

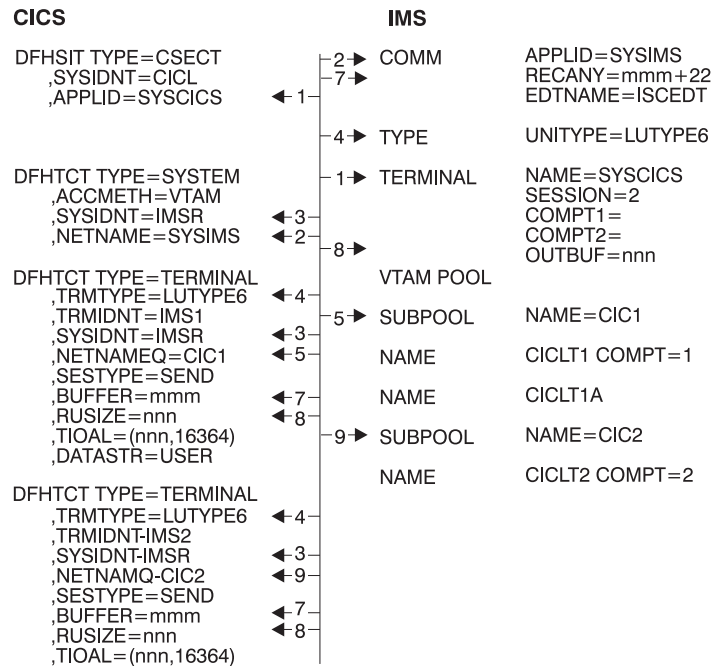


Figure 76. Defining Compatible CICS and IMS Nodes - Macro Level

## Defining Multiple Links to an IMS System

You can define more than one intersystem link between a CICS and an IMS system by defining two or more connections (systems), with their associated session definitions, having the same NETNAME but different SYSIDs (see Table 54). Although all the system definitions resolve to the same NETNAME, and therefore to the same IMS system, using a SYSID name in CICS causes CICS to allocate a session from the link with the specified SYSIDNT.

Table 54. Defining Multiple Links to an IMS Node

RDO Definition	Macro-Level Definition
DFHSIT TYPE=CSECT ,SYSIDNT=CICL ,APPLID=SYSCICS	DFHSIT TYPE=CSECT ,SYSIDNT=CICL ,APPLID=SYSCICS
<b>CICS-Initiated Distributed Transaction Processing</b>	
DEFINE CONNECTION(IMSA) ACCESSMETHOD(VTAM) NETNAME(SYSIMS)	DFHTCT TYPE=SYSTEM ,ACCMETH=VTAM ,SYSIDNT=IMSA ,NETNAME=SYSIMS
DEFINE SESSIONS(csdname) PROTOCOL(LU61) SESSNAME(IMS1) CONNECTION(IMSA) NETNAMEQ(DTP1)	DFHTCT TYPE=TERMINAL ,TRMIDNT=IMS1 ,SYSIDNT=IMSA ,NETNAMQ=DTP1
DEFINE SESSIONS(csdname)	DFHTCT TYPE=TERMINAL
⋮	⋮
<b>CICS-Initiated Asynchronous Processing</b>	

## Defining Multiple Links to an IMS System

Table 54. Defining Multiple Links to an IMS Node (continued)

RDO Definition	Macro-Level Definition
<pre>DEFINE CONNECTION(IMSB)       ACCESSMETHOD(VTAM)       NETNAME(SYSIMS)</pre>	<pre>DFHTCT TYPE=SYSTEM       ,ACCMETH=VTAM       ,SYSIDNT=IMSB       ,NETNAME=SYSIMS</pre>
<pre>DEFINE SESSIONS(csdname)       PROTOCOL(LU61)       SESSNAME(IMS1)       CONNECTION(IMSB)       NETNAMEQ(ASP1)</pre>	<pre>DFHTCT TYPE=TERMINAL       ,TRMTYPE=LUTYPE6       ,TRMIDNT=IMS1       ,SYSIDNT=IMSB       ,NETNAMQ=ASP1</pre>
<pre>DEFINE SESSIONS(csdname) : :</pre>	<pre>DFHTCT TYPE=TERMINAL : :</pre>
<b>IMS-Initiated Asynchronous Processing</b>	
<pre>DEFINE CONNECTION(IMSC)       ACCESSMETHOD(VTAM)       NETNAME(SYSIMS)</pre>	<pre>DFHTCT TYPE=SYSTEM       ,ACCMETH=VTAM       ,SYSIDNT=IMSC       ,NETNAME=SYSIMS</pre>
<pre>DEFINE SESSIONS(csdname)       PROTOCOL(LU61)       SESSNAME(IMS1)       CONNECTION(IMSC)       NETNAMEQ(IST1)</pre>	<pre>DFHTCT TYPE=TERMINAL       ,TRMTYPE=LUTYPE6       ,TRMIDNT=IMS1       ,SYSIDNT=IMSC       ,NETNAMQ=IST1</pre>
<pre>DEFINE SESSIONS(csdname) : :</pre>	<pre>DFHTCT TYPE=TERMINAL : :</pre>

**Recommendation:** Define up to three links (that is, groups of sessions) between a CICS and an IMS system, depending upon the application requirements of your installation:

1. A group of sessions for CICS-initiated distributed transaction processing (synchronous processing).  
CICS applications that use the SEND/RECEIVE interface can use the SYSIDNT of this group to allocate a session to the remote system. The session is held ("busy") until the conversation is terminated.
2. A group of sessions for CICS-initiated asynchronous processing.  
CICS applications that use the START command can name the SYSIDNT of this group. CICS uses the first "nonbusy" session to ship the start request.  
IMS sends a positive response to CICS as soon as it has queued the start request, so that the session is in use for a relatively short period. Consequently, the first session in the group shows the heaviest usage, and the frequency of usage decreases towards the last session in the group.
3. A group of sessions for IMS-initiated asynchronous processing.  
This group is also useful as part of the solution to a performance problem that can arise with CICS-initiated asynchronous processing. An IMS transaction that is initiated as a result of a START command shipped on a particular session uses the same session to ship its "reply" START command to CICS. For the reasons given in (2) above, the CICS START command is probably shipped on the busiest session, and, because CICS is the contention winner, the replies from IMS can create a backlog while waiting for a chance to use the session.

However, facilities exist in IMS for a transaction to alter its default output session, and a switch to a session in this third group can reduce backlog problems.

---

### Defining CICS Transactions for IMS-CICS ISC

This section describes the unique considerations for transactions that participate in IMS-CICS ISC.

**Related Reading:** For complete details on the definition of CICS transactions, see *CICS/ESA 4.1 Resource Definition Guide*.

### Defining CICS Backout In-Doubt Processing

During the period between the sending of the syncpoint request to IMS and the receipt of the positive response, CICS does not know whether the remote system has committed. This period is known as the in-doubt period. CICS processing during the in-doubt period is controlled by the DEFINE TRANSACTION IN-DOUBT parameter (or the DFHPCT DTB= parameter).

DEFINE TRANSACTION IN-DOUBT (WAIT) or DTB=(YES,WAIT) must be specified in order to ensure consistency between the IMS and CICS subsystems within an ISC network.

### Defining CICS Transactions for Asynchronous Communication to IMS

A DEFINE TRANSACTION REMOTENAME (or DFHPCT TYPE=REMOTE RMTNAME) is needed for every IMS edit name or MID name that is to be referenced by a transaction created as the result of a CICS START command. The 4-character CICS transaction ID used on the START command is converted to the name of the IMS editor or to an MFS MID name, each of which can be up to 8 characters.

### Initiating and Allocating a Session from CICS

CICS can initiate a session in one of the following ways:

- The session can be initiated explicitly by use of AUTOCONNECT YES on the DEFINE CONNECTION (or CONNECT=AUTO on the DFHTCT TYPE=TERMINAL) macro. When CICS is initiated, it attempts to establish all sessions for which AUTOCONNECT is specified. In order for session initiation to occur, IMS must be active when CICS is initiated.
- The master terminal operator can initiate a session by entering the command:  
`CEMT SET TERMINAL(ttt) ACQUIRED|COLDACQ`

where *ttt* is the SESSNAME on the DEFINE SESSIONS or the TRMIDNT on the DFHTCT TYPE=TERMINAL.

If ACQUIRED is specified, normal resynchronization with the IMS half session is attempted.

If COLDACQ is specified, no resynchronization is performed.

The status of the CICS half session is typed over in the display area of the CEMT INQUIRE|SET TERMINAL command. If the session is initiated successfully, the status is changed from REL to ACQ. If the attempt to initiate the session is unsuccessful, an error message is written to the transient data destination CSMT.

## Defining CICS Transactions for IMS-CICS ISC

- A session can be initiated implicitly by an application program using the ALLOCATE command. The ALLOCATE command is used on the SEND/RECEIVE interface only.

**Recommendation:** Use the ALLOCATE SYSID form of this command, because it allows CICS to select an available session.

If a session is not immediately available, control is returned to the application program in the following situations:

- A HANDLE CONDITION for this condition is issued by the application program.
- NOQUEUE is specified on the ALLOCATE command.

Otherwise, the command is queued.

The following conditions cause a session to be “not immediately available”:

- All sessions to the specified system (or the specified session) are in use.
  - The only available sessions are not bound.
  - The only available sessions are contention losers.
- A session can be initiated by CICS automatic task initiation (ATI).

The bind parameters for CICS are built from a hard-coded model. The RECEIVESIZE and SENDSIZE parameters on the DEFINE CONNECTION/SESSION (or the RUSIZE and BUFFER parameters on the DFHTCT TYPE=TERMINAL/SYSTEM) are used to determine the primary and secondary RU sizes.

**Related Reading:** For more information on the ALLOCATE command, and subsequent processing, see *CICS/ESA 4.1 Intercommunication Guide*.

## Other Ways of Initiating a Session

Sessions not initiated by CICS can be initiated in one of the following ways:

- By IMS
- By the VTAM operator, NCCF, or NetView

## Terminating a Session from CICS

Sessions can be terminated by CICS only by using master terminal operator commands. The master terminal operator can use the CEMT command to release the session by entering:

```
CEMT SET TERMINAL(ttt) [RELEASED|OUTSERVICE]
```

In the example above, *ttt* is the SESSNAME on the DEFINE SESSIONS or the TRMIDNT on the DFHTCT TYPE=TERMINAL.

If the master terminal operator specifies RELEASED, the session terminates when any active transaction completes and the session is in a between-brackets state.

If the master terminal operator specifies OUTSERVICE, the session also terminates when any active transactions complete and the session is in a between-brackets state. However, in this case, the session is taken out of service and cannot be used until the master terminal operator places it back into service.

If RELEASED is specified, it initiates an orderly termination between IMS and CICS. Although, from the CICS view, the session might appear to be in warm-start state, it is actually in cold-start state as a result of the SBI/BIS flow.

## Defining CICS Transactions for IMS-CICS ISC

**Related Reading:** For more information on session states resulting from these shutdown protocols, see “Symmetrical Session Shutdown for LU 6.1 (SBI and BIS)” on page 320.

If OUTSERVICE is specified, the session is left in a warm-start state; that is, CICS requests resynchronization upon reinitiation. In order to initiate a session in cold-start mode, the CICS master terminal operator must specify:

```
CEMT SET TERMINAL (tttt) COLDACQ.
```

You can specify RELEASED and OUTSERVICE together.

**Related Reading:** For more information on initiating a session, see “Initiating and Allocating a Session from CICS” on page 515.

**Recommendation:** Although orderly session termination can occur as a result of the CICS application program issuing EXEC CICS DISCONNECT, this approach is not recommended in a normal application. However, an operator-control type of application can be written to use this function.

Any messages relative to the session’s termination are sent to transient data destination CSMT.

## Designing CICS Applications for ISC

CICS differs from IMS in that many of the flows that must occur on an ISC session occur under the control of the CICS application program. The CICS application must do the following:

- Build outbound ATTACH function management headers
- Supply fields for SCHEDULER function management headers (if required)
- Examine inbound ATTACH function management headers and process their contents
- Issue sync points at appropriate intervals during the application program’s processing.

The SNA bracket, SEND/RECEIVE, and sync-point protocols associated with CICS-generated messages depend upon the EXEC commands<sup>41</sup> used by the application program, and when the sync points are issued during the application flow.

Information is inserted into the ATTACH and SCHEDULER function management header DPN, RDPN, PRN, and RPRN fields for message editing and routing by IMS system code, and can be overridden only by using the IMS Message Format Service (MFS). For CICS, insertion of information into the ATTACH and SCHEDULER function management headers is performed under the control of the CICS application.

---

## Understanding Application-Related Concepts

This section describes some application-related concepts to establish a common frame of reference for both CICS and IMS users.

---

41. The EXEC commands are SEND/RECEIVE (for synchronous) and START/RETRIEVE (for asynchronous).

## Understanding Application-Related Concepts

### Subsystem Design—Direct-Control versus Queued

CICS is a direct-control subsystem, while IMS is a queued subsystem. That is, CICS accepts data entered from a terminal, and as a result, invokes the appropriate application program to process that data. The terminal and other system resources are owned by the invoked application until the application task completes its processing. Information that results from processing is held in main storage rather than being queued. The implication for ISC is that the CICS application program is directly involved in generating the appropriate SNA data flow control, sync point, and response protocols, and in controlling most system services.

In contrast, IMS is a queued subsystem. In this case, all input and output transactions and message switches are queued by the IMS control region on behalf of the related IMS applications and terminals. Thus, the input and output of a message to or from the terminal are asynchronous from the processing of the message.<sup>42</sup> The implication for ISC is that the SNA protocols and many system services are handled under the control of IMS system code; the application program does not need to provide them.

### Synchronous and Asynchronous Processing

Message transmission can be synchronous or asynchronous between the terminal entering the message and the receiving subsystem. Messages can also be either synchronous or asynchronous with respect to the ISC session.

**Related Reading:** For more information on ISC message transmission, see “Relationship of ISC and IMS Execution Modes” on page 238.

From the point of view of the CICS application, using the CICS SEND/RECEIVE EXEC commands produces synchronous processing on the session, while the CICS START/RETRIEVE EXEC commands result in asynchronous processing. When CICS is the front-end subsystem, both processing types are supported. When IMS is the front-end subsystem, only asynchronous processing can occur, unless the special support for front-end/back-end system utilization provided by IMS Front-End Switch exit routine is used.

**Related Reading:** For more information on the Front-End Switch exit routine (DFSFEBJ0), see *IMS/ESA Customization Guide*.

Table 55 shows how the CICS START/RETRIEVE and SEND/RECEIVE EXEC commands are supported from a CICS point of view.

*Table 55. The SEND/RECEIVE and START/RETRIEVE Commands*

Message Type	CICS to IMS	IMS to CICS
SEND/RECEIVE (synchronous)	YES	YES
START/RETRIEVE (asynchronous)	YES	YES

Table 56 on page 519 shows the differences in the CICS application program interface when using SEND/RECEIVE or START/RETRIEVE.

42. However, the processing might appear to be synchronous to the terminal because of the way in which the message is defined to IMS; for example, a response-mode message or a conversation might appear to be synchronous.

## Understanding Application-Related Concepts

Table 56. CICS API for SEND/RECEIVE and START/RECEIVE

Attribute	SEND/RECEIVE	START/RETRIEVE
Mode	Synchronous with (SEND INVITE) or without (SEND LAST) reply	Asynchronous
FMH	ATTACH	ATTACH SCHEDULER
Name length	Supports 8-byte names	Supports 4-byte names <sup>1</sup>
Change Direction (CD)	Supported	CICS front-end cannot send CD. IMS sends CD to CICS if COMPT=SINGLE2 or MULT2.
Multiple chains per message (IUTYPE)	Supported <sup>2</sup>	Not supported
IMS component selection (DATASTREAM)	Supported	Not supported for input. Input component=1. Output component=any.
RECORDFORMAT: <sup>3</sup>		
Undefined (U)	Supported <sup>4</sup>	Not supported
RU	Supported <sup>4</sup>	Not supported
VLVB	Supported	Supported
CHAIN <sup>5</sup>	Supported	Supported
FMH built by	Application program (BUILD ATTACH)	System: DPN, PRN, RDPN, RPRN specified in EXEC START command <sup>6</sup>
Access to FMH	Yes (EXTRACT Attach)	Any supplied parameters available on EXEC RETRIEVE command <sup>6</sup>
Relation to ISC session	Directly related (ALLOCATED)	Independent
IMS transaction types supported	Response mode Conversation mode Message switches <sup>7</sup> Commands Nonresponse, nonconversational <sup>7</sup>	Nonresponse mode Nonconversation Message switches Commands <sup>8</sup>
Transaction Attributes:		
Recoverable I/O	Supported	Supported
Nonrecoverable I/O	Supported	Supported
Multisegment input	Supported	Single segment only
Multisegment output	Supported	Single segment only
IMS edits	Basic edit	Basic edit
Available	ISC Edit	ISC Edit
MFS:		
Without paging	Supported	Supported
Autopaged input:		
Single chain	Supported	Supported
Multichain	Supported	Not supported
Autopaged output	Supported	Not supported
Demand-paged output	Supported	Not recommended

## Understanding Application-Related Concepts

Table 56. CICS API for SEND/RECEIVE and START/RECEIVE (continued)

Attribute	SEND/RECEIVE	START/RETRIEVE
<b>Notes:</b>		
1. Names can be embedded in the message text. The DPN field can be converted from a 4-character to an 8-character name by using DEFINE TRANSACTION REMOTENAME (or DFHPCT TYPE=REMOTE).		
2. Only used to send or receive MFS autopaged input or output messages.		
3. The ATTACH parameters are defined in the CONNECTION definition. However, the application using START/RETRIEVE must understand and provide the correct data format within the application.		
4. Marked as reserved by CICS, but can be specified on the BUILD ATTACH. For more information on the ATTACH parameters, see "ATTACH FM Header Format" on page 339.		
5. Chained output from IMS requires the use of MFS.		
6. The RDPN and RPRN are limited to 4-character names.		
7. Nonresponse, nonconversational, and message switch input use SEND LAST only. CICS acquires replies to nonresponse and nonconversational transactions using RECEIVE.		
8. /DIS, /RDIS, /FOR only.		

## Principal and Alternate Facility

The functions of CICS synchronous EXEC terminal control commands (such as SEND, RECEIVE, CONVERSE) are exercised against a principal or an alternate facility. In CICS, the principal facility for a task is the terminal or ISC session that is made available to the application program when the task is initiated. The alternate facility is the ISC session that is allocated as needed by the application program. Commands issued against the principal facility are issued without the use of the SESSION (name) option. Commands issued against the alternate facility are issued with this option.

When CICS is the front-end subsystem, the user terminal is the principal facility. The session allocated to IMS is the alternate facility. However, when a restart transaction is attached from the ISC session, the session is the principal facility. The restart transaction does not have direct access to the user terminal.

**Related Reading:** For more information on restart and recovery processing within CICS, see "Recovery and Restart Concepts" on page 531.

## CICS versus IMS Conversation Mode

In IMS, conversation mode is an attribute of a transaction in which the conversation is carried on through the transfer of a scratch pad area (SPA) and an associated message between the terminal and transaction message queues. Each transfer of information to the transaction message queue and back to the terminal queue results in a sync point, and causes the commitment of resources and the release of locked resources. Also, all of the steps of an IMS conversation must occur entirely within a bracket. EB cannot be sent at any intermediate point in the conversation, because the receipt of EB causes IMS to discard the output message, terminate the conversation, and invoke the conversational abnormal termination exit. During a conversation, if a primary resource name (PRN) parameter is supplied in the input function management header, it is ignored, because the transaction code is carried in the SPA.

In CICS, a conversation is a series of interactions within a bracket between a terminal and an application. End-bracket occurs upon termination of the application.



## Understanding Application-Related Concepts

CICS also permits a “pseudo-conversation,” which is a sequence of transactions between a terminal and an application. Each transaction has one terminal input and one terminal reply within its own bracket. Either of these types of conversations can interact with IMS on the ISC session between the requests and replies by using the EXEC CONVERSE command. A message with end-bracket being sent to the terminal by CICS also causes an end-bracket message to be returned on the ISC session to IMS, if IMS has not previously sent an end-bracket status on the reply.

The differences in the way in which conversations are defined in CICS and IMS have the following implications for ISC:

- A terminal connected to CICS can only be held synchronously with the ISC session and IMS transaction processing if the CICS transaction is conversational in the CICS sense.
- If more than one input and output is to occur on the ISC session within a single bracket, the CICS transaction can interact with an IMS transaction that is defined as conversational. However, in the event of a session failure, the conversation might not be restartable.
- If only one input and output is to occur on the ISC session within a single bracket, the CICS transaction can interact with an IMS transaction defined as response mode. A series of transactions that make up a CICS pseudo-conversation can interact on an ISC session with a series of IMS transactions defined as response mode.

**Related Reading:** For more information on restart processing, see “Recovery and Restart Concepts” on page 531.

IMS conversational mode requires that IMS terminate the conversation by sending the last reply with RQD2,EB. Intermediate conversational messages are sent RQE2,CD. The CICS application can decide to terminate the conversation normally by issuing SYNCPOINT or RETURN. CICS can then terminate the conversation by sending LUSTATUS RQD2,EB in response to the last conversational output from IMS. In this case, the LUSTATUS acts as the acknowledgment for the RQE2,CD sent by IMS and closes that logical unit of work. The RQD2,EB causes IMS to respond with DR2 to close the bracket, dequeue the output message, and notify the Conversational Abnormal Termination exit routine that the conversation has been terminated normally by the remote program.

## Sending IMS Commands from CICS

IMS commands can be entered from any terminal authorized to use them. For terminals that are defined statically, this authorization is validated using the Security Maintenance Utility (SMU).

**Related Reading:** For more information on using SMU, see *IMS/ESA Utilities Reference: System* .

For terminals that are defined dynamically using ETO, this authorization is validated using the Resource Access Control Facility (RACF) or an equivalent product.

**Related Reading:**

- For more information on using RACF, see *IMS/ESA Administration Guide: System* .
- For more information on the ETO feature, see *IMS/ESA Administration Guide: Transaction Manager* .

## Understanding Application-Related Concepts

However, permitting a CICS application to issue IMS commands via the ISC session introduces IMS dependencies into the CICS application program. The consequences of this decision should be weighed before this facility is used.

### **Related Reading:**

- For more information on IMS commands, see *IMS/ESA Operator's Reference*.
- For more information on using IMS commands with ISC, see "Issuing IMS Commands from an ISC Session" on page 236.

A CICS application program can issue IMS commands by using the SEND/RECEIVE application program interface.

**Exception:** A CICS application program cannot issue the IMS commands /DISPLAY, /RDISPLAY, /FORMAT, and /TEST.

The IMS command verb is embedded in the message. It cannot be specified in the PROCESS parameter of the BUILD ATTACH command.

/DISPLAY, /RDISPLAY, and /FORMAT can be sent using START/RETRIEVE or SEND/RECEIVE. These commands differ from other IMS commands in that replies to these commands are queued by IMS rather than being immediately processed.

If these commands are issued with SEND (without LAST), IMS returns an LUSTATUS NO-OP as the reply to force end-bracket. IMS replies to these commands by sending the reply ATTACH BB/EB. CICS must use RECEIVE to obtain the reply. If a command is issued with SEND LAST, the reply message, when it is returned, is processed by a CICS transaction whose name has been previously specified in the RPROCESS field of the BUILD ATTACH.

If a command is issued using START/RETRIEVE, the reply is returned with both the ATTACH and SCHEDULER FM headers.

You can use the /TEST command for testing communication protocols and editing facilities on an ISC session when IMS is a back end to a CICS front end. Test mode requires the SEND/RECEIVE application program interface to be used.

**Related Reading:** For information on the functions of test mode, see "Using IMS Test Mode for ISC Sessions" on page 237.

## Sync Points

**Definition:** A *synchronization point* (also referred to as a *sync point* throughout this manual) is an occurrence within a program or subsystem wherein resources are committed and a reference point is established for subsequent restart, if necessary.

ISC session protocols define RQ\*2 requests and their associated responses (DR2 or exception DR2) as sync point requests and responses. These requests and responses are functionally independent from RQ\*1 requests and their associated responses (DR1 and exception DR1), because these latter requests and responses do not cause a sync point. The sync-point responses (DR2) are used between ISC session partners to ensure that both partners' subsystem sync point managers can commit or back out recoverable resources in synchronism.

All messages sent or received on an ISC session are defined as either recoverable or nonrecoverable, depending on the message type. The session response

## Understanding Application-Related Concepts

protocols are used to ensure that both partners of an ISC session mutually understand and agree with the recoverability attributes associated with each message. The response protocols used must be consistent with the message type and are enforced by the sending and receiving subsystem sync point managers.

When one session partner commits a message, the other session partner is notified by positive session sync-point responses. When a session partner backs out a message as the result of errors detected during input or output processing, the other session partner is notified either by session termination, by an exception response sent by the receiving subsystem, or by an LUSTATUS-function abort sense code sent by the sending subsystem. Backout results in discarding the currently active message and resetting of the associated ISC data flow control (DFC) and ATTACH states to those of the last sync point.

Because CICS is a direct-control subsystem, synchronous transactions directly control the ISC session. Therefore, backout during synchronous transaction processing also results in backing out application updates made since the last sync point.

Queued subsystems and asynchronous application output do not provide direct control of the ISC session to the transaction; therefore commit or backout does not affect application updates that are made asynchronously to the message on the ISC session. Because IMS is a queued subsystem, session sync points are separate from application sync points, and always occur outside control of the executing application.

**Related Reading:** For information on the relationship of session sync points, see “Keeping Half Sessions Synchronized” on page 287.

### Sync Points on IMS Input

For any type of input, IMS does not schedule the intended transaction until the complete input message is successfully received. Processing errors and session failure prior to the receipt of the complete input message cause the entire message to be discarded or backed out and have no effect on other recoverable IMS resources, such as databases. However, the input message cannot be cancelled by ISC session failures or protocols after the complete message is received, enqueued, and made available for scheduling. Only the current input message is backed out, even when several consecutive input (nonrecoverable) messages are received and enqueued because of a previous input sync point was requested. The definition and relationship of ISC input sync points to the application sync points depend upon whether the internal IMS execution mode for the input is synchronous or asynchronous.

**Related Reading:** For more information on ISC and IMS execution modes, see “Relationship of ISC and IMS Execution Modes” on page 238.

Table 57 summarizes the characteristics of the various IMS transaction types.

*Table 57. Characteristics of IMS Transaction Types*

<b>Synchronous<sup>1</sup></b>	<b>Asynchronous<sup>2</sup></b>
<ul style="list-style-type: none"><li>• Response mode transaction</li><li>• Conversational mode transaction</li><li>• IMS commands</li><li>• Test mode input</li></ul>	<ul style="list-style-type: none"><li>• Nonresponse, Nonconversational mode transaction</li><li>• IMS message switch</li></ul>

## Understanding Application-Related Concepts

Table 57. Characteristics of IMS Transaction Types (continued)

Synchronous <sup>1</sup>	Asynchronous <sup>2</sup>
<b>Notes:</b>	
1. Synchronous transactions from CICS are sent using SEND/RECEIVE, which generates ATTACH with CD.	
2. Asynchronous transactions from CICS are sent using SEND LAST, which generates ATTACH with EB, or START/RETRIEVE, which generates ATTACH SCHEDULER with EB.	

For IMS, the sync-point response returned to CICS can indicate successful receipt of the message or can indicate the results of IMS processing. For asynchronous input to IMS, the sync-point response returned to CICS indicates only that the input message has been successfully enqueued and that the responsibility for message recovery is assumed by IMS. For synchronous input to IMS, the sync-point response returned to CICS indicates that the transaction has executed successfully and a sync point has been taken, even though transaction execution is independent of the session. IMS reflects this by returning the input sync-point response only after the application-inserted reply message is made available for output as the result of a successful application sync point.

**Related Reading:** For more information on this processing, see “Logical Unit of Work” on page 531.

### Sync Points on IMS Output

For output, IMS commits the output message when the requested sync-point response is returned by CICS. This means that the message has been successfully sent and dequeued and the session sync point information has been updated as appropriate. IMS backs out (depending upon the sense code used) the output message when an exception response is returned by CICS. In this case, the message is returned to the message queue for later retransmission, and the associated DFC and Attach states are reset to those of the last ISC sync point. Backout on the ISC session does not affect the IMS application program or other recoverable resources, such as database updates. Back out of IMS conversational-mode output, in most cases, causes termination of the IMS conversation and the Conversational Abnormal Termination exit routine to be invoked. Based on the contents of the conversational SPA, user-provided exit logic might schedule another IMS transaction to back out database changes.

### CICS Sync Points

For CICS, sync points occur under the control of the application program and can be issued at any time. For synchronous processing, when a sync-point request is issued by the application program, CICS logs the completion of the logical unit of work (that is, the resources are committed). In addition, when the SYNCPOINT command is issued within a CICS application, CICS is responsible for appending the ISC message that was created as a result of previous processing to the SNA response request (RQD1 or RQD2) that IMS expects. However, when a CICS asynchronous transaction is scheduled as a result of IMS input, the mirror transaction has already issued a sync point, that causes the appropriate response to be returned to IMS. Any sync points issued subsequently by the application have no effect on the ISC session.

---

## Coding Function Management Headers for CICS

“Function Management Headers” on page 321 describes the content and format of the SNA-defined function management headers used by IMS. The fields used by IMS are described in detail there. This section describes how CICS uses some of the same function management header fields.

Unlike IMS, the CICS application program must prepare these function management headers and send them to IMS. CICS, itself, does not send them.

Both ends of an ISC session have an “attach manager” that performs the functions requested in the FMH.

### ATTACH Function Management Header

This section describes the ATTACH function management header fields. The primary fields that CICS uses are ATTDPN, ATTPRN, ATTRDPN, and ATTRPRN.

#### ATTDPN

The CICS transaction specifies the outbound destination process name field (ATTDPN) in the PROCESS field of the BUILD ATTACH command. This field, when sent by CICS to IMS, contains the name of an IMS editor (basic edit or ISC edit) or an MFS MID name that is to receive the inbound message. When received by CICS in a reply from IMS, this field contains the value that CICS sent to IMS as the return destination process name field (ATTRDPN) on the originating outbound message. If this is a reply to a nonresponse or nonconversational message sent to IMS with SEND LAST, this field identifies the transaction to be initiated to process this reply. In other cases, this value can be a CICS transaction code that identifies a restart transaction. If this value is not available, CICS uses the first four characters of the incoming data stream as the transaction code. If the reply is returned on the same session as that of the incoming message, IMS system code automatically wraps the incoming RDPN field into the DPN field of the reply message.

CICS ignores the ATTDPN if it is received in a reply returned from IMS.

**Exception:** After a session restart or a reply to a message sent with SEND LAST, ATTDPN is used to attach a CICS transaction.

When used with the SCHEDULER, the ATTDPN contains the DPN of the SCHEDULER model. In this case, the remaining ATTACH parameters are not sent by IMS, but rather, the information is supplied on the associated SCHEDULER model parameters.

**Related Reading:** For information on the SCHEDULER model parameters, see “SCHEDULER Function Management Header” on page 528.

#### ATTPRN

The CICS transaction specifies the primary resource name field (ATTPRN) in the RESOURCE field of the BUILD ATTACH command. This field, when sent by CICS to IMS, names an IMS LTERM or transaction code that is the message destination. When this parameter is omitted, IMS uses the first eight characters of the data stream to identify the message destination.

## Coding Function Management Headers for CICS

**Recommendation:** To maintain consistency between IMS and CICS, use this field to name an LTERM for IMS message switches and place transaction codes in the first eight characters of the data stream.

When received by CICS in a reply from IMS, this field contains the value that CICS sent to IMS as the return primary resource name (ATTRPRN) on the outbound message. CICS ignores the ATTRPRN if it is received in a reply returned from IMS.

### **Exceptions:**

- After session restart, ATTRPRN is used to determine the identity of the terminal that originated this transaction and must be acquired during the restart.
- When the reply is to a nonresponse or nonconversational transaction issued by CICS with SEND LAST, ATTRPRN is used.

If the reply is to be returned on the same session as that of the incoming message, IMS automatically wraps the value received on the incoming RPRN field into the PRN field. MFS can be used to set or override this value on IMS output.

### **ATTRDPN**

The CICS transaction specifies the return destination process name (ATTRDPN) in the RPROCESS field of the BUILD ATTACH command. When CICS sends this field to IMS, it contains the code of a transaction to be executed in the event that a session restart is required or a reply is to be returned to a message that CICS sent to IMS using SEND LAST. IMS sends this to CICS when a next MID is specified within an MFS message output descriptor (MOD). When CICS receives the ATTRDPN field on an input message, it can be examined by the EXTRACT ATTACH command. This value should be saved and returned to IMS as the DPN on subsequent replies.

### **ATTRPRN**

The CICS transaction can place the identification of the originating terminal in this field. This identification is wrapped into the PRN field by IMS if a session restart is required or a reply is required to a message sent to IMS by CICS with SEND LAST. The CICS transaction places this value in the field by using the RRESOURCE field of the BUILD ATTACH and examines this field by using the EXTRACT ATTACH command.

For asynchronous unsolicited output, IMS uses the source LTERM name as the default, if a reply is necessary. MFS can be used to set or override this value on any type of output. When a CICS application receives this value, the value should be saved and returned as the PRN field for subsequent replies.

### **ATTDQN and ATTDP**

When sent to CICS from IMS, on the ATTACH for a demand-paged output message, ATTDQN contains a unique message identifier. This field is only used for IMS demand paging. The CICS transaction specifies the destination message ID (ATTDQN) for subsequent demand-paged requests as the QNAME value in the returned QMODEL FM header. When CICS receives ATTDQN from IMS, the ATTDP field is set to 1. These fields are not set by IMS on any other conditions nor are they accepted by IMS on input.

### **ATTIU**

The CICS transaction specifies the interchange unit code (ATTIU) in the IUTYPE field of the BUILD ATTACH command. Two values can be specified: single chain, which applies to all non-MFS input or output, and multichain, which applies only to MFS-autopaged input and output. Single chain should be used unless the input to IMS is multiple-page MFS-autopaged input.

## Coding Function Management Headers for CICS

### ATTDSP

The CICS transaction specifies the data stream profile (ATTDSP) in the DATASTR field of the BUILD ATTACH command to determine the IMS component. This field can contain the following values:

X'00' - identifies IMS component 1  
X'01' - identifies IMS component 2  
X'02' - identifies IMS component 3  
X'03' - identifies IMS component 4

When received by IMS, this input component determines the default output component, the input security requirements, and whether MFS can be used for input. The output component determines whether MFS is used for output, and the protocols that are to be sent on asynchronous output are sent ATTACH SCHEDULER. On output, IMS sets this value to the value of the output LTERM.

### ATTDBA

The CICS transaction specifies the deblocking algorithm (ATTDBA) in the RECFM field of the BUILD ATTACH command.

### ATTACC

This parameter is not supported on an ISC session and causes session termination if sent.

Table 58 summarizes the source of the values placed in the ATTACH FM header fields.

*Table 58. Source of Values Placed in the ATTACH FM Header Fields*

Major Field	A <sup>1</sup>	B <sup>2</sup>	C <sup>3</sup>
ATTDPN <sup>4</sup>	PROCESS	DPN	
ATTPRN	RESOURCE	PRN	
ATTRDPN	RPROCESS	RDPN	
ATTRPRN	RRESOURCE	RPRN	
ATTDQN	QUEUE	Message ID <sup>5</sup>	
ATTIU	IUTUPE <sup>6</sup>	Paging <sup>6</sup>	
ATTDSP	DATASTR <sup>7</sup>		COMPTn <sup>8</sup>
ATTDBA	RECFM <sup>7</sup>		VLVB DPM-Bn <sup>9</sup>

## Coding Function Management Headers for CICS

*Table 58. Source of Values Placed in the ATTACH FM Header Fields (continued)*

**Notes:**

1. CICS EXEC BUILD/EXTRACT ATTACH.
2. Set by IMS MFS; or, if MFS is not used, and the output is asynchronous unsolicited output, the RPRN defaults to the source LTERM name.
3. IMS COMPTn in TERMINAL/NAME macros.
4. For asynchronous processing, this field indicates "SCHEDULER follows" by containing the value X'02'. In this case, the remaining header fields are not contained in the ATTACH FM header, but are contained instead in the concatenated SCHEDULER FM header.
5. Message ID for MFS demand paging.
6. Value is set depending on MFS autopage.
7. Second byte of these halfword binary fields is used.
8. Components numbered 1 through 4.
9. DPM-Bn permits MFS to be used. Using MFS MODE=STREAM on output changes the ATTDDBA to chain mode; otherwise, IMS uses the default of VLVB.

## SCHEDULER Function Management Header

The SCHEDULER function management header can be used to send asynchronous messages between IMS and CICS. On input, receipt of the SCHEDULER FMH causes CICS to invoke the mirror transaction to schedule the asynchronous transaction that is to process the input. On output, it is generated as a result of CICS using the START command with its associated fields as follows:

### SCDDPN

The CICS transaction specifies the value to be placed in the outbound destination process name field (SCDDPN) in the TRANSID field of the START command. This field, when sent by CICS to IMS, contains the name of an IMS editor (MFS, basic edit, or ISC edit) that is to receive the inbound message. If SYSID is not coded on the START command, this TRANSID can be modified (to contain an 8-character name, for example) as a result of the RMTNAME option in the program control table (PCT).

When received by CICS in a reply from IMS, this field contains the value that CICS sent to IMS as the return destination process name field (SCDRDPN) on the outbound message. IMS automatically wraps the incoming RDPN field into the DPN field of the reply message. Alternatively, this field might have been set by the IMS MFS. This value is a CICS transaction code. If this field is not supplied on input to CICS, CICS uses the first four characters of the input data stream as the transaction code.

### SCDPRN

The CICS transaction specifies the primary resource name field (SCDPRN) in the TERMID field of the START command. This field, when sent by CICS to IMS, names an IMS LTERM or transaction code that is the message destination. If this field is omitted when CICS sends a message to IMS, IMS examines the first eight bytes of the incoming message to extract the message destination.

**Recommendation:** To maintain consistency between IMS and CICS, use this field to name an LTERM for IMS message switches and place transaction codes in the first eight characters of the data stream.

When received by CICS in a reply from IMS, this field contains the value that CICS sent to IMS as the return primary resource name (SCDRPRN) on the outbound



## Coding Function Management Headers for CICS

message or a value placed into the field by MFS. This field contains a value that identifies the terminal that is to be used as a principal facility and to which the reply should be sent.

### SCDRDPN

The CICS transaction specifies the return destination process name (SCDRDPN) in the RTRANSID field of the START command. When CICS sends this field to IMS, it contains the name of the asynchronous transaction to be scheduled by the CICS mirror transaction when the reply is received.

For unsolicited asynchronous output, the RDPN can be set by MFS to indicate the next message input descriptor (MID) to be used for subsequent replies. If this parameter is received as input by CICS, this parameter should be saved and returned to IMS in the TRANSID (DPN) of the START command.

### SCDRPRN

The CICS transaction specifies the return primary resource name (SCDRPRN) in the RTERMID field of the START command. When CICS sends SCDDPRN to IMS, it contains the identification of the terminal to which the return reply should be routed. On output from IMS to CICS, IMS sets SCDDPRN to the name of the source LTERM. MFS can also set SCDDPRN. The SCDDPRN value should be saved by CICS and returned to IMS as the TERMID (PRN) on any subsequent replies.

### SCDDQN and SCDDP

Because asynchronous demand paging is not recommended, these fields are not used in a CICS-IMS ISC session.

The SCHEDULER header is preceded by an ATTACH header. Some mandatory ATTACH FM header fields also apply to asynchronous messages. These fields are not carried on the SCHEDULER header, but are retained on the ATTACH header. These fields are the ATTIU, ATTDSP, and ATTDDBA fields.

**Related Reading:** For more information on these fields, see “ATTACH Function Management Header” on page 525. Their values in the asynchronous environment are as follows:

- ATTIU  
CICS only supports single-chain messages between itself and IMS.
- ATTDSP  
When sent using the START command, the ATTDSP value is determined by the terminal-control table generation, and should be X'00' (component 1). For input and output, the definition of the ATTDSP field for IMS is the same as that described in “ATTACH Function Management Header” on page 525.
- ATTDDBA  
The CICS transaction specifies the deblocking algorithm (ATTDDBA) in the RECFM field of the BUILD ATTACH command.

Table 59 summarizes the source of the values placed in the SCHEDULER FM header fields.

*Table 59. Source of Values Placed in the SCHEDULER FM Header Fields*

Major Field	A <sup>1</sup>	B <sup>2</sup>	C <sup>3</sup>	D <sup>4</sup>
SCDDPN	TRANSID <sup>5</sup>		DPN	
SCDPRN	TERMID <sup>5</sup>		PRN	
SCDRDPN	RTRANSID <sup>5</sup>		RDPN	

## Coding Function Management Headers for CICS

Table 59. Source of Values Placed in the SCHEDULER FM Header Fields (continued)

SCDRPR	RTERMID <sup>5</sup>	RPRN
SCDDQN	QUEUE	Note 6
ATTIU <sup>7</sup>		Note 6
ATTDSP <sup>7</sup>	DATASTR <sup>8</sup>	COMPTn <sup>8</sup>
ATTDBA <sup>7</sup>	RECFM	VLVB DPM-Bn <sup>9</sup>

### Notes:

1. CICS EXEC START/RETRIEVE.
2. CICS DEFINE CONNECTION or DFHTCT TYPE=SYSTEM.
3. Set by IMS MFS; or, if MFS is not used, for output that is unsolicited asynchronous output, the RPRN defaults to the source LTERM name.
4. IMS COMPTn in TERMINAL/NAME macros or on an ETO logon descriptor.
5. Four-byte name; MFS supports an eight-byte name.
6. MFS paging is not recommended on an asynchronous session between IMS and CICS.
7. ATTACH FMH fields preceding SCHEDULER FMH.
8. DATASTR must be specified as USER. This causes X'00', specifying IMS component 1, to be indicated as the input component. The IMS output component can be components 1 through 4 (X'00'- X'03').
9. DPM-Bn permits MFS to be used. Using MFS MODE=STREAM on output changes the ATTDBA to chain mode; otherwise, IMS uses the default of VLVB.

## Queue Model Function Management Headers

CICS uses the QMODEL headers to access IMS MFS demand-paged messages. QMODEL headers are not sent by IMS to CICS. For SEND/RECEIVE, the CICS application program must prepare the QMODEL function management headers in order to receive demand-paged output from IMS. Temporary storage function shipping can be used with START/RETRIEVE. No EXEC commands exist to support the coding of QMODEL FM headers. The following QMODEL headers are used for MFS demand paging:

<b>QGET</b>	Requests a page directly
<b>QGETN</b>	Requests the next sequential page
<b>QPURGE</b>	Requests termination of demand paging
<b>QXFR</b>	Sent with the requested page
<b>QSTAT</b>	Sent when QPURGE is received, or when the requested page number is invalid

## Data Descriptor Function Management Header

IMS uses this header to specify a device page (DPAGE). It is not used by CICS, but can be passed to a CICS transaction. A CICS synchronous transaction can build and send data descriptor FM headers to IMS as appropriate. However, no EXEC commands exist to support the coding of data descriptor FM headers.

## System Message Process (SYSMSG) Function Management Header

IMS uses this header to send system messages. System messages are appended to an ATTACH carrying DPN=SYSMSG. CICS routes these messages to a synchronous application when the session is allocated to that transaction or to destination CSMT

## Coding Function Management Headers for CICS

at any other time. If routed to a transaction, that transaction is responsible for processing the SYSMMSG function management header. CICS does not send SYSMMSG, but a synchronous application program can cause a SYSMMSG to be sent by setting the appropriate process name in the BUILD ATTACH. The application program is responsible for building the appropriate SYSMMSG FMH to be appended to the ATTACH.

SYSMMSG is sent by IMS for broadcast messages or for the case in which a response to input has been sent, but has had a subsequent processing error. After a response to asynchronous input, SYSMMSG is sent in lieu of the reply message. During synchronous processing, SYSMMSG is sent:

- After a session restart, if the transaction in progress abends during the session outage
- During MFS output, if MFS output format blocks are not available or are not valid

## Error Recovery Procedure Function Management Header

CICS sends and receives this header, which is used to transmit error information from one process to another. The header is sent after an exception response (X'0846') and carries sense codes, which are used to inform the half-session partner of the nature of the error. It is followed by an error message. CICS writes the received IMS error message to the transient data destination CSMT; IMS writes the received CICS error message to the master terminal or the input source node.

---

## Recovery and Restart Concepts

This section describes the system and user functions that must be performed to recover an ISC session after a session, system, or application failure. It assumes the reader understands the role of the STSN command in session resynchronization.

**Related Reading:** For the format of the STSN action and response commands, see “Determining Session Synchronism Using STSN” on page 276.

The concepts of sync point, commit, backout, and logical unit of work are common to IMS and CICS. However, their meanings differ slightly within IMS and CICS.

**Related Reading:** For information on sync points, see “Sync Points” on page 522.

## Logical Unit of Work

A logical unit of work is any processing that occurs between sync points. Within subsystems where the application directly controls the session, such as CICS, the application sync point and the ISC session sync point occur simultaneously as the result of a single explicit or implicit application command. Within queued subsystems, such as IMS, the application processing and sync points are independent of those of the ISC session. IMS must map the application sync point to the ISC session sync-point request.

In the synchronous case (SEND/RECEIVE), CICS is the front-end subsystem. One logical unit of work includes all of the processing performed in both the IMS and CICS subsystems, from the point at which the CICS application last issued a sync point to the point at which the next sync point is issued, and a sync-point response is returned.

## Recovery and Restart Concepts

If the transaction is recoverable, only one message can be sent to IMS before the sync point must be requested by the CICS application. However, a series of nonrecoverable requests and replies can occur between a CICS sync-point request and an IMS response.

For recoverable messages, IMS returns this response only after the IMS application has inserted a response or conversational mode reply message and caused an application sync point. IMS then begins the next logical unit of work by sending the reply message requesting a sync point. The reply message satisfies the RECEIVE issued by the CICS transaction. The CICS transaction should complete all necessary processing prior to issuing a SYNCPOINT or RETURN (implicit sync point) command.

The application sync point causes CICS to return a sync-point response to IMS. IMS then completes the logical unit of work by dequeuing the reply message from the output message queue.

In the asynchronous case, where CICS is the front-end subsystem, for recoverable input to IMS, the input request and the returned reply message are separate logical units of work. When the CICS application issues a SEND LAST or START and requests a sync point, IMS returns the sync-point response as soon as the transaction is enqueued (made available for scheduling). IMS sends any recoverable asynchronous reply messages as they are made available by the transaction and requests a sync point on each. When the reply message is read by the CICS mirror transaction, an appropriate response is automatically returned to IMS. If the reply message is read using RECEIVE as a result of a previous SEND LAST, the sync point response is not returned to IMS until a subsequent SYNCPOINT, FREE, or RETURN command is issued.

One or more nonrecoverable transactions can be sent to IMS between CICS sync points using multiple START commands, each followed by a RETRIEVE. When the reply message is read by the CICS mirror transaction, an appropriate response is automatically returned to IMS.

The concept of logical unit of work (or work unit) is important in that session resynchronization must be performed when initiating a session and up to one work unit is considered to be in-doubt on a flow from either the primary to the secondary half session or from the secondary to the primary. An in-doubt work unit is one that is waiting to be committed or backed out based on the results of the session BIND and STSN flow.

## Recovering Outstanding Message Traffic after a Failure

A CICS-IMS session can fail or terminate in any of the following situations:

- A communication component (for example, VTAM or NCP) fails.
- The CICS or IMS subsystem fails.
- A direct VTAM or subsystem command (for example, an IMS /CLSDST or /STOP command) is entered.
- The CICS transaction or IMS transaction fails (indirectly as a result of subsequent error processing).

In each of these situations, the session failure appears similar to the remaining operative subsystems. However, the resulting recovery and resynchronization processes you must follow differ. This section describes what happens in the event

## Recovery and Restart Concepts

of a communication component failure, an IMS or CICS subsystem failure, or the issuing of a direct command against the session between the two subsystems.

**Related Reading:** For information on abnormal termination of an IMS or CICS transaction, see “Handling Transaction Abends” on page 536.

### Reestablishing the Session

The failed session can be restarted by either half-session partner. When it is restarted, the relationship of the half sessions is the same as when the session failed; that is, the former primary half session continues to be primary and the former secondary half session to be secondary. Both IMS and CICS remember this orientation by logging the session qualifier pair (SQP) used to initiate the session with an indicator of the session polarity.

The CICS master terminal operator can reinitiate the session using the command:  
CEMT SET TERMINAL (*tttt*) ACQUIRED

where *tttt* is the TERMIDNT of the CICS session as defined in the terminal control table. CICS then brings up the session, maintaining the session polarity that has been fixed by CICS system definition parameters. If the session is being initiated by an authorized IMS terminal operator's issuing the /OPNDST command, that command must specify the same subpool name (IMS local session qualifier name) and session ID (CICS local session qualifier name) that was allocated to the session at the point of failure. These are available to the operator as necessary using an IMS /DISPLAY command. IMS automatically reestablishes the session, while maintaining the same session polarity as was in effect at session failure.

If the session is being restarted with IMS in cold-start mode, and if CICS is defined to be the primary half session, CICS must initiate the restart.

When the session is reestablished, IMS attempts to reset the data flow control bracket state to its status at the last sync point before session failure. CICS allows the session to be reestablished as between-brackets or as in-brackets with IMS in SEND state.

### Resynchronizing the Session

When a session is active, both IMS and CICS maintain a set of SNA message sequence numbers for that session. When a failure occurs and restart is attempted, the total of sync points on the session is checked between the subsystems by the STSN command.

**Related Reading:** For more information on STSN content and flow, see “Determining Session Synchronism Using STSN” on page 276.

The CICS/IMS session resumes if both of the following are true:

- The session is being cold-started.
- The half sessions' sequence numbers agree, or are within acceptable limits.

**Related Reading:** For more information on the acceptable range of session sequence numbers, see “Set-and-Test-Sequence-Numbers (STSN)” on page 435.

If a session fails, a CICS transaction having pending activity for that session also fails. CICS uses dynamic transaction backout (DTB) to ensure integrity of recoverable resources. The DTB in-doubt parameter must be specified as

## Recovery and Restart Concepts

IN-DOUBT(WAIT) or DTB=(YES,WAIT) in order to ensure consistency between the IMS and CICS subsystems within an ISC network.

**Related Reading:** For more information on CICS backout in-doubt processing, see “Defining CICS Backout In-Doubt Processing” on page 515.

WAIT holds locks on recoverable resources until resynchronization occurs.

If this situation is undesirable, you can specify IN-DOUBT(BACKOUT) or DTB=YES on an IMS-CICS ISC transaction. However, doing so might cause duplicate messages to be sent to IMS after session resynchronization, and can require logic in the IMS user transaction to resolve them.

If the DTB in-doubt parameter does not specify WAIT, you must specify FORCSESS on the IMS TERMINAL macro system definition OPTIONS parameter or use an authorized IMS terminal operator /CHANGE command.

### Processing Outstanding Traffic

In IMS, transaction processing is independent of session status; that is, if a message is received and successfully queued, it is always processed. However, the way in which IMS handles subsequent replies and continues processing conversations depends on the type of message, the type of error that occurs, and whether the IMS subsystem fails.

In general, when only the session fails (and IMS does not fail), and no synchronous processing is occurring between IMS and CICS, the following is true:

- The session is bound between-brackets when it is reinitiated.
- IMS always sends or resends asynchronous replies or unsolicited asynchronous output after successful session restart.
- IMS sends or resends queued asynchronous replies resulting from /DISPLAY, /RDISPLAY, and /FORMAT commands.
- If the IMS asynchronous transaction abends during the session outage, IMS sends an error message using ATTACH SYMSG, because exception responses are no longer possible.

For replies processed synchronously by IMS:

- If response mode or conversational output is pending, the session is bound with IMS in-brackets/SEND to permit the pending reply to be sent or re-sent.
- When in conversation mode and input is required, IMS attempts to bind in-brackets/RECEIVE. CICS negotiates this bind to a between-brackets state, causing IMS to terminate the conversation, discard the output message, and invoke the Conversational Abnormal Termination exit routine. This exit routine can invoke user processing to schedule an IMS transaction to back out any database changes resulting from previous conversational processing based on the contents of the SPA.

When a session is reestablished and sent to between-brackets as described in the preceding paragraph, if the IMS transaction has not yet completed processing the conversational or response mode input (that is, output is not yet available), IMS terminates the session with a message to the master terminal operator requesting that the session be reinitiated later (when output is available). This occurs because IMS cannot terminate the in-process transaction as required by the between-brackets bind.

## Recovery and Restart Concepts

- Except as noted above, IMS cancels any commands received at the point of failure and discards pending synchronous reply messages that result from an IMS command.
- If the IMS synchronous transaction abends during the session outage, IMS sends an error message using ATTACH SYSMMSG, because exception responses are no longer possible.

IMS handles these transactions in the following manner:

- After an emergency restart, IMS discards a pending reply message resulting from a nonrecoverable transaction.
- After IMS receives and enqueues an input transaction, session protocols or failures cannot cancel that transaction.
- After a reply message to a CICS synchronous transaction (response mode or conversation) is enqueued, it indicates that the results of all previous IMS processing have been committed. These updates are not backed out even if the reply message is discarded.

If a CICS transaction must be re-sent to IMS as determined by STSN processing, the CICS application, rather than the CICS subsystem itself, must make this determination and either reconstruct the transaction or request that it be reentered.

**Related Reading:** For more information on IMS transaction abends, see “IMS Transaction Abend” on page 536.

After a session fails, CICS cannot reestablish the environment in which the original transaction was executing. That is, the connection to the terminal that originally entered the transaction no longer exists. Further, the transaction has abnormally terminated; therefore, it is necessary for the CICS application programmer to define a “restart transaction” to be invoked by CICS to handle any IMS output that CICS might receive on the restarted session. When this transaction is invoked, the session, rather than the terminal, is now the primary facility. This transaction must in turn invoke an asynchronous transaction to acquire the terminal if output exists that needs to be sent to it.

The transaction code of the restart transaction to be invoked is carried in the DPN field of the function management header that is sent to CICS with the input message. This transaction code is the one specified in the RPROCESS field of the BUILD ATTACH command sent by CICS with the outbound message. This becomes the RDPN parameter automatically wrapped by IMS to the DPN field, which IMS sends on the outbound FMH unless modified by the IMS MFS.

The ID of the terminal to which CICS is to send any output reply is found in the PRN field of the incoming function management header. This is the value specified in the RRESOURCE field of the BUILD ATTACH sent by CICS with the outbound message. This becomes the RPRN parameter automatically wrapped by IMS to the PRN field of the outbound FMH unless modified by MFS. This value is used as the TERMID of the asynchronous transaction scheduled (START) by the restart transaction to acquire the terminal.

The restart transaction issues a RETRIEVE carrying both TRANSID and TERMID fields. When the transaction specified by the TRANSID is initiated, it owns the terminal specified by the TERMID as its principal facility.

**Related Reading:** For more information on designing CICS restart transactions, see “Coding CICS Applications for Restart” on page 537.

### Handling Transaction Abends

In addition to the considerations for session and subsystem failure, the design of CICS recovery transactions must also take into account the actions to be taken in the case of transaction termination as described in the sections that follow.

#### IMS Transaction Abend

When an IMS transaction terminates abnormally, all changes to IMS resources that were performed by this transaction are backed out by IMS. If the transaction is synchronous (CICS SEND/SYNCPOINT), IMS sends a negative response (exception DR2) to CICS. This causes the CICS transaction to also terminate abnormally. As a result of the negative response and subsequent CICS transaction abend, CICS backs out any changes made to recoverable resources after the previous CICS sync point. An error message is also sent to CICS; CICS routes this message to the master terminal destination CSMT. The exception occurs when MFS output formats do not exist in the format library or incur an I/O error. In this case, IMS has already sent a positive sync-point response and must indicate the MFS error to CICS by using an ATTACH SYSMSG.

If the transaction is asynchronous (CICS START or SEND LAST), IMS sends an error message to CICS; CICS routes this message to its master terminal destination. The error message is preceded by either a SYSMSG or an ERP header, depending upon the type of error that occurs. If the error occurs before IMS sends a sync-point response to input, an exception response and ERP are used; if after the sync-point response, ATTACH SYSMSG is used.

**Related Reading:** For more information on function management headers, see “Coding Function Management Headers for CICS” on page 525.

#### CICS Transaction Abend

If a transaction is initiated by a CICS subsystem acting as a front end for IMS, and that CICS transaction terminates abnormally before reaching sync point, any processing performed is handled in accordance with the specification on the DEFINE TRANSACTION IN-DOUBT parameter (or DFHPCT DTB= parameter).

If an asynchronous transaction is initiated by an IMS front end and the CICS transaction abends upon receipt of the transaction, IMS is not notified. This is because the receipt of the message causes the CICS mirror transaction to return an immediate DR2 to the asynchronous input and to schedule a transaction to process it. If this scheduled transaction fails, the ISC link to IMS is no longer active and no notification is possible. CICS does, however, notify the CICS destination CSMT of the transaction failure.

During synchronous processing, if CICS is the secondary half session, the ALLOCATE command has been successfully issued, and no message is available to be sent to IMS (due, for example, to DTB causing backout), CICS automatically deallocates the session by sending LUSTATUS BB/EB to IMS.

If the synchronous transaction terminates abnormally any time after it has sent a message to IMS using the SEND/SYNCPOINT command, and that message has been enqueued on an IMS input queue, IMS processes it. When IMS attempts to send an output message to CICS, that message receives a negative response from CICS. As a result, the message can remain on the IMS output queue until it can be re-sent or until it is dequeued by the IMS master terminal operator.



**Related Reading:** For information on the ERP sense code actions, see “Error Recovery Procedure Function Management Header” on page 531.

---

### Coding CICS Applications for Restart

When a failure occurs on an ISC session as a result of session or subsystem failure, session restart and resynchronization can be attempted. If session resynchronization (STSN processing) is unsuccessful and the session cannot be restarted without intervention, the master terminal operators of both subsystems are notified. This situation can require that the terminal user also be notified of the session’s status and of any actions that must be taken, such as reentering the failing transaction or waiting until corrective action is taken by the MTO.

After an ISC transaction defined as recoverable is received and logged by IMS, it can be recovered across session and subsystem failures. After restart, when the session has been recovered, if IMS has pending output, it is sent as follows:

- When IMS sends output on the same session as that on which the input message is received, that output message is sent with the same type of headers as those originally sent with the input. Thus, a reply to a message sent with SEND/RECEIVE or SEND LAST is returned with the ATTACH FM header and a reply to a message sent with START/RETRIEVE is returned with the ATTACH and SCHEDULER FM headers.
- When IMS sends output on a session other than that on which the input message is received, that output message is considered as unsolicited output and is sent asynchronously (with ATTACH SCHEDULER).

After a restart, when IMS sends an output message, CICS reads the input message and initiates a transaction.

- Within the application, the EXEC ASSIGN STARTCODE parameter is examined to determine whether this transaction had been initiated synchronously or asynchronously.
- If the restart transaction is initiated asynchronously:
  - A RETRIEVE is issued to obtain the ISC reply. The EIB indicators should be saved. CICS attaches the end-user’s terminal to this CICS program as the principal facility. This permits the IMS output to be issued directly to the terminal using a CICS EXEC SEND command.
  - The EIB indicators, saved after RETRIEVE execution, must be tested to determine whether SYNCPOINT or RETURN can be issued to end this transaction.
- If the transaction is initiated synchronously by SEND/RECEIVE or asynchronously by SEND LAST:
  - The input message is obtained using RECEIVE. In this case, the session is now the principal facility. Therefore, the RECEIVE command does not require specification of the SESSION parameter.
  - After the RECEIVE, the CICS application saves and then checks the EIB fields EIBSYNC, EIBFREE, and EIBRECV, in that order.
  - The application now issues an EXTRACT ATTACH to determine the format of the input data and the ID of the terminal that originally submitted the transaction.
  - The restart logic must now START an additional CICS transaction to acquire the end user’s terminal as the principal facility. The START request is made with the terminal ID found in the PRN field by using the EXTRACT ATTACH as the TERMID parameter on the START.

## Coding CICS Applications for Restart

- The EIB indicators are now tested to determine whether a sync point is required and whether the session can be freed. If this can be done, the application can issue a RETURN command to implicitly cause a sync point to be issued and the session to be freed.

Within the context of the restart transaction, data can be retrieved for inquiry purposes or databases and files can be updated.

In addition to the foregoing logic, a CICS restart transaction can contain logic to:

- Store input received from the originating terminal on temporary storage. If it is necessary to re-create an input transaction, this information can be obtained and re-sent to IMS.
- Create logic to inform the terminal user of any special processing to be performed, such as reentering a transaction or waiting for some master terminal operator action before proceeding.

---

## Appendix H. ISC Data Flow Control Examples

This appendix provides ISC data flow control examples.

### In this Appendix:

- “Non-MFS Bracket and Half-Duplex Protocol Examples”
- “MFS Bracket and Half-Duplex Protocol Examples” on page 540
- “SBI/BIS Examples” on page 545
- “Signal Protocol Example” on page 546

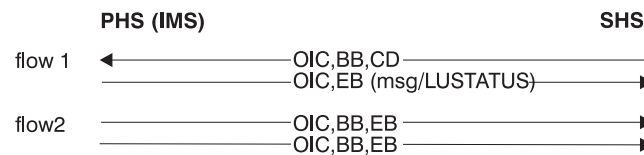
---

## Non-MFS Bracket and Half-Duplex Protocol Examples

This section contains examples that illustrate ISC data flow control protocols. The items in parentheses are optional on the flow.

For simplicity, the examples in this section assume only-in-chain SCHEDULER messages for flows in both directions, and the response, ATTACH, and SCHEDULER protocols have been excluded. Also, the bracket protocol is considered symmetrical in that either half session can initiate the bracket as illustrated.

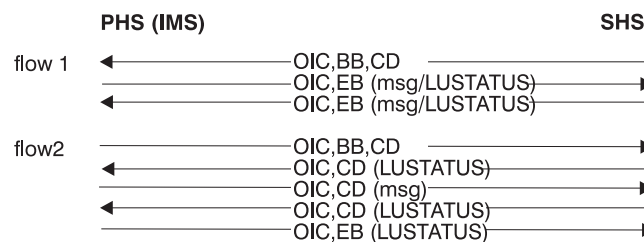
### **Example 1: Bracket Protocol for a PHS Component Defined to IMS as SINGLE1**



Sample flow 1 shows the PHS ending a bracket on the first output following receipt of change-direction. When no output exists and the input cannot guarantee output, the bracket is ended through a stand-alone LUSTATUS.

Sample flow 2 shows the PHS sending output that occurs while in a between-brackets state.

### **Example 2: Bracket Protocol for a PHS Component Defined to IMS as SINGLE2**

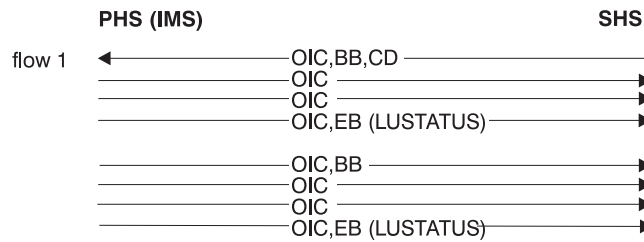


Sample flow 1 shows the PHS returning the flow to a bracket initiator after receipt of a change-direction. The bracket initiator can optionally continue input or end the bracket. When no output exists and the input cannot guarantee output, the flow is returned to the bracket initiator using a stand-alone LUSTATUS.

## Non-MFS Bracket and Half-Duplex Protocol Examples

Sample flow 2 shows the PHS sending asynchronous output. Also, IMS detects a potential LUSTATUS CD loop and forces end-bracket using LUSTATUS.

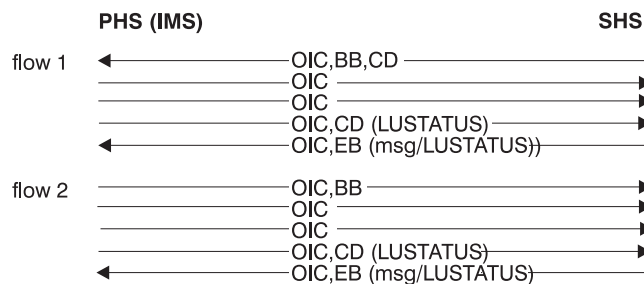
### Example 3: Bracket Protocol for a PHS Component Defined to IMS as MULT1



Sample flow 1 shows the PHS ending a bracket using an LUSTATUS. The bracket ends following the last output from a queue after receipt of change-direction. If no output exists and the input cannot guarantee output, the bracket is ended immediately by using a stand-alone LUSTATUS.

Sample flow 2 shows the PHS both beginning and ending a bracket by using an LUSTATUS following the last output from a queue. Additional output available from other queues causes subsequent brackets to be initiated.

### Example 4: Bracket Protocol for a PHS Component Defined to IMS as MULT2



Sample flow 1 shows the PHS returning the flow to the bracket initiator using an LUSTATUS. The bracket ends following the last output from a queue after receipt of a change-direction. The bracket initiator can optionally continue input or end the bracket. If no output exists and the input cannot guarantee output (that is, the input is not in conversational or response mode), the flow is returned by using a stand-alone LUSTATUS.

Sample flow 2 shows the PHS sending output that occurs in a between-brackets state.

---

## MFS Bracket and Half-Duplex Protocol Examples

This section provides examples for both MFS output and MFS input.

### MFS Output Examples

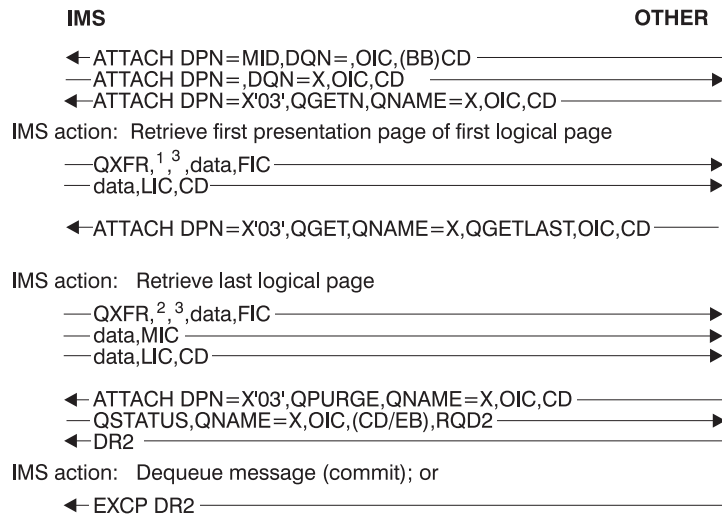
In examples 1 through 4, the IMS message consists of 3 presentation pages:

- Logical page 1 that makes up 2 presentation pages

- Logical page 2 that makes up 1 presentation page



## MFS Bracket and Half-Duplex Protocol Examples

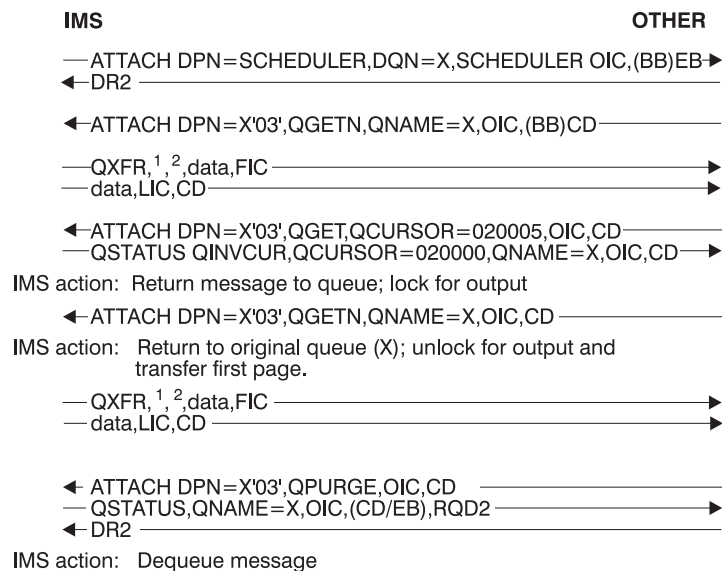


### Notes:

1. QCURSOR=0400010001, QCOUNT=020002
2. CURSOR=0400020001, QCOUNT not present
3. Data descriptor FM header can precede the data

### Example 3: Demand-Paged SCHEDULER Output, OLP Defined, QGET by Cursor Request for a Page Not within the Range of Output Message

These same conditions for non-SCHEDULER demand-paged output result in an exception response and in subsequent ERP FMH7 being sent.

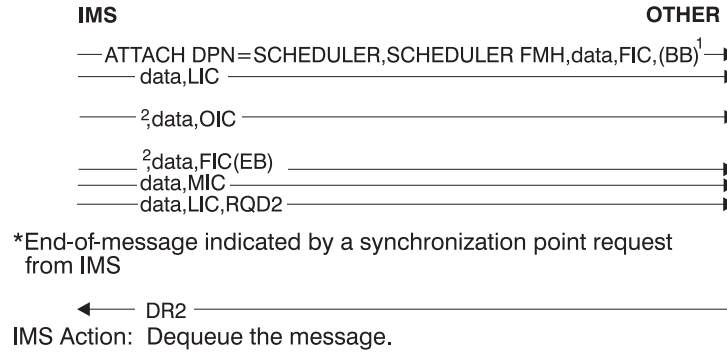


### Notes:

1. QCURSOR=0400010001, QCOUNT=020002
2. Data descriptor FM header can precede data

### Example 4: Autopaged Output

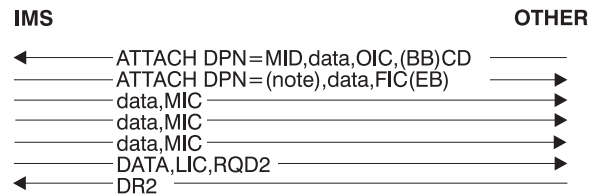
## MFS Bracket and Half-Duplex Protocol Examples



### Notes:

1. Sending a first-in-chain with BB on autopaged output requests DR1 (RQD1).
2. Data descriptor FM header can precede the data in each chain. The ATTACH indicates a multichain message.

### Example 5: Nonpaged Output Message



Data descriptor FM header can precede the data.

## MFS Input Examples

The following description is provided for examples 6 and 7.

Three DPAGES are defined:

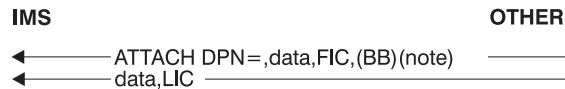
- DPAGE 1 defines data for 2 segments
- DPAGE 2 defines data for 1 segment
- DPAGE 3 defines data for 3 segments

### Example 6: Autopaged Input, Three Chains

Three LPAGES are created for the message. A data descriptor function management header with a DSN supplied with each chain is used to select each DPAGE. The first DSN name selects DPAGE 1, the second DSN name selects DPAGE 2, and the third DSN name selects DPAGE 3.

## MFS Bracket and Half-Duplex Protocol Examples

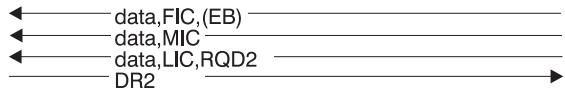
\*ATTACH and data for first DPAGE:



\*Data for second DPAGE:

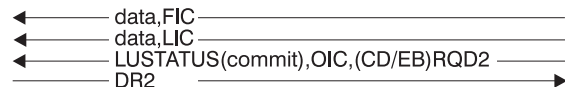


\*Data for third DPAGE:



\*End-of-message is indicated via a sync-point request (RQD2) with data

- OR -



\*End-of message is indicated via a sync-point request (RQD2) in the subsequent RU without data

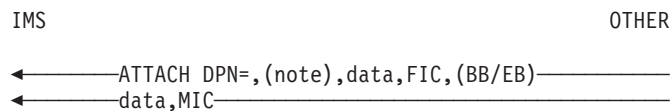
IMS action: Enqueue the input message.

A data descriptor FM header can precede the data.

### Example 7: Autopaged Input, Single Transmission Chain

Three LPAGEs are created for the message. A conditional test on the data is performed to select each DPAGE. Results of first test selects DPAGE 1, results of the second test selects DPAGE 2, and results of the third test selects DPAGE 3.

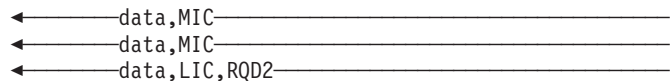
\* ATTACH and data for first DPAGE:



\* Data for second DPAGE:



\* Data for third DPAGE:



\* End-of-message indicated via a sync point request with data



IMS action: Enqueue the input message

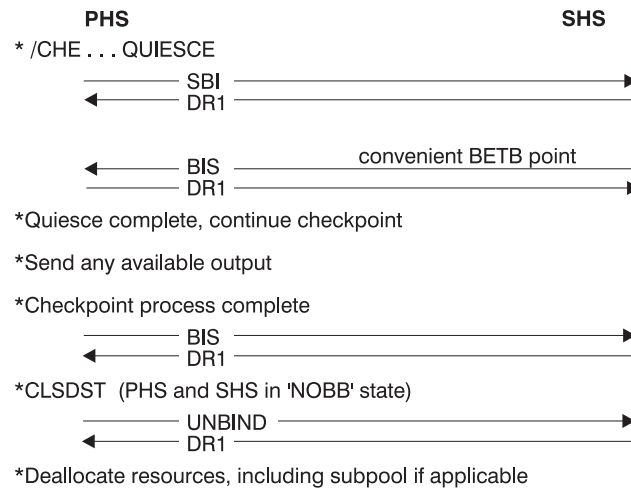
The data descriptor FM header can be sent to select one DPAGE only. The ATTACH indicates a multichain message.



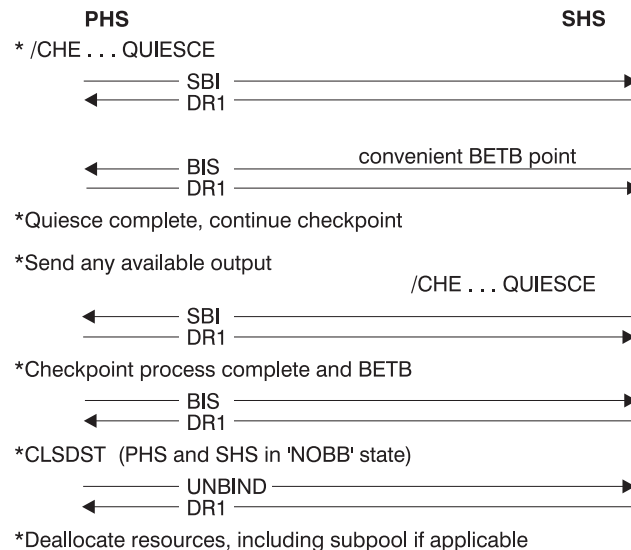
## SBI/BIS Examples

IMS can be either the primary half session (PHS) or the secondary half session (SHS) for all of the examples given. Therefore, all of the functions and commands shown for PHS and SHS are IMS functions and commands. Although all SBI and BIS sequences are valid for any session using LU 6.1 protocols, the following examples assume IMS to be both the primary and the secondary half session. No assumptions are made here as to when or how a subsystem other than IMS might send or receive these sequences.

### Example 1: PHS Shutdown Using SBI/BIS



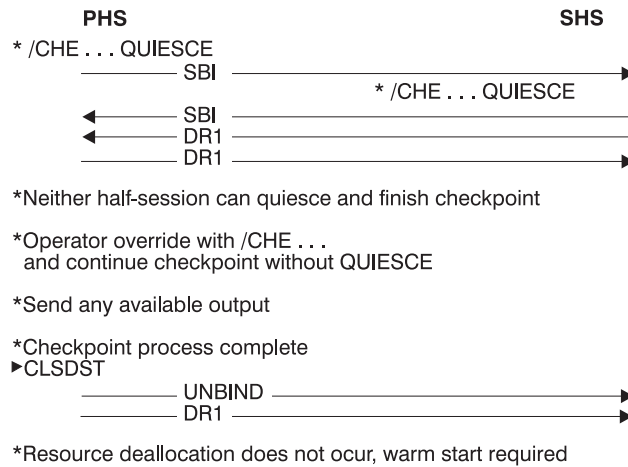
### Example 2: Nonsimultaneous Shutdown Using SBI/BIS by Both Half Sessions



In this example, the SHS BIS had already been sent at the time the SHS /CHE command was entered.

### Example 3: Simultaneous PHS and SHS Shutdown Using SBI/BIS Fails Because of Operator Override

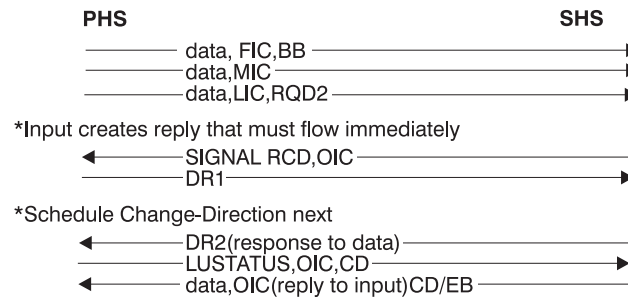
## Signal Protocol Example




---

## Signal Protocol Example

The following example illustrates the use of SIGNAL RCD:



# Appendix I. ISC Error Recovery Procedure Examples

This appendix provides ISC error recovery procedure examples.

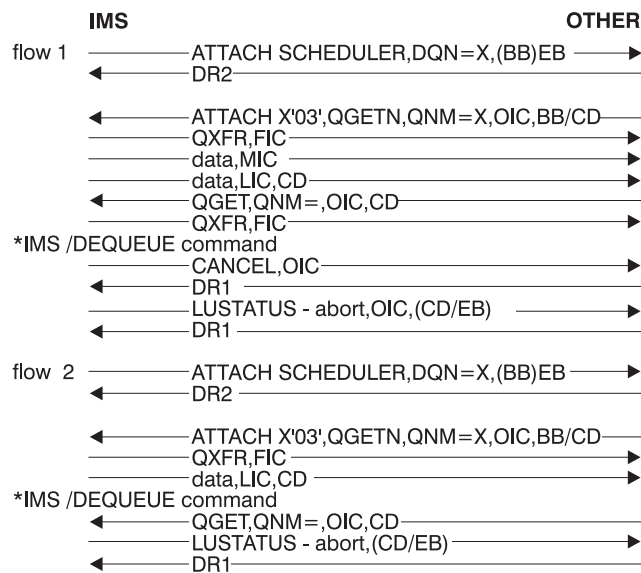
**In this Appendix:**

- “Sender-Detected Error Examples”
- “Receiver-Detected Error Examples” on page 548

## Sender-Detected Error Examples

Following are examples of sender-detected errors during paged output (or input). IMS is the sender in all examples below.

**Example 1: Sender-Detected Error While Sending Demand-Paged Output Message**

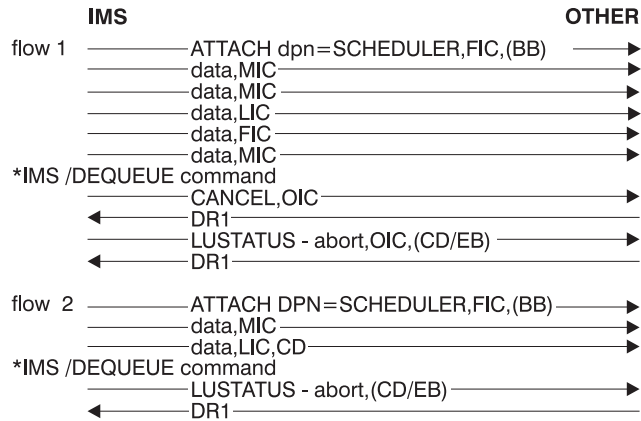


Sample flow 1 shows an IMS /DEQUEUE command occurring during the second output demand page. The CANCEL command terminates the chain (page) in progress, and the LUSTATUS terminates the receiving process. If the paged output message follows another input or output message that began a bracket, the BB is not sent on the first page OIC.

Sample flow 2 is the same, except that the IMS /DEQUEUE command occurs while between output pages so that the CANCEL command is unnecessary.

**Example 2: Sender-Detected Error While Sending Autopaged Output Message**

## Sender-Detected Error Examples



Again, the CANCEL command terminates the chain (page) in progress and the LUSTATUS terminates the receiving process. The initial begin-bracket and the resulting bracket send/receive protocol are the same as for demand-paged output above.

Sample flow 2 is the same as flow 1, except that the IMS /DEQUEUE command occurs while between output pages, so that the CANCEL command is unnecessary.

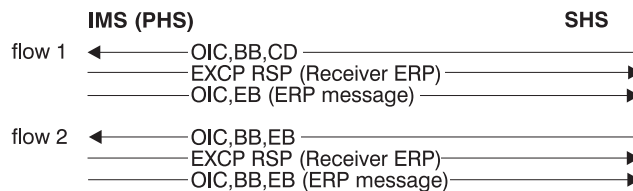
---

## Receiver-Detected Error Examples

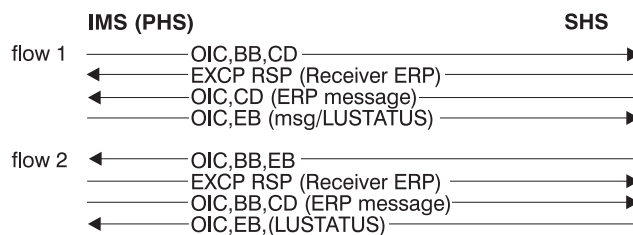
For simplicity in illustration, all examples assume only-in-chain (OIC) messages for flows in both directions. Also the bracket and ERP protocol are considered symmetrical in that either half session can initiate the bracket or ERP procedures as illustrated.

For all session terminations not involving system failures, ATTACH information is recovered and backed up to the last sync point. In the examples that follow, IMS is the receiver.

### Example 1: Receiver ERP for an IMS Component Defined as SINGLE1

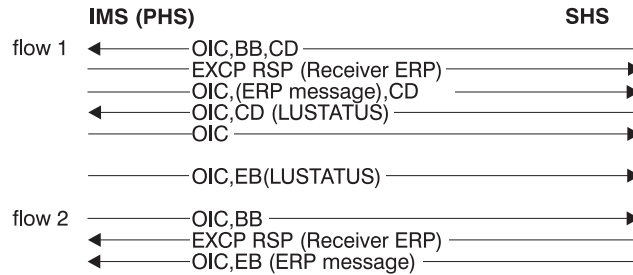


### Example 2: Receiver ERP for an IMS Component Defined as SINGLE2

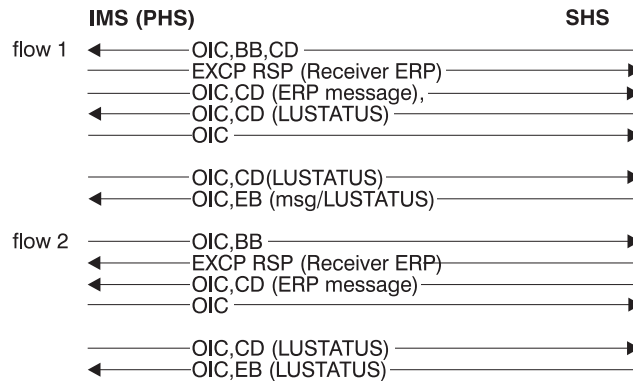


### Example 3: Receiver ERP for an IMS Component Defined as MULT1

## Receiver-Detected Error Examples

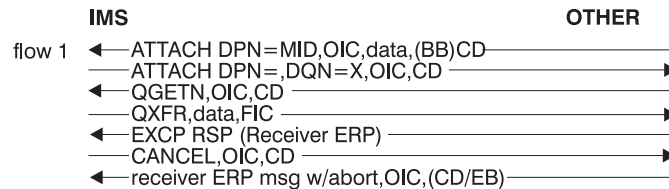


### Example 4: Receiver ERP for an IMS Component Defined as MULT2



### Example 5: Demand-Paged Output with Receiver-Detected Error

In this case, the receiver is the other subsystem. The IMS action follows selective receiver ERP sequence.



IMS action: Return message to queue and continue output.



IMS action: Return message to queue and continue output.

In this example, flow 1 shows the receiver using receiver ERP to reject a demand-paged message. Flow 2 shows alternate methods with the same end result, but not involving an ERP message from the receiver.

## Receiver-Detected Error Examples

---

## Appendix J. Sample Program for IMS-CICS ISC

This appendix provides a sample program that illustrates the use of ISC between IMS and CICS. By combining it with the corresponding sample program in the *CICS/ESA 4.1 Intercommunication Guide*, you can use the ISC function in both products.

The sample program is shipped as object and source code data set DFSISC00. The load library containing the object code is IMS.LOAD. The source library containing the source code is IMS.TMSOURCE.

### **In this Appendix:**

- “Installation Procedure”
- “IMS Sample Program (DFSISC00)”
- “Job Control Statements for the Sample Program” on page 554
- “IMS System Definition Statements” on page 554
- “MFS Formats” on page 555
- “Program Specification Block (PSB) Generation for the Sample Program” on page 556
- “Application Control Block (ACB) Generation” on page 557

---

## Installation Procedure

Before the sample program can be used to perform ISC functions between IMS and CICS, the following steps must be accomplished:

1. Compile and link-edit the sample COBOL program.
2. Define the transaction codes to IMS using the system definition procedure.
3. Define an ISC session between IMS and CICS using the system definition procedure.
4. Compile the MFS utility statements to place the necessary formats into the online library.
5. Perform the PSB generation and then the ACB generation.
6. Initialize the IMS system.
7. Establish a session between IMS and CICS.
8. Run the corresponding CICS sample program.

In this appendix, sample code is provided for performing steps 1 through 5.

---

## IMS Sample Program (DFSISC00)

The following IMS sample program, written in COBOL, accepts an input message from an ISC session with CICS and returns the same message to CICS. The program has two transaction codes—SAMPLA1 and SAMPLA2. When invoked using SAMPLA1, the output message is not formatted by MFS. When invoked using SAMPLA2, the output message is formatted by MFS using distributed presentation management (DPM).

**Related Reading:** To understand the source statements for the MFS utility, see “MFS Formats” on page 555.

## IMS Sample Program (DFSISC00)

```
CBL APOST
  IDENTIFICATION DIVISION.
    PROGRAM-ID. SAMPLA.
    AUTHOR:
  ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  SOURCE-COMPUTER. IBM-370.
  OBJECT-COMPUTER. IBM-370.
  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
  DATA DIVISION.
  FILE SECTION.
    EJECT
  WORKING-STORAGE SECTION.
  77 BEGIN-LIT          PIC X(16) VALUE 'BEGIN 77 ENTRIES'.
  77 FILLER             PIC X(45) VALUE
    '****DATE LAST COMPILED: xxx xx, 19xx****'.
  77 CALL-FUNCTION     PIC XXXX.
  77 MOD-NAME          PIC X(08).
    EJECT
  01 INPUT-AREA.
    02 IN-LL           PIC S9999 COMP.
    02 IN-ZZ           PIC S999 COMP.
    02 TRAN-CODE       PIC X(08).
    02 INPUT-DATA      PIC X(79).
    02 INPUT-DATA-D    REDEFINES INPUT-DATA.
      05 INPUT-DATA-1  OCCURS 3 TIMES
        PIC X(20).
      05 INPUT-DATA-2  PIC X(19).
  01 OUTPUT-AREA.
    02 OUT-LL          PIC S9999 COMP.
    02 OUT-ZZ          PIC S999 COMP.
    02 OUTPUT-DATA     PIC X(79).
  01 OUTPUT-AREA-1.
    02 OUT-LL-1        PIC S9999 COMP.
    02 OUT-ZZ-1        PIC S999 COMP.
    02 OUTPUT-DATA-1   PIC X(20).
    EJECT
  01 DLI-FUNCTIONS.
    02 GET-UNIQ        PIC XXXX VALUE 'GU '.
    02 GET-NEXT        PIC XXXX VALUE 'GN '.
    02 ISRT            PIC XXXX VALUE 'ISRT'.
    EJECT
  01 STATUS-ERROR-SEG.
    02 FILLER          PIC S999 COMP VALUE +83.
    02 FILLER          PIC S999 COMP VALUE +0.
    02 FILLER          PIC X(28) VALUE
    '***** STATUS ERROR *****'.
    02 FILLER          PIC X(10) VALUE ' TRANCODE:'.
    02 ERROR-TRAN      PIC X(8).
    02 FILLER          PIC X(17) VALUE
    ' STATUS RECEIVED:'.
    02 ERROR-STATUS    PIC XX.
    02 FILLER          PIC X(10) VALUE ' FUNCTION:'.
    02 ERROR-FUNCTION  PIC XXXX.
    EJECT
  LINKAGE SECTION.
  01 IOTP-PCB.
    02 IOTP-LTERM      PIC X(8).
    02 FILLER          PIC XX.
    02 IOTP-STATUS     PIC XX.
    02 IOTP-PREFIX.
      03 IOTP-DATE     PIC S9(7) COMP-3.
      03 IOTP-TIME     PIC S9(7) COMP-3.
      03 IOTP-MSG-NUMBER PIC S999 COMP.
      03 FILLER        PIC XX.
    02 IOTP-MOD-NAME   PIC X(8).
```



## IMS Sample Program (DFSISCO0)

```

EJECT

PROCEDURE DIVISION.
  ENTRY 'DLITCBL' USING IOTP-PCB.
100-RETRIEVE-MESSAGE-SEGMENT.
  MOVE GET-UNIQ TO CALL-FUNCTION.
  CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB INPUT-AREA.
  IF IOTP-STATUS = 'QC'
    GO TO 800-GOBACK-ROUTINE.
  IF IOTP-STATUS NOT = SPACES
    GO TO 700-INVALID-STATUS-CODE.
200-CHECK-TRAN-CODE.
*****
*   THE ONLY DIFFERENCE BETWEEN THESE TWO TRANSACTIONS   *
*   IS THE ABSENCE OR PRESENCE OF MFS.  SAMPLA1 DOES NOT  *
*   CONTAIN MFS.  SAMPLA2 CONTAINS MFS.                  *
*****
  IF TRAN-CODE = 'SAMPLA2'
    PERFORM 400-SAMPLA2-ROUTINE
      THRU 450-SAMPLA2-ROUTINE-EXIT,
  ELSE
    PERFORM 300-SAMPLA1-ROUTINE
      THRU 350-SAMPLA1-ROUTINE-EXIT.
  GO TO 100-RETRIEVE-MESSAGE-SEGMENT.
EJECT

300-SAMPLA1-ROUTINE.
  MOVE ISRT TO CALL-FUNCTION.
  SUBTRACT 8 FROM IN-LL GIVING OUT-LL.
  MOVE IN-ZZ TO OUT-ZZ.
  MOVE INPUT-DATA TO OUTPUT-DATA.
  CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB OUTPUT-AREA.
  IF IOTP-STATUS NOT = SPACES
    GO TO 700-INVALID-STATUS-CODE.
350-SAMPLA1-ROUTINE-EXIT.
  EXIT.

400-SAMPLA2-ROUTINE.
  MOVE 'MODA' TO MOD-NAME,
  MOVE +0 TO OUT-ZZ-1.
  MOVE 24 TO OUT-LL-1.
  MOVE INPUT-DATA-1 (1) TO OUTPUT-DATA-1.
  PERFORM 500-INSERT-ROUTINE
    THRU 550-INSERT-ROUTINE-EXIT.
  SUBTRACT 20 FROM IN-LL.
  IF IN-LL IS LESS THAN 13 GO TO 450-SAMPLA2-ROUTINE-EXIT.
  MOVE INPUT-DATA-1 (2) TO OUTPUT-DATA-1.
  PERFORM 600-INSERT-ROUTINE
    THRU 650-INSERT-ROUTINE-EXIT.
  SUBTRACT 20 FROM IN-LL.
  IF IN-LL IS LESS THAN 13 GO TO 450-SAMPLA2-ROUTINE-EXIT.
  MOVE INPUT-DATA-1 (3) TO OUTPUT-DATA-1.
  PERFORM 600-INSERT-ROUTINE
    THRU 650-INSERT-ROUTINE-EXIT.
  SUBTRACT 20 FROM IN-LL.
  IF IN-LL IS LESS THAN 13 GO TO 450-SAMPLA2-ROUTINE-EXIT.
  MOVE INPUT-DATA-2 TO OUTPUT-DATA-1.
  PERFORM 600-INSERT-ROUTINE
    THRU 650-INSERT-ROUTINE-EXIT.
450-SAMPLA2-ROUTINE-EXIT.
  EXIT.
EJECT

500-INSERT-ROUTINE.
  MOVE ISRT TO CALL-FUNCTION.
  CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB OUTPUT-AREA-1
    MOD-NAME.
  IF IOTP-STATUS NOT = SPACES
    GO TO 700-INVALID-STATUS-CODE.

```

## IMS Sample Program (DFSISC00)

```
550-INSERT-ROUTINE-EXIT.  
EXIT.  
600-INSERT-ROUTINE.  
MOVE ISRT TO CALL-FUNCTION.  
CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB OUTPUT-AREA-1.  
IF IOTP-STATUS NOT = SPACES  
GO TO 700-INVALID-STATUS-CODE.  
650-INSERT-ROUTINE-EXIT.  
EXIT.  
700-INVALID-STATUS-CODE.  
MOVE CALL-FUNCTION TO ERROR-FUNCTION.  
MOVE IOTP-STATUS TO ERROR-STATUS.  
MOVE TRAN-CODE TO ERROR-TRAN.  
MOVE ISRT TO CALL-FUNCTION.  
CALL 'CBLTDLI' USING CALL-FUNCTION IOTP-PCB STATUS-ERROR-SEG.  
800-GOBACK-ROUTINE.  
GOBACK.
```

---

## Job Control Statements for the Sample Program

Use the following JCL to compile and link-edit the sample program. This JCL uses the MVS COBOL 2.4 compiler and places the resulting load module in IMS.PGMLIB.

```
//IMSCOBOL JOB ACCT,NAME,CLASS=A,MSGLEVEL=(1,1),MSGCLASS=A  
// PROC MBR=,PAGES=60,RGN=512K,  
// SOUT=A  
//C EXEC PGM=IKFCBL00,REGION=&RGN,  
// PARM='SIZE=130K,BUF=10K,LINECNT=50,APOST,BATCH'  
//SYSLIN DD DSN=&&LIN,DISP=(MOD,PASS),UNIT=SYSDA,  
// DCB=(USER.PROCLIB),  
// SPACE=(3520,(40,10),RLSE,,ROUND)  
//SYSPRINT DD SYSOUT=&SOUT,DCB=(LRECL=121,BLKSIZE=605,RECFM=FBA),  
// SPACE=(605,(&PAGES.0,&PAGES),RLSE,,ROUND)  
//SYSUT1 DD UNIT=SYSDA,DISP=(,DELETE),  
// SPACE=(3520,(100,10),RLSE,,ROUND)  
//SYSUT2 DD UNIT=SYSDA,DISP=(,DELETE),  
// SPACE=(3520,(100,10),RLSE,,ROUND)  
//SYSUT3 DD UNIT=SYSDA,DISP=(,DELETE),  
// SPACE=(3520,(100,10),RLSE,,ROUND)  
//SYSUT4 DD UNIT=SYSDA,DISP=(,DELETE),  
// SPACE=(3520,(100,10),RLSE,,ROUND)  
//L EXEC PGM=IEWL,REGION=&RGN,PARM='XREF,LET,LIST',  
// COND=(4,LT,C)  
//*STEPLIB DD DSN=IMS.RESLIB,DISP=SHR  
//SYSLIB DD DSN=SYS1.COBLIB,DISP=SHR  
//RESLIB DD DSN=IMS.RESLIB,DISP=SHR  
//SYSLIN DD DSN=&&LIN,DISP=(OLD,DELETE),VOL=REF=*.C.SYSLIN  
// DD DSN=IMS.PROCLIB(CBLTDLI),DISP=SHR  
// DD DDNAME=SYSIN  
//SYSLMOD DD DSN=IMS.PGMLIB(&MBR),DISP=SHR  
//SYSPRINT DD SYSOUT=&SOUT,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=605),  
// SPACE=(605,(&PAGES.0,&PAGES),RLSE,,ROUND)  
//SYSUT1 DD UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),DISP=(,DELETE),  
// SPACE=(3520,(100,10),RLSE,,ROUND)  
// PEND  
//IMSCOBOL EXEC IMSCOBOL,SOUT=A,MBR=SAMPLA  
//C.SYSLIN DD *
```

You can also link-edit the sample program directly from IMS.LOAD to IMS.PGMLIB.

---

## IMS System Definition Statements

Use the following statements to define to IMS the two transaction codes supported by the sample program:

## IMS System Definition Statements

```

APPLCTN PSB=SAMPLA
  TRANSACT CODE=SAMPLA1, INQ=(YES, NORECOV)
  TRANSACT CODE=SAMPLA2, INQ=NO
TYPE  UNITYPE=LUTYPE6, OPTIONS=(TRANRESP, OPNDST, NONTOMSG,
      SYNCSESS),
      MSGDEL=SYSINFO,
      OUTBUF=256,
      SEGSIZE=256
X
X
X
X
*
**CICSA1 PARALLEL SESSION NODE
*
  TERMINAL NAME=CICSA1,
      SESSION=5,
      COMPT1=(SINGLE1, DPM-B1, IGNORE),
      COMPT2=(SINGLE2, DPM-B1, IGNORE),
      COMPT3=(MULT1, DPM-B1, IGNORE),
      COMPT4=(MULT2, DPM-B1, IGNORE)
X
X
X
X
X
VTAMPOOL
*
  SUBPOOL NAME=PS01
      NAME PSLT01, COMPT=1, ICOMPT=1
      NAME PSLT02, COMPT=2, ICOMPT=2
      NAME PSLT03, COMPT=3, ICOMPT=3
      NAME PSLT04, COMPT=4, ICOMPT=4

```

**Restriction:** The DPM-Bn specified here must match the DPM-Bn specified on the MFS format DEV statement.

---

## MFS Formats

The statements below comprise the JCL and MFS statements needed to produce the MFS formats used by the sample program. This is shipped as copy code in data set MFSSISCO in IMS.GENLIBB.

```

//MFSUTL JOB ACCT, NAME, CLASS=A, MSGLEVEL=(1,1)
//JOB LIB DD DSN=IMS.RESLIB, DISP=SHR
//      PROC RGN=360K, SOUT=A, SNODE=IMSVS,
//      SOR=NOLIB, MBR=NOMBR, PXREF=NOXREF,
//      PCOMP=NOCOMP, PSUBS=NOSUBS, PDIAG=NODIAG,
//      COMPR=NOCOMPRESS, COMPR2=COMPRESS,
//      LN=55, SN=8, DEVCHAR=0
//S1      EXEC PGM=DFSUPAA0, REGION=&RGN,
//      PARM=(&PXREF, &PCOMP, &PSUBS, &PDIAG, &COMPR,
//      'LINECNT=&LN, STOPRC=&SN, DEVCHAR=&DEVCHAR')
//SYSLIB DD DSN=IMS.MACLIB, DISP=SHR
//SYSIN  DD DSN=&SNODE..&SOR. (&MBR), DISP=SHR
//REFIN  DD DSN=IMS.REFERAL, DISP=OLD
//REFOUT DD DSN=IMS.REFERAL, DISP=OLD
//REFRD  DD DSN=IMS.REFERAL, DISP=OLD
//SYSTEXT DD DSN=&&TXTPASS, UNIT=SYSDA,
//      SPACE=(CYL, (1,1)), DCB=BLKSIZE=800
//SYSUT3 DD UNIT=SYSDA, SPACE=(CYL, (1,1))
//SYSUT4 DD UNIT=SYSDA, SPACE=(CYL, (1,1))
//UTPRINT DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT, DCB=(RECFM=FBA, LRECL=133, BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//SEQBLKS DD DSN=&&BLKS, DISP=(NEW, PASS),
//      UNIT=SYSDA, SPACE=(CYL, (1,1))
//S2      EXEC PGM=DFSUNUB0, REGION=&RGN,
//      PARM='&COMPR2, DEVCHAR=&DEVCHAR',
//      COND=(8, LT, S1)
//SEQBLKS DD DSN=&&BLKS, DISP=(OLD, DELETE)
//UTPRINT DD SYSOUT=&SOUT, DCB=(RECFM=FBA, LRECL=133, BLKSIZE=1330)
//SYSUDUMP DD SYSOUT=&SOUT
//FORMAT DD DSN=IMS.FORMAT, DISP=OLD
//SYSPRINT DD SYSOUT=&SOUT

```

## MFS Formats

```
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
// PEND
//MFSUTL EXEC MFSUTL,SOUT=A
//S1.SYSIN DD *
        PRINT ON,NOGEN
FMTA    FMT
        SPACE 3
        DEV  TYPE=DPM-B1,FEAT=IGNORE,DSCA=X'00A0'
        SPACE 2
        DIV  TYPE=OUTPUT,OPTIONS=(DPAGE,NODNM)
        SPACE 3
DPAGE01 DPAGE FILL=NULL
PPAGE01 PPAGE
        DFLD 'PAGE'
LPGNO   DFLD LTH=04
        DFLD ' '
        SPACE 2
DATA01  DFLD LTH=20
        SPACE 3
        FMTEND
        EJECT
MODA    MSG  TYPE=OUTPUT,SOR=(FMTA,IGNORE),PAGE=YES
        SPACE 3
        SEG
        SPACE 2
        MFLD (LPGNO,LPAGENO)
        MFLD DATA01,LTH=20
        SPACE 3
        MSGEND
        END
/*
```

---

## Program Specification Block (PSB) Generation for the Sample Program

The following JCL and program specification block (PSB) generation statements produce the IMS control blocks necessary to execute the sample program:

```
//PSBGEN JOB ACCT,NAME,CLASS=A,MSGLEVEL=(1,1)
//      PROC  MBR=TEMPNAME,SOUT=A,RGN=2000K
//C      EXEC  PGM=IFOX00,REGION=&RGN,PARM='OBJ,NODECK'
//SYSLIB DD  DSN=IMS.MACLIB,DISP=SHR
//SYSGO  DD  UNIT=SYSDA,DISP=(,PASS),
//          SPACE=(80,(100,100),RLSE),
//          DCB=(BLKSIZE=80,RECFM=F,LRECL=80)
//SYSPRINT DD  SYSOUT=&SOUT,DCB=BLKSIZE=1089,
//          SPACE=(121,(300,300),RLSE,,ROUND)
//SYSUT1  DD  UNIT=SYSDA,DISP=(,DELETE),
//          SPACE=(1700,(100,50))
//SYSUT2  DD  UNIT=SYSDA,DISP=(,DELETE),
//          SPACE=(1700,(100,50))
//SYSUT3  DD  UNIT=(SYSDA,SEP=(SYSLIB,SYSUT1,SYSUT2)),
//          SPACE=(1700,(100,50))
//L      EXEC  PGM=IEWL,PARM='XREF,LIST',COND=(0,LT,C),REGION=120K
//STEPLIB DD  DSN=IMS.RESLIB,DISP=SHR
//SYSLIN  DD  DSN=*.C.SYSGO,DISP=(OLD,DELETE)
//SYSPRINT DD  SYSOUT=&SOUT,DCB=BLKSIZE=1089,
//          SPACE=(121,(90,90),RLSE)
//SYSLMOD DD  DSN=IMS.PSBLIB(&MBR),DISP=SHR
//SYSUT1  DD  UNIT=(SYSDA,SEP=(SYSLMOD,SYSLIN)),
//          SPACE=(1024,(100,10),RLSE),DISP=(,DELETE)
// PEND
//PSBGEN EXEC PSBGEN,SOUT=A,MBR=SAMPLA
//C.SYSIN DD *
        PSBGEN  LANG=COBOL,PSBNAME=SAMPLA
        END
/*
```

### Application Control Block (ACB) Generation

Use the following JCL and application control block (ACB) utility statements to place the PSB into the appropriate IMS online library:

```
//ACBGEN JOB ACCT,NAME,CLASS=A,MSGLEVEL=(1,1)
//ACBGEN EXEC ACBGEN,SOUT=A
//G.SYSIN DD *
BUILD PSB=SAMPLA
BUILD PSB=SAMPLB
/*
```

## Application Control Block (ACB) Generation

---

## Bibliography

This bibliography includes all the publications cited in this book, including the publications in the IMS library.

- *ACF/VTAM System Programmer's Guide*, SC38-0258
- *Advanced Function for Communication System Summary*, GA27-3099
- *BTAM SP Program Summary*, GC27-0599
- *CICS/ESA 4.1 Application Programming Reference*, SC33-1170
- *CICS/ESA 4.1 Intercommunication Guide*, SC33-1181
- *CICS/ESA 4.1 Recovery and Restart Guide*, SC33-1182
- *CICS/ESA 4.1 Resource Definition Guide*, SC33-1166
- *CICS/ESA CICS-Supplied Transactions*, SC33-1168
- *CICS/ESA Customization Guide*, SC33-1165
- *CICS/ESA Operations Guide*, SC33-0668
- *Common Programming Interface Communications Specification*, SC31-6180
- *IBM 3270 Information Display System: 3274 Control Unit Description and Programmer's Guide*, GA23-0061
- *IBM 4700 Finance Communication System: Controller Programming Library, Volume 3: Communication Program*, GC31-2068
- *IBM 4700 Finance Communication System: System Summary*, GC27-2016
- *IBM 4730 Personal Banking Machine General Information*, GC31-2073
- *IBM 4730 Personal Banking Machine Operations Support Manual*, GC31-2560
- *IBM DATABASE 2 Application Programming and SQL Guide*, SC26-4889
- *IMS/ESA Version 4 Extended Terminal Option (ETO) Installation and Planning Guide*, GG24-3980
- *OS/390 MVS Programming: Sysplex Services Guide*, GC28-1771
- *OS/390 V1R1 MVS Planning: APPC Management*, GC28-1807
- *MVS/ESA Planning: Extended Recovery Facility (XRF)*, GC28-1139
- *MVS/ESA Planning: Workload Management*, GC28-1493
- *MVS/ESA Programming: Workload Management Services*, GC28-1494
- *MVS/ESA Programming: Writing Transaction Programs for APPC/MVS*, GC28-1471
- *Network Program Products General Information*, GC30-3350
- *Network Program Products Planning*, SC30-3351
- *SAA CPI Communications Reference*, SC26-4399
- *SAA CPI Resource Recovery Reference*, SC31-6821
- *Screen Definition Facility II General Information*, GH19-6114
- *System/7 Functional Characteristics Manual*, GA34-0003-07
- *System/390 MVS Sysplex Overview: An Introduction To Data Sharing*, GC28-1208
- *Systems Network Architecture Format and Protocol Reference: Architectural Logic*, SC30-3112-02
- *Systems Network Architecture: Sessions between Logical Unit Types*, GC20-1868
- *Systems Network Architecture Technical Overview*, GC30-3073-04
- *VTAM Customization*, LY43-0068
- *VTAM Operation*, SC31-6549
- *VTAM V4R3 for MVS/ESA Programming*, SC31-6550
- *VTAM V4R2 for MVS/ESA Resource Definition Reference*, SV40-1012

---

## IMS/ESA Version 6 Library

SC26-8725	ADB	Administration Guide: Database Manager
SC26-8730	AS	Administration Guide: System
SC26-8731	ATM	Administration Guide: Transaction Manager
SC26-8727	APDB	Application Programming: Database Manager
SC26-8728	APDG	Application Programming: Design Guide
SC26-8726	APCICS	Application Programming: EXEC DLI Commands for CICS and IMS
SC26-8729	APTM	Application Programming: Transaction Manager

SC26-8732	CG	Customization Guide
SC26-9517	CQS	Common Queue Server Reference
SC26-8733	DBRC	Database Recovery Control Guide and Reference
LY37-3731	DGR	Diagnosis Guide and Reference
LY37-3732	FAST	Failure Analysis Structure Tables (FAST) for Dump Analysis
GC26-8736	IIV	Installation Volume 1: Installation and Verification
GC26-8737	ISDT	Installation Volume 2: System Definition and Tailoring
SC26-8740	MIG	Master Index and Glossary
GC26-8739	MC	Messages and Codes
SC26-8743	OTMA	Open Transaction Manager Access Guide
SC26-8741	OG	Operations Guide
SC26-8742	OR	Operator's Reference
GC26-8744	RPG	Release Planning Guide
SC26-8767	SOP	Sample Operating Procedures
SC26-8769	URDB	Utilities Reference: Database Manager
SC26-8770	URS	Utilities Reference: System
SC26-8771	URTM	Utilities Reference: Transaction Manager

#### **Supplementary Publications**

GC26-8738	LPS	Licensed Program Specifications
SC26-8766	SOC	Summary of Operator Commmands

#### **Online Softcopy Publications**

LK3T-2326	CDROM	IMS/ESA Version 6 Softcopy Library
SK2T-0730	CDROM	IBM Online Library: Transaction Processing and Data
SK2T-0710	CDROM	MVS Collection
SK2T-6700	CDROM	OS/390 Collection



# Index

## Special Characters

// as escape sequence (MFS edit) 105  
\*\* for segment discard (MFS edit) 105  
/ASSIGN command  
    cold starting ISC sessions 272  
    limitations with ETO 159  
/CHANGE command, initiating an ISC session 272  
/DEQUEUE command 211, 273, 289  
/DISPLAY command 77, 209, 273  
    QSTOP state 77  
/EXIT command 37, 286  
/IDLE command 207  
/MSASSIGN command 175, 207, 208  
/MSVERIFY command 204, 205  
/OPNDST command 73, 150, 159  
/PSTOP LINK command 206  
/RSTART LINK command 206, 208  
/SIGN command for ETO STSN devices 168  
/START command 75, 431  
/STOP command 38

## Numerics

3270 copy command 109  
3600 terminal, supported as Finance or SLU P 400  
3650 system, supported as SLU P 400  
4700 terminals, supported as Finance or SLU P 400  
4730 Personal Banking Machine, connecting in the XRF environment 408  
8100 system, supported as SLU P 400

## A

abend  
    CICS 536  
    IMS 536  
    MSC conversations 197  
ACK option  
    Finance Communication System 458  
    SLU P 458  
administration  
    ETO 123  
    IMS, in client/server environment 359  
    IMS TM network 7  
    ISC 221  
    MFS 96  
    MSC 193  
    SLU P and Finance communication 399  
affinity, definition 20  
affinity management  
    bypassing 62  
allocating a session, CICS 515  
alternate facility, CICS 520  
alternate PCB, secondary transaction 30  
alternate response PCB  
    message switches, use 241  
    modifying 412  
    use 412

API (application program interface), ISC  
    asynchronous, CICS 497, 518  
    IMS use of VTAM 233  
    synchronous, CICS 497, 518  
APPC  
    Communications Manager as the 366  
    SYNCLVL=SYNCPT 366  
APPC generic resource name, specifying 58  
APPC/IMS (Advanced Program to Program)/IMS 11  
    APPC/IMS application programs 369  
        CPI-C driven 369  
        modified IMS 369  
        standard IMS 369  
    application program interface (API) 11  
        explicit 11, 368  
        implicit 11, 368  
    asynchronous output delivery 393  
    CPI-C initialization 27  
    CPI Communications application program 359  
    default conversation characteristics 393  
    DFSAPPC 390  
        message switching 390  
        option keywords 391  
    discardable messages 368  
    establishing APPC/IMS 374  
        APPC=Y or N 374  
        system definition 374  
    introduction 367  
    local service for remote APPC transaction 370, 372  
    LTERM interface 373  
    LU 6.2  
        relationship to APPC/IMS 367  
        terminal support 27  
    LUADD option keywords 379  
    MOD name 373  
    mode name 378  
    modified IMS application programs 371  
    MSC 382  
        and standard IMS applications 370  
        recovering transactions in APPC 382  
    MVS system file manager utility example (ATBSDFMU) 376  
    network-qualified LU name 389  
    nondiscardable messages 368  
    PARMLIB 378  
    partner LU name 378  
    RACF 394  
    recoverability 77  
    recoverable versus irrecoverable transactions 382  
    remote service for remote APPC transaction 370, 372  
    SIADD 378  
    side information 27  
        mode\_name 27, 377  
        partner\_LU\_name 27, 377  
        TP\_name 27, 377  
    standard IMS application programs 369  
    sym\_dest\_name 378

APPC/IMS (Advanced Program to Program)/IMS 11  
*(continued)*  
 Sync\_Level options (NONE, CONFIRM, SYNCPT) 370, 371  
 SYS1.APPCSI 378  
 SYS1.APPCTP 374  
 SYS1.PARMLIB(APPCPMxx) example 378  
 TP name 378  
 TP Profile 372  
 ATBSDFMU utility 374  
 definition 372  
 DFSTPPE0 376  
 DFSTPROF 375  
 dialog example 375  
 TSO ICQASRM0 375  
 VSAM data set 374  
 TPN (transaction program name) 377, 391  
 transaction retry characteristics 389  
 deadlock 389  
 Fast Path retry conditions 389  
 lock reject 389  
 transaction security 394  
 ACCESS (EXECUTE) 394  
 UACC (NONE) 394  
 XRF 388

APPC/IMS (Application Program to Program)/IMS  
 MSC and modified IMS applications 371  
 application program interface 233  
 application programs  
 abnormal termination and MSC 211  
 CICS, with IMS 517  
 converting Finance to SLU P 404  
 existing programs, with ISC 244, 246  
 functions IMS provides 10  
 ISC sample 551  
 SLU P controller 404  
 XRF considerations, SLU P 403

APPLID= parameter, ISC session definition 260  
 APPLID name, definition 20  
 asynchronous processing  
 ATTACH EB 238  
 IMS-CICS session 504  
 ISC execution mode 238  
 SCHEDULER FM header 325

ATCCONxx member (VTAM nodes) 74  
 ATCSTRyy member (VTAM start lists) 74  
 ATTACH FM header 238  
 bit contents 339, 340  
 EB indicator 238  
 format 339, 347  
 MFS 333  
 parameter description 340, 347  
 process initiation 323  
 with CICS 525

ATTIU parameter, FM header length 324  
 automated operator programs 22  
 autopaged output, synchronous 500

**B**  
 back-end subsystem  
 CICS 503  
 back-end subsystem *(continued)*  
 definition 223  
 IMS 501, 502  
 routing transactions 256  
 backout work unit, message resynchronization 270  
 balancing group  
 definition 20  
 routing codes and 21  
 balancing resource demand in MSC 194  
 basic edit 94  
 bypassing 104  
 editing options with ISC 102  
 IMS functions 101  
 IMS systems 94  
 input message segments 105  
 input messages 101  
 ISC messages 236  
 non-MFS programs 245  
 output message segments 105  
 output messages 102  
 SLU 1 transparent data 102

BB (begin bracket) indicator  
 definition 243  
 LUSTATUS command 308  
 use (figure) 408

BID command 295, 416, 426  
 BID option  
 caused by session termination 461  
 design considerations 416, 426  
 effects  
 display screen protection 421  
 MFS paging 421  
 output messages 418

bind  
 action/response matrix 276  
 IMS-CICS session 533  
 ISC session  
 parallel 269  
 single 269  
 negotiable 269  
 nonnegotiable 269  
 parameters  
 Finance Communication System 471  
 ISC, IMS as primary half session 473  
 ISC, IMS as secondary half session 476  
 SLU 1 481  
 SLU P 471  
 rejected, ISC 316  
 requesting asynchronous process 325

BIND race 269  
 BINPDSB1= parameter, BINTRNDS option 102  
 BIS (bracket initiation stopped) command 320  
 IMS-CICS session 516  
 session shutdown 320

bracket and send/receive management  
 Finance Communication System  
 direction indicators 408  
 protocol 408  
 ISC, how determined 242  
 SLU P  
 direction indicators 408

- bracket and send/receive management (*continued*)
  - protocol 408
- bracket contention
  - invalid paging 425
  - resolving 269, 424
- bracket initiation stopped (BIS) command 320
- bracket protocol
  - IMS 296, 441
  - input bracketing 446
  - input messages, ISC 296
  - output bracketing 453
  - output messages, ISC 299
- bracket rejection
  - Finance Communication System 433, 461
  - ISC 317
  - SLU P 433, 461
- BREAK code X'0811'
  - actions taken 460
  - output messages 460
- buffer
  - MSC considerations 193
  - MSC linking 173
- buffer sizes
  - BUFSIZE parameter 199
  - IMS-to-IMS sessions 256
  - ISC 262
  - MSC LU 6.2 application transactions 199
  - MSC non-VTAM links 199
  - MSC VTAM links 199
- BUFSIZE parameter for MSC LU 6.2 199

## C

- CANCEL command
  - paging errors 311, 318
  - protocol 304
  - sender ERP 316
  - SLU P session 462
- candidate printers 108
- CD (change direction) indicator 296
  - definition 243
  - Finance Communication System 406
  - LUSTATUS command 308
  - send/receive protocol 408, 441
  - SLU P 406
  - soliciting 292
- chained message communication sequence 447
- chains, logical terminal 33
- CHANGE command, initiating an ISC session 272
- channel-to-channel (CTC) link in MSC 173
- character string controls 487
- CHASE command 305
- CICS-IMS communication
  - alternate facility 520
  - application coding for 517
  - asynchronous processing flow 504
  - ATTACH parameters 525
  - CICS transactions 515
  - coding function management headers 525
  - coding system generation options 508
  - Command Level API 497
  - configuration 226

- CICS-IMS communication (*continued*)
  - conversation mode 520
  - defining CICS transactions 515
  - device mapping function 226
  - facility
    - alternate 520
    - principal 520
  - functions
    - description 497, 499
    - overview 228
  - IMS commands 521
  - initiating sessions 515
  - integrity of session 533
  - LU 6.1 links
    - compatible nodes 509
    - description 508
    - Macro-Level Resource Definition 508
    - multiple links 513
    - Resource Definition Online 508
  - MFS support 530
  - mirror transaction 504
  - MSC links 227
  - passing data to IMS with ISC 229
  - preparing CICS tables 508
  - principal facility 520
  - processing flows
    - RECEIVE 503
    - RETRIEVE 505
    - SEND INVITE 501
    - SEND LAST 502
    - SEND/RECEIVE 501
    - START/RETRIEVE 505
    - synchronous 501
  - recovery and restart 531
  - SCHEDULER parameters 528
  - session
    - binding 533
    - initiation 515
    - integrity 533
    - processing outstanding traffic 534
    - reestablishing 533
    - resynchronizing 533
    - sync points 522
    - termination 516
  - sync points 522
  - transaction
    - abnormal termination 518, 536
    - attributes supported 499, 518
    - definition 515
- CICS-ISC installation options 507
- CICS resource definition 508
- Class 1 terminals
  - overview 65
  - XRF support 65
- Class 2 terminals
  - overview 66
  - XRF support 65
- Class 3 terminals
  - overview 66
  - XRF support 65

- coding CICS
  - applications for ISC 517
  - system generation macros 508
  - tables 508
- cold start
  - general considerations 206
  - message integrity 210
  - MSC 206, 210
- cold start, recovering ISC sessions 272
- COMM macro statement 260
  - APPLID= keyword 70
  - PASSWD= keyword 70
  - RECANY= keyword 71
- Command Level API, CICS 497
- commands
  - affecting MSC operation 207
  - asynchronous reply 500
  - CICS EXEC
    - asynchronous API 497
    - creating DFC protocols 497
    - functions available 499
    - program flow 500
    - synchronous API 497
  - IMS
    - parallel ISC sessions 236
    - start system 75
  - issuing from an ISC session 236
  - keywords 207
  - LUSTATUS protocol, ISC 311
  - PRNs, not specified on IMS commands 345
  - recovery at ISC session failure 275
  - transaction reply 502
  - VTAM
    - BID 416
    - BIS 320, 516
    - CANCEL 304, 462
    - CHASE 305
    - commands and indicators 232, 405
    - LUSTATUS command 308, 461
    - LUSTATUS with CICS 500, 521
    - ready to receive (RTR) 312
    - Request-Recovery 463
    - RSHUT 312
    - RTR 462
    - SBI 320, 516
    - SIGNAL 320, 463
- commit
  - LUSTATUS command 310
  - work unit 270, 531
- Common Queue Server (CQS) 15
- communication controllers 10
- communications, establishing 14
- communications network 7
  - components
    - illustration 8
    - responsibilities 8, 20
  - Finance Communication System 400
  - SLU P system 400
- COMPINOP state 39
- component definition
  - LTERM naming 240, 411
- component definition (*continued*)
  - parameter definitions 300
  - selection
    - input component 241, 411
    - output component 243, 244, 412
    - SLU P input component 412
  - SLU P system 411
- component protection
  - extended output 422
  - state 39
- configurations
  - Finance Communication System 400
  - ISC 226, 228
  - SLU P system 400
  - VTAM 74
- connecting multiple IMS systems 193
- connection
  - description 233, 407
  - IMS session parameters
    - ISC 267, 268
    - restriction against altering 407
  - XRF complex, establishing connections 408
- contention, bracket 269, 425
- control block 98
  - ISC 237
  - MFS 98
- control character 106
- control function, MFS 93
- controlling output 107, 110
- Conversation Abnormal Termination exit routine 191
- conversation mode
  - errors
    - detection 285
    - ISC 285, 287
  - explained 37
  - IMS-CICS 520
  - normal termination, ISC extension 287
  - restriction 256
- conversational processing 31
- conversational processing in MSC 196, 197
  - abend 197
  - planning for 194
  - remote destination verification 196
- conversational transactions 31
- coordinating performance information for MSC 212
- coupling facility
  - definition 15
- CPI (Communications Programming Interface) 27
  - communications initialization 27
  - transactions 29
- CPI Communications application program 359
  - abnormal termination 360
  - APSB call 359
  - ATBCMTP verb 360
  - Backout call 361
  - backout processing 360
  - Commit call 359
  - commit processing 359
  - DB2 plan name, use of 359
  - ESS Attach Facility 359
  - in-doubt unit of recovery, definition 364

CPI Communications application program 359  
 (continued)  
 in-flight unit of recovery, definition 364  
 normal termination 360  
 programming requirements 362  
 pseudonym files 362  
 recovery 362  
 resolve-in-doubt processing 361  
 resource translation table (RTT) 359  
 return codes 361  
   RR\_BACKED\_OUT 361  
   RR\_OK 361  
   RR\_PROGRAM\_STATE\_CHECK 361  
 session failure 361  
 SQL calls 359  
 SRRRCMIT 359  
 system restart 361  
 two-phase commit process and 363  
 CQS (Common Queue Server) 15  
 checkpoint data set 17  
 client, definition 15  
 definition 15  
 overview 18  
 cross-system queuing determination 213  
 CTC link in MSC 173

## D

data descriptor FM header  
 IMS use 530  
 input format 347  
 output format 348  
 use on input 334  
 use on output 334  
 data flow 30  
 message switching 32  
 MSC 179  
 program-to-output terminal 30  
 program-to-program 30  
 terminal-to-program 30  
 data flow control (DFC) protocols 285  
 data flow reset, handling errors 316  
 data partitioning, MSC 176  
 data-set-full condition 108  
 data stream profile, ATTACH FM header 341  
 data transmission  
 definite responses, ISC 231  
 error handling 459  
 error recovery procedure 531  
 exception responses, protocol  
   CICS-IMS 521, 531  
   Finance Communication System 405  
   ISC 231  
   SLU P 405  
 messages  
 input 441  
 output 447  
 response requirements  
 IMS commands and indicators 232, 405  
 IMS message switches 293, 458  
 irrec overable-inquiry transactions 293  
 irrecoverable-inquiry transactions 457  
 data transmission (continued)  
 response requirements (continued)  
   LU 6.2 application program 77  
   MFS control requests 459  
   recoverable-inquiry transactions 293, 457  
   summary 455  
   VTAM commands and indicators 232, 405  
   SLU P, message data types 441  
 data transparency 235  
 data types, IMS advanced function network 441  
 database division 194  
 DBFHAGU0 (Input Edit/Routing exit routine) 21  
 DCCTL (DC Control), overview 13  
 deadlock 211  
 DEADQ STATUS 159  
 deblocking algorithm, ATTACH FM header 342  
 deferred program switch 32  
 defining  
   CICS 507  
   IMS 257  
   ISC node 257, 507  
 definite responses  
 Finance Communication System 405  
 response requirements (figure) 455  
 SLU P 405  
 DEQUEUE command  
 committing ISC output messages 289  
 for MSC 211  
 FORCESS versus SYNCSESS 273  
 IMS-to-IMS ISC session 273  
 dequeuing messages, implications 272  
 descriptor  
 generic logon, definition 118  
 group logon, definition 118  
 logon, definition 118  
 MFS device 119  
 MSC 118  
 specific logon, definition 118  
 user 118  
 destination process name 343  
 destination queue name, ATTACH FM header 347  
 destination system (MSC) 182  
 device class control 106  
 device input format (DIF) 99  
 device output format (DOF) 99  
 DFC (data flow control) protocols 285  
 bracket and half-duplex 296  
 bracketing messages 296  
 input 296  
 output 299  
 chaining 304  
 commands 295  
 BID 295  
 BIS 320  
 CANCEL 304  
 CHASE 305  
 LUSTATUS 308, 311  
 RSHUT 312  
 RTR 312  
 SBI 320  
 SIGNAL 320

DFC (data flow control) protocols 285 *(continued)*  
 ERP PURGE 306  
 error handling 286  
   conversation mode 286  
   response mode 286  
   selective receiver ERP 312  
 examples 539  
 exception response 286  
 half-session synchronization 287  
 input messages, backing out 288  
 paged message errors 311  
 PURGE after exception response 305  
 recoverability aided by LWA 287  
 response requirements 287  
   irrecoverable-inquiry transactions 287  
   recoverable-inquiry transactions 287  
 sense codes 319  
 symmetrical session shutdown 320  
 sync point and response 288  
   availability 288  
   CICS-IMS 522  
   exceptions for synchronous input 289  
   input 292  
   irrecoverable messages 291  
   MFS output messages, ISC 290  
   output 293  
   recoverable messages 290  
   requested on input 288  
   requested on output 289  
 DFS3650 (session status message) 72  
 DFSAPPC message switching 390  
 DFSCMLR0 exit routine 201  
 DFSCMLR1 exit routine 201  
 DFSCMPR0 exit routine 201  
 DFSCMTR0 exit routine  
   capabilities 201  
   relationship to DFSNPRT0 exit routine 190  
 DFSCMUX0 exit routine 202  
 DFSFEBJ0 223  
 DFSMSxxx control block 198  
 DFSNPRT0 exit routine 190  
 DFSSIML0 exit routine 110  
 DIF (device input format) 99  
 direct-control subsystem 518  
 directed routing  
   MSC-directed routing 191  
 Directed Routing 205  
   password not passed across link 205  
 directed routing in MSC 190  
 DISPLAY command 77  
   MSC 209  
   pending ISC output 273  
 display screen protection  
   BID option 421, 422  
   definition 421, 422  
   Finance Communication System terminals 421  
   MFS 421, 422  
   NOBID option 421, 422  
 Distributed Presentation Management (DPM) 104  
 distributed transaction processing, ISC 223, 246

documentation  
   exit routines 202  
   terminal profiles 69  
 DOF (device output format) 99  
 DPM (distributed presentation management)  
   FM headers 330  
 DPM (Distributed Presentation Management) 104  
   option 104  
   OPTIONS=DPAGE or PPAGE 330, 449  
   output 449  
   SLU P supports 418  
 DPN (destination process name)  
   ATTACH FM header 343  
   message routing, ISC 246  
   SCHEDULER FM header 343  
 DR2 response  
   exception  
     Fast Path 428  
     irrec overable-inquiry transactions 455  
     irrecoverable-inquiry transactions 455  
     MFS output 455  
     nonrecoverable output 413  
     when MOD does not specify PAGE=YES 439, 456  
   requirements  
     MFS paged output messages, not Fast Path 413  
     normal IMS output messages 413  
     recoverable-inquiry transaction 457  
     recoverable transaction output 413  
 DSCA operand, DEV statement 110  
 duplicate databases 194  
 dynamic allocation VTAM subpools 240

## E

EB (end bracket) indicator  
   definition 243  
   LUSTATUS command 308  
   use (figure) 408  
   used with ATTACH 238  
 edit  
   basic edit during ISC 236  
   MFS  
     CICS 500  
     during ISC 235  
   options  
     data communication exit routine 236  
     input and output 235, 236  
     ISC input and output, list 225  
 editing facilities  
   invoking FM headers 323  
   ISC overview 235, 236  
 editing messages  
   basic edit and nongraphic messages 105  
   bypassing Basic or MFS editing 104  
   editing performed by IMS 102  
   output segments 104  
   transparency option 103  
 editing options  
   COMPTn parameter of TERMINAL macro 262  
   MFS-SCS1 487  
   OUTBUF parameter of TERMINAL macro 262, 424

editing options *(continued)*  
   SLU P, COMPTn parameter of TERMINAL  
     macro 423  
     types  
       DPM-An 423, 450  
       MFS-SCS1 423  
       SCS1 423  
 EMH (expedited message handler), described 20  
 EMH buffer 62  
 EMHL control region initialization parameter 63  
 end bracket indicator 238, 408  
 ERP (error recovery procedure)  
   CICS-IMS session 531  
   extended 531  
   FM header 324  
   function management header 531  
   implemented by IMS 405  
 ERP FM header, format 349  
 ERP PURGE, after ISC exception response 305  
 error handling 459  
   BREAK code X'0811' 460  
   CANCEL command 304  
     use 462  
     when sent by IMS 462  
   controller-detected errors  
     system sense field 459  
     use 460  
     user sense field 461  
   error messages 331  
   FM header 324  
   IMS-detected errors 459  
   IMS-issued error messages in XRF complex 459  
 ISC  
   CICS-IMS session 531  
   paging errors, during data flow reset state 316  
   paging errors, for nonpaged messages 318  
   paging errors, for paged messages 311  
   selective receiver ERP 312  
   sender ERP 316  
   sender ERP sense codes 317  
   length of message 324  
   LUSTATUS command 309, 461  
   MFS-detected errors 331  
   queuing messages 324  
   RQR command 463  
   SIGNAL command 463  
   SLU P 459  
   VTAM logical unit status command 521  
 error messages  
   conditions causing  
     BB-only specified on input 446  
     BB-only specified on output 453  
   length 324, 460  
   MFS-detected errors 331  
 error recovery procedure (ERP) 324  
 errors  
   ISC session termination, causing 316  
 establishing connection 233  
 ESTAE process  
   bypassing affinity management during 62  
 ETO (Extended Terminal Option) 12 *(continued)*  
   /SIGN command for ETO STSN devices 168  
   3275 devices 129  
   3600/Finance 168  
   ABENDU0015 149  
   advantages 123  
     availability 123  
     LTERMs 123  
   algorithm 140, 155  
     logon descriptor 140  
     LTERM allocation 155  
   associated printing techniques 156  
   asynchronous output 164  
   autologoff 161  
   autologon 163  
   autosignoff 160  
   benefits of using 115  
   commands that reset status and release control  
     blocks 156  
   commands that retain status 155  
   common logon descriptors 138  
   control blocks, creation and deletion 120  
   conversations 32, 169  
   creating descriptors 137  
   customizing 119, 124  
   dead-letter queue 164  
   default CINIT/BIND user data formats 150  
   deleting control blocks 166  
     after logoff 166  
     after signoff 166  
   delivering output to non-originating terminal 165  
   descriptors 136, 138, 142, 145, 148, 149  
     added 149  
     deleted 149  
     generic logon, definition 118  
     group logon, definition 118  
     introduction 136  
     logon 138  
     logon, definition 118  
     MFS 145  
     MFS device, definition 119  
     MSC 148  
     MSC, definition 118  
     specific logon, definition 118  
     updated 149  
     user 142  
     user, definition 118  
     VTAM TERMINAL macro 138  
   descriptors, definition 117  
   descriptors, using 119  
   device characteristics table 130, 145  
   device type, defining 128  
   DFS2085 152  
   DFS3641W 149  
   DFS3645 152  
   DFS3649A 153  
   DFS3650 154  
   DFS3672 152  
   DFSINSX0 145  
   DFSSGNX0 145  
   DFSUSER descriptor 144

ETO (Extended Terminal Option) 12 *(continued)*

- DLQT 164
- dynamic terminal 12
- dynamic terminal, definition 116
- dynamic user, definition 116
- exit routines, coding 148
- exit routines, list of 148
- exit routines, using 119
- guideline selection 139
  - logon descriptors 139
  - LOGOND parameter 139
- initialization 149
  - descriptor validation 149
  - DFSINTX0 149
- logon 138
  - /OPNDST command 150
  - descriptor format 138
  - INITOTHER 150
  - INITSELF 150
  - signon data 150
  - USS LOGON 150
- LTERM with specific destination 154
- LU 2
  - devices 129
  - screen size and model information 130
- LU 6.1 (ISC) terminals 167
- MFS 135
- MFS device characteristics table 130, 145
- MFSUCT utility 130
- MODETBL on ETO logon descriptor 73
- MSC 136
  - descriptor 136
  - descriptor format 148
  - MSNAME macro 136
  - support 136
- multiple signons 152
- node user descriptor 143
- non-SNA 3270 devices
  - printers and displays 129
  - screen size and model information 130
- NTO devices 130
- overview 115
- queue (dynamic user message), definition 116
- RACF and 12
- recommendations 144
- requirements 124
- response mode 170
- screen definitions examples (non-SNA 3270) 132
  - display (model specified) 133, 134
  - display (screen size specified) 133
  - LU0 video 132
  - model 2 printer 133
- screen size control byte 130
- security 134
  - number of profiles 42
  - RACF 134
  - Security Maintenance Utility (SMU) 134
  - signon 134
- shared printing 158
- signing off, definition 165
- signing on, definition 151

ETO (Extended Terminal Option) 12 *(continued)*

- signon 134
  - ETO terminals 134
  - LTERM allocation 151
  - static terminals 134
- SLU P 168
- SMU, nonsupport of 12
- SNA commands 150
- special processing modes 155
- starting ETO 149
- static terminal, definition 116
- storing descriptors 137
- structure
  - definition 116
  - terminal, definition 116
  - user, definition 116
- structure, creation and deletion 119
- STSN terminals 167
  - support for /SIGN command 168
- system definition 137
- system generation time 12
- terminal, definition 116
- terminal-LTERM relationship 154
- terminology 116
- undeliverable data, dead-letter queue 164
  - /DISPLAY STATUS USER command 165
  - /DISPLAY USER DEADQ command 165
- user descriptors 142
  - DFSUSER 144
  - format 142
  - installation-created 142, 143
  - node user descriptor 142, 143
- VTAM CINIT LUNAME 140
- VTAM considerations 128
  - logon CINIT session control blocks 128
  - VTAM PSERVIC parameters 128
- examples, ISC data flow control 539
- exception response
  - DFC command, purging 285
  - ISC response mode errors 286
- protocol
  - response requirements (figure) 455
  - VTAM facilities 231
- exclusive mode 38
- execution mode
  - IMS-CICS communication 499
  - recovery at ISC session failure 275
  - specification, ISC 238, 240
  - supported by ISC, list 224
- EXIT command
  - ISC conversation mode errors 286
  - terminating a conversation abnormally 37
- exit routines
  - for ETO 148
  - for MSC 201
- expedited message handler (EMH), described 20
- expedited message handling 20
- extended output component protection 422
- extended recovery facility (XRF) 408
- Extended Terminal Option (ETO) 12
- external execution mode, ISC 238



external subsystem, definition 12  
extracting multiple system transaction statistics 215

## F

### facilities

ISC 235  
test mode for ISC 237

### failure

CICS-IMS session 532  
ISC, recovering from in-bracket 273  
system, during CICS-IMS session 532  
while in-brackets 271, 276

### Fast Path

EMH buffer size 62  
facilities 427  
Input Edit/Routing exit routine (DBFHAGU0) 21  
input message 413  
MFS NEXTMSG or NEXTMSGP control  
commands 413  
MFS paged output messages 413  
NFPACK option on Terminal macro 428, 429  
output messages 413  
parameters  
FORCERESP 427  
FPACK/NFPACK 427  
OPTACK 427  
TRANRESP 427  
recoverable-inquiry transactions 77, 456  
restrictions for applications 22  
routing code 21  
RTR command 428  
TERMINAL macro  
options 427  
required parameters 427  
terminal-response mode 413  
terminals 62

### Finance Communication System

application program  
converting Finance to SLU P 404  
functions available 402  
MFS considerations 402  
TERMINAL macro 402  
bracket and send/receive protocols 408, 441  
IMS facilities 411  
major parts 400  
sample configuration 400  
session parameters 471  
system takeover considerations 408  
terminals supported 400  
VTAM  
commands and indicators 405  
facilities 404  
workstations 401  
XRF complex, establishing connections 408

### first speaker

ISC contention winner 234  
secondary logical units 234

### FM header 105

### format, message

control characters in message prefix 106  
creating with SDF II 99

### format, message (*continued*)

transparency option 103

### forms 107

FPACK/NFPACK option, Fast Path 427

### front-end subsystem

#### CICS

SEND LAST command 502  
SEND/RECEIVE command 501  
START/RETRIEVE command 505

function of 223

IMS 503, 506

Front-End Switch exit routine (DFSFEBJ0), special support 223

### function abort

detecting loop, ISC 313, 318  
indicated on LUSTATUS command 309

### function management (FM) headers 105

coding for use with CICS 525  
Finance Communication System 443, 451

IMS, data descriptor 530

input FM header length 324

ATTACH FM header 324

output FM header 450

input message 235

input process names 346

inserted by IMS 105

ISC 321

ATTACH 323

ATTACH, MFS 333

data descriptor 334, 347

DPM messages, MFS 330

error recovery procedure 324, 531

header format 339, 357

initiating a process 323

introduction 321

invoking ISC edit 323

MFS 329, 338

QMODEL 335, 338, 349, 355

QMODEL, CICS 530

RAP (reset attached process) 324

RAP (reset attached process), MFS 338

SCHEDULER 325, 528

SCHEDULER, MFS 333

supported by IMS, summary 322

synchronous processing 323

SYSMSG 326

system message 326, 530

ISC routing 222

message descriptor byte format 444

input 444, 445

output 450, 452

MFS 530

output message 238

output process names 346

processing mode requested 238

security 225

SLU P 444, 451, 453

type X'42' (SLU P) 446

### functions

ISC and CICS 228, 497

ISC versus MSC 222

## G

- generic resource affinities
  - managing with IMS 57
  - managing with VTAM 58
- generic resource groups 19
  - definition 20
  - overview and benefits 19
- generic resource member, definition 20
- generic resource name
  - definition 20
  - planning for 58
- generic resources
  - planning for 56
  - requirements for using 57
  - restrictions on using 57
- GRAFFIN command 57, 58
- GRAPHIC= parameter
  - NO option 104
  - SEG statement 104
- GRESTAE command 62

## H

- half-duplex protocol, IMS use 296
- half session
  - definition 266, 269
  - pairs 263, 269
  - synchronization, ISC 287
- HANDLE CONDITION, CICS 516
- header
  - input message 105
  - ISC 105
  - output message 105
  - type X'42'
    - component identification 446
    - data bytes 452
    - message descriptor byte 445, 452
- HIOP (high input/output pool)
  - VTAM output buffers
    - RECANY execution parameter 71
    - RECASZ execution parameter 71
- horizontal partitioning in MSC 176

## I

- IDLE command 207
- immediate program switch 32
- immediate session termination 279
- IMS-CICS communication
  - alternate facility 520
  - application coding for 517
  - asynchronous processing flow 504
  - ATTACH parameters 525
  - CICS transactions, definition 515
  - coding
    - function management headers 525
    - system generation options 508
  - facility
    - alternate 520
    - principal 520
  - IMS-CICS ISC 515

## IMS-CICS communication (*continued*)

- IMS commands 521
- initiating sessions 515
- integrity of session 533
- LU 6.1 links
  - compatible nodes 509
  - description 508
  - Macro-Level Resource Definition 508
  - multiple links 513
  - Resource Definition Online 508
- MFS support 530
- preparing CICS tables 508
- principal facility 520
- processing flows
  - RECEIVE 503
  - RETRIEVE 507
  - SEND INVITE 501
  - SEND LAST 502
  - SEND/RECEIVE 501
  - START/RETRIEVE 505
  - synchronous 501
- recovery and restart 531
- SCHEDULER parameters 528
- session
  - binding 533
  - initiation 515
  - processing outstanding traffic 534
  - reestablishing 533
  - resynchronizing 533
  - sync points 522
  - termination 516
- sync points 522
- terminating a session 516
- transaction
  - attributes supported 499, 520
  - types supported 499, 520
- IMS commands
  - CICS-IMS session 521
  - response requirements 458
  - used to start system 431
- IMS execution modes 499
- IMS facilities
  - available on IMS-CICS session 497, 518
  - component definition 411, 412
  - display screen protection 421, 422
  - Finance Communication System 411
  - Message Format Service 418
  - message recovery 419
  - SLU P 411
  - terminal-response mode 413
- IMS.FORMAT, output from MFS 94
- IMS message switches, response requirements 293, 458
- IMS Monitor
  - MSC considerations 212
  - Report Print Program 213
- IMS.PROCLIB 66, 198
- IMS-to-IMS communication, LU 6.1 protocols 255, 257
- IMS-to-IMS sessions, ISC protocol restrictions 257
- IMS transaction abend 536
- IMSCTRL macro, MSC system definition 198

- IMSMSV procedure 198
- initiation, session 75
- INOP state 39
- input bracketing
  - replies 446
  - rules 446
- input component
  - identification for header type X'42' 446
  - relationship to output component 240
  - versus output component 411
- input editing option, in ISC 235
- input editing option, SLU P 423
- input errors, IMS detection of ISC 315
- input message
  - backing out ISC 288
  - bracketing 444, 446
  - chained messages 304, 447
  - Fast Path 427
  - Finance Communication System chaining
    - indicator 443
  - header
    - DPM 445
    - Finance format 443
    - MID name location 443, 445
    - optional MFS fields 445
    - SCS 445
    - SLU P format 444
    - use 443, 445
  - ISC bracketing 296
  - message descriptor byte
    - header type X'42' 445
    - MID name indicator bit 444, 445
  - MFS input formatting, activating 329, 446
  - multiple transmissions for one message 447
  - origin of for MFS purposes 447
  - requirements 443
  - response requirements 455
  - SLU P
    - chaining indicator 443
    - restriction 446
  - sync point
    - availability 288
    - ISC 288
- input message, response requirements 293
- Input Message Routing exit routine (DFSNPRT0) 190
- input system (MSC) 182
- integrity
  - IMS-CICS session 533
  - message 533
    - IMS-CICS session 533
    - in ISC 225
    - NOCHECK PROTECT 505
- interchange unit code, ATTACH FM header 341
- intermediate system (MSC) 182
- internal execution mode, ISC 238
- Intersystem Communication (ISC) 221
- invalid destinations in MSC 191
- irrecoverable-inquiry transactions
  - LU 6.2 application program 77
  - response requirements 77, 457
- irrecoverable messages, ISC sync points 291
- irrecoverable transactions 257
- ISC (Intersystem Communication) 221
  - application program 244, 551
    - accessing 244, 246
    - example 551
    - not using MFS 245
    - using MFS 245
  - basic functions 221, 222
  - binding an ISC session 268
  - CICS and IMS functions 228
  - CICS-IMS 497, 538
  - Class 2 terminal 225
  - configurations 226, 228
  - control block storage, parallel sessions 237
  - conversation mode termination extension 287
  - data flow control 221
  - DFC protocols 295
    - availability 288
    - backing out input messages 288
    - BID command 295
    - bracket and half-duplex 296
    - bracketing input messages 296
    - bracketing output messages 299
    - CANCEL command 304
    - chaining 304
    - CHASE command 305
    - conversation mode errors 286
    - exception response 285
    - half-session synchronization 287
    - LUSTATUS 308
    - paged message errors 311
    - recoverability supplemented by LWA 287
    - response mode errors 286
    - response requirements 287
    - RSHUT command 312
    - RTR command 312
    - selective receiver ERP 312
    - sync point, input messages 288
  - DFC sync point 288
    - exception 289
    - irrecoverable messages 291
    - MFS messages 290
    - on output 289
    - recoverable messages 290
  - distributed processing 223
  - distributed services, support for 224
  - editing facilities
    - invoking FM headers 323
    - overview 235, 236
  - editing options 102
  - error handling
    - MTO notified 313
    - sender ERP 316
  - error recovery, purge after exception response 305
  - execution mode 224, 238
    - external specification 238
    - internal, with CICS 499
    - internal definition 238
    - processing mode 239
    - synchronous versus asynchronous 239
  - FM headers 246, 321, 525

ISC (Intersystem Communication) 221 *(continued)*

- ATTACH 323
- ATTACH, MFS 333
- CICS session 525
- data descriptor 334
- DPM messages, MFS 330
- error recovery procedure 324
- header format 339, 357
- initiating a process 323
- introduction 321
- invoking ISC edit 323
- MFS 329, 338
- QMODEL 335, 338
- RAP (reset attached process) 324
- RAP (reset attached process), MFS 338
- routing messages 246
- SCHEDULER 325
- SCHEDULER, MFS 333
- supported by IMS 322
- synchronous processing 323
- SYSMSG 326
- half session, primary versus secondary 268
- IMS commands, issuing 236
- IMS facilities 223, 225, 235, 242
- IMS support, CICS 497
- IMS support for 223
- IMS-to-CICS configuration 226
- IMS-to-IMS configuration 227
- IMS-to-IMS session 255
  - buffer sizes 256
  - defining parallel session 255
  - defining single session 255
  - dequeuing messages 273
  - design considerations 255, 257
  - limit of parallel sessions 255
  - MSC and ISC coexistence 255
  - protocol restrictions 257
  - remote control 256
  - restriction on conversation mode 256
  - routing to back-end IMS 256
- introduction 221, 229
- message integrity 225
- message integrity, CICS 532
- message resynchronization 270
  - sessions 270, 279
  - summary 294
- message routing 246
  - FM headers 246
  - parameters 246, 247
  - through MSC links 255
- MSC coexistence 255
- network operation 266, 267
- node definition 258
  - COMM macro 260
  - example 258
  - macros used 258
  - NAME macro 261
  - SUBPOOL macro 261
  - summary 263
  - TERMINAL macro 262
- output editing option 235

ISC (Intersystem Communication) 221 *(continued)*

- output protocols 243, 244
- parallel sessions issuing IMS commands 236
- passing CICS data to IMS 229
- protocols 221
  - defined 265
  - restrictions on IMS-to-IMS sessions 257
  - summary 221, 222
- resynchronization procedure 271
  - cold start, to recover sessions 272
  - commands, recovery at failure 275
  - controlling backout of work unit 272
  - execution mode, recovery at failure 275
  - maintaining sequence numbers 271
  - output available at restart 273
  - recovering from in-bracket failure 273
  - restart process 275, 276
  - session failure, IMS failure 274
  - session failure, IMS still running 274
- routing 222
- routing examples 247, 255
- routing parameters 247
- SC protocols 267
  - binding sessions 268
  - cold start recovery 272
  - commands, recovery at failure 275
  - controlling pending work units 272
  - designing restart procedures 271
  - execution mode, recovery at failure 275
  - maintaining sequence numbers 271
  - output available at restart 273
  - recovering from in-bracket failure 273
  - resolving bind race 269
  - restart process 275, 276
  - session failure, IMS failure 274
  - session failure, IMS still running 274
  - session initiation 267
  - session states 277
  - session termination 279, 316
  - XRF, session initiation 268
- security 225
- sense codes 319
  - received 319
  - sent 319
- session control 221
- statically defining 257
- test mode 237
- transaction types supported, CICS session 499, 520
- versus CICS 520
- VTAM API 233
  - CICS 497
  - IMS 233
- VTAM facilities 231, 234
- XRF complex 225, 268

ISC-CICS installation options 507

ISC edit, non-MFS programs 245

## K

keywords 69

## L

- libraries, online change 99
- link priorities, setting 200
- link queuing time assessments 215
- Link-Receive Routing exit routine in MSC 189
- link security in MSC 205
- links in MSC 173
- list structure
  - definition 17
  - overflow, definition 17
  - primary, definition 17
- local systems and MSC 177
- lock mode 38
- LOCK state 77
- Log Analysis report 216
  - ID column for MSC entries 216
  - MSC transactions 216
- Log Merge utility 215
  - log output, control of 215
  - MSC logs 215
- Log Transaction Analysis utility, MSC statistics 215
- log write-ahead (LWA) 225
- logging, MSC 212
- logging off, definition 14
- logging on, definition 14
- logical destinations in MSC 181
- logical link path 191
  - commands for controlling 210
  - in MSC 191
- logical network design 32
- logical page advance function (NEXTLP), MFS 420
- logical page requests function (PAGE=nn), MFS 420
- logical terminal (LTERM)
  - chains 33
  - definition 12, 26
  - master terminal 35
  - MSC and 189
  - network design 32
  - queues 33
  - relationship to physical terminals 32
  - separating input and output devices 34
- logical unit
  - definition 8
  - programmable, definition 9
- logical unit of work 531
- logical unit status command 308
- logon descriptor
  - definition 118
  - generic, definition 118
  - group, definition 118
  - specific, definition 118
- logon mode
  - default logon mode table 233
  - session initiation 407
- LOGON MODE table 73
- lost messages in MSC 206
- LTERM
  - component definition 341
  - convention for naming 412
  - method for naming 411
  - naming convention for 240

- LTERM (*continued*)
  - subpools, users and components 240
- LUSTATUS command 308
  - CICS 500, 521
  - commit 310
  - conversation mode
    - requesting normal termination 287
    - sending errors 286
  - function abort 309
  - NO-OP 310
  - paging errors 311
  - protocol 308
  - queue empty 309
  - response mode errors 286
  - SLU P system 461
  - terminating test mode 237
- LWA (log write-ahead) 287

## M

- macro keywords 71
  - APPLID on COMM macro 70
  - COMM on BUFPOOLS macro 71
  - MODETBL on ETO logon descriptor 73
  - MODETBL on TERMINAL macro 73
  - PASSWD on COMM macro 70
  - RECANY on COMM macro 71
- maintenance 97
- making CICS ready 515
- making IMS ready 266, 431
- master-slave MSC relationship 173
- master terminal 74
  - device choice 35
  - logon requirements 74
  - MSC routing 189
  - reserving an LTERM 26
  - XRF complex 35
- master terminal operator (MTO) 74
- memory-to-memory link in MSC 173
- merging logs for MSC 215
- message 27
  - APPC/IMS 27
  - bypassing MFS editing 104
  - control characters 106
  - destination code 28
  - destination structure 28
  - DFSAPPC 27
  - editing and formatting 93
  - editing of output segments 105
  - editing performed by IMS 102
  - formatting and editing 99
  - in-flight
    - device LUs 75
    - program LUs 75
  - input segment 28
  - LU 6.2 messages 27
  - output segment editing 104
  - queuing, SERIAL=YES option 28
  - scheduling
    - definition 29
    - Fast Path 20
    - overview 27

- message 27 (*continued*)
  - segments, single or multiple 28
  - sensitivity to nongraphic message data 104
  - structure 28
  - switching 32
  - types 27
  - Z2 field 106
- message advance function (NEXTMSG) 420
- message advance protect function (NEXTMSGP) 420
- message buffering with a Fast Path-capable system 21
- message contention 270, 425
- Message Control/Error exit routine (DFSCMUX0) 202, 211
- message descriptor byte
  - input FM header 444, 445
  - output message 450, 452
- Message Format Service 93
- message headers 105
- message input descriptor (MID) 99
- message integrity 225
- message output descriptor (MOD) 98, 454
- message processing program (MPP)
  - definition 11
- message queue, definition 21
- message recovery
  - Finance Communication System 425
  - message resynchronization 294, 426
  - MFS 419
  - MSC 211
  - restriction 426
  - SLU P system 425
- message resynchronization 270
  - associated system definition options 271
  - CICS-IMS session 533
  - description 271, 533
  - design considerations 271, 433
  - direction and bracket indicators 441
  - effects on normal data transmission 426, 433
  - Fast Path 429
  - how and when initiated 270, 426
  - last inbound/outbound 426
  - OPTACK, results of specifying 424
  - options for message sequence numbers 433
  - performing 533
  - polarity of half-session pairs 271
  - purpose 270
  - requirements 270, 426
  - send/receive and bracket protocol 441
  - sequence numbers 272, 434
  - STSN
    - flow, primary-to-secondary 280
    - flow, secondary-to-primary 282
    - format 283, 435
    - use of 276
- message routing
  - examples 247, 255
  - in ISC
    - across an MSC link 255
    - use of routing parameters 247
  - types of 180
- message switch
  - ATTACH FM header 339
  - examples 248, 255
  - ISC 246, 247
- message switch, ISC 236
- message switching, DFSAPPC 390
- MFS (Message Format Service) 93
  - administration 96
  - advantages 97
  - application programs, accessing with ISC 245
  - Bid options 421
  - bypassing MFS 104
  - bypassing with ISC application programs 245
  - components, overview 99
  - control functions 420
    - Finance Communication System 420
    - NEXTLP 420
    - NEXTMSG 420
    - NEXTMSGP 420
    - NEXTPP 420
    - PAGE=nn 420
    - SLU P 420
  - control requests, response requirements 459
  - data bytes output message 452
  - default format 356
  - defining 418
  - delete characters 105
  - DPM 449
  - DPM option 104
  - edit of ISC messages 235
  - editing 104
  - escape characters, not supported in ISC 329
  - facilities available 418
  - FM headers 329
    - activating input formatting 329
    - activating output formatting 330
    - ATTACH 333
    - data descriptor 334
    - DPM messages 330
    - editing 329
    - facilities available 329
    - MOD name 330
    - QGET 336
    - QGETN 336
    - QMODEL 335, 338, 530
    - QPURGE 337
    - QSTATUS 337
    - QXFR 337
    - RAP (reset attached process), MFS 338
    - SCHEDULER 333
    - specifying MID name 330
    - types 329, 338
  - format errors not detected 329
  - formatting 432
    - activating input 432
    - activating output 447
  - input formatting 329
  - input messages 93
  - input segments 105
  - introduction 93
  - invalid paging request 422, 425

- MFS (Message Format Service) 93 *(continued)*
  - Language utility 94, 99
  - libraries, online change 99
  - message editor 99
  - message recovery 419
  - MID/MOD chaining 419
  - MSC 197
  - online error detection 331
  - online performance 98
  - output formatting 419, 454
    - MOD name 454
    - typical application program procedure 454
  - output formatting, ISC 330
  - output messages
    - editing segments 104
    - how MFS defines 93
  - overview 13, 93
  - page delete function, not supported in ISC 331
  - paging, CICS 499, 520
  - pool manager 99
  - Service utility 99
  - SLU P 419
    - benefits 419
    - station-by-station, availability 419
  - sync point, ISC messages 290
  - terminal keyboard lock and unlock 104
- MFS bypass option effect 94
- MFS device characteristics table
  - entries 23
  - how used 130
  - use with non-SNA 3270 devices 131
- MFS device descriptor, definition 119
- MFS Distributed Presentation Management (DPM) 235
- MFS Service utility 95
- MFSTEST procedure 99
- MID (message input descriptor) 99
- MID/MOD chaining 419
- mirror transaction, CICS 504
- MOD (message output descriptor)
  - name specification 330, 454
  - purpose 98
- modes, terminal
  - conversation 37
  - ETO and exclusive mode 38
  - exclusive 38
  - lock 38
  - LU 6.2 37
  - response 37
  - SNA QUIESCE 39
  - test mode 38
- MODETBL= keyword
  - overriding the defaults 233
  - specifying a default LOGON MODE identifier 73
  - use during ISC logon 233
- modified application program
  - LU 6.2 descriptor 380
  - remote execution, MSC 371
- monitoring and tuning, MSC 212
- MPP 11
- MSASSIGN command
  - MSC considerations 207
- MSASSIGN command *(continued)*
  - MSC linking 175
  - usage considerations 208
- MSC (Multiple Systems Coupling) 382
  - /DEQUEUE command for MSC 211
  - DEQUEUE command for MSC 211
  - administration
    - APPC 193
    - multiple systems 193
  - asynchronous LU 6.2 transactions 195
  - capabilities 13
  - coexistence with ISC 255
  - comparison to ISC 222
  - concepts 173
  - data flow 179
  - data partitioning 176
  - defining priorities 200
  - definition 173
  - definition of transaction codes 197
  - dequeuing messages, responses, and transactions 211
  - design considerations 193
  - destination system 182
  - directed routing, program-to-program switch 191
  - exit routines 201
  - functions compared to ISC 223
  - input system 182
  - intermediate system 182
  - introduction 173
  - ISC facility 222
  - logical destinations 181
  - logical links 175
  - logical links, SDLC link 175, 194
  - message routing 180
  - minimizing resource consumption 193
  - monitoring and tuning 212
  - MSC conversation failure 197
  - non-VTAM links buffer size 199
  - operating procedures 206
  - performance 212
  - physical links 173
  - Queuing Summary Report 213
  - recoverable versus irrecoverable transactions 382
  - recovering transactions in APPC 382
  - recovery considerations 210
  - routing path 181
  - security considerations 205
  - support for APPC/IMS 381
  - system definition 197
  - Traffic Report 214
  - utility for verifying names 203
  - VTAM links buffer size 199
- MSC descriptor, definition 118
- MSC LU 6.2 application transactions buffer size 199
- MSC Program Routing exit routine 189
- MSLINK macro, logical links 199
- MSNAME macro
  - logical link paths 192
  - logical path name 200
  - MSC 190
  - reports 213

MSNAME macro (*continued*)  
 for MSC 213  
 MSC Queuing Summary 215  
 MSC Summaries 214  
 MSC Traffic 213  
 SYSID keyword 200  
 MSPLINK macro, MSC partner definition 199  
 MSVERIFY command  
 dynamic validation 204  
 error responses 205  
 SYSIDs 204  
 MTM link in MSC 173  
 MTO (master terminal operator) 74  
 /OPNDST command 73  
 /RSTART LINK command 206  
 RSTART LINK command 206  
 assignment of logical link 173  
 ISC errors 313  
 MSC and master terminal operators 207  
 notification of session rejection 279  
 restarting the terminal 103  
 termination of transmission 206  
 MULT1 parameter 244, 300  
 MULT2 parameter 244, 300  
 multichain input message switches restriction 341  
 multiple signons 72  
 description 152  
 naming conventions for 152  
 Sysplex environment, in 152  
 multiple systems 193  
 Multiple Systems Verification utility (DFSUMSV0) 191  
 Multiple Systems Verification utility, using 203  
 multisystem communication  
 initialization 206  
 termination 206

## N

NAME macro 261  
 defining an ISC session 261  
 for remote logical terminals 192  
 MSNAME macro 203  
 used to relate components 241  
 naming conventions 96  
 NCP (Network Control Program) 10  
 negotiable bind 269  
 network 7  
 administration 23  
 APPC/IMS, operating with 14  
 communications 7  
 definition 70, 203  
 design considerations 23  
 documenting requirements 24  
 logical terminal network design 32  
 multiple systems, effect of 203  
 nonswitched communications network 35  
 operating  
 establishing communication 14  
 Finance Communication System 75, 400, 431, 441  
 IMS-CICS 515, 516

network 7 (*continued*)  
 operating (*continued*)  
 ISC 266, 267  
 SLU P 400  
 SLU P system 75, 431  
 operations, preparing for 69  
 restarting after remote takeover 78  
 shutting down 266, 440  
 starting  
 Finance Communication System 75, 431  
 SLU P system 75, 431  
 terminating  
 Finance Communication System 439, 441  
 SLU P system 441  
 VTAM and NCP parameters 70  
 network, communications 7  
 network-qualified LU name 389  
 networking 193  
 NO-OP indicated on LUSTATUS command 310  
 node definition, ISC 257, 264  
 node name 25  
 node user descriptor 143  
 nongraphic message data 104  
 nonnegotiable bind 269  
 nonswitched communications network 35  
 NOSCAN option 448  
 NRESTART command 108  
 null output message  
 purpose 447  
 when not sent 419  
 when sent 453

## O

online change  
 DISPLAY MODIFY command 77  
 libraries 99  
 Online Change utility 99  
 online performance 98  
 operator logical paging  
 paged messages 335, 337  
 paging errors 332  
 QXFR FM header 354  
 SLU P, MFS option 420  
 sync points 290  
 VTAM indicators 301, 303  
 when in effect 295  
 OPNDST command  
 logging on 150  
 relation to MODETBL keyword 73  
 results of using 159  
 OPTACK option  
 Fast Path 427  
 Finance Communication System 458  
 message resynchronization, effects 424  
 SLU P 425, 455  
 options, CICS  
 system generation 508  
 table preparation 508  
 OPTIONS= DPAGE or PPAGE  
 MFS DPM 330, 449



OPTIONS= DPAGE or PPAGE (continued)  
     output message 354  
 orderly session termination  
     CICS 516  
     ISC 279  
     VTAM 439  
 output  
     available at ISC restart 273  
     controlling 107, 110  
     MFS DPM 449  
 OUTPUT= parameter, not used in ISC 261  
 output algorithms  
     RU chain 342  
     VLVB 342  
 output bracketing 453  
 output component  
     defining 244  
     relationship to input component 239  
     selection  
         identification number 241, 412  
         modifying 240, 412  
         system messages 326, 412  
 output component ID byte, output message 451  
 output component protection, extended 422  
 output devices, control characters by type 106  
 output editing option, SLU P 423  
 output errors, MFS online detection 331  
 output function management headers, ISC 525  
 output message  
     between brackets  
         description 417  
         design considerations 418  
         figure 417  
         how handled 417  
         ISC 276  
         message switching 458  
         sequence of operations with BID specified 417  
     description 447  
     Fast Path  
         Finance systems 427  
         SLU P systems 428  
     Finance Communication System  
         multiple transmission 447  
         read-flags field (SMSCRF/SMSCRE) 447  
         read type field (SMSCRT) 447  
     formatting, activating MFS 330, 454  
     ISC bracketing 299  
     MFS DPM 330  
     null  
         purpose 447  
         when sent 453  
     output bracketing 453  
     output FM header  
         Finance format 450  
         ID byte 451  
         message descriptor byte (Finance) 450  
         message descriptor byte (SLU P) 452  
         MFS data bytes (Finance) 451  
         MFS data bytes (SLU P) 452  
         response requests 455  
         SLU P format 451

output message (continued)  
     segmenting 447  
     SLU P system  
         multiple transmission 447  
         read-flags field (SMSCRF/SMSCRE) 447  
         read type field (SMSCRT) 447  
     sync point requested, ISC 289  
     temporarily stopping 438  
     types 447  
     when committed 289  
 output protocols, determining 243  
 output response requested by message type 456  
 overload avoidance, MSC 194

## P

page advance function (NEXTTP), MFS 420  
 page delete function, MFS 331  
 page protection state 39  
 paging errors  
     ISC 311  
     online detection, MFS 332  
 parallel sessions  
     bind requirements 269  
     ISC, defining IMS-to-IMS sessions 255  
     users permitted 240  
 partner systems in MSC linking 175  
 PCB (program communication block)  
     alternate PCB 241  
     for an LTERM not in I/O PCB 412  
     I/O PCB 241  
 physical link  
     CTC type 199  
 physical link failure 208  
 physical links, MSC 173  
 physical terminal  
     defining 125  
 physical terminals  
     device class control 106  
     separating input and output devices 34  
 pool manager 99  
     MFS description 99  
     MFSTEST 99  
 primary error recovery procedure 405  
 primary resource name (PRN) 345  
 printed output  
     /ASSIGN command 107  
     ASSIGN command 107  
     3270 printer components 108  
     3270R and ETO 108  
     candidate printers 108  
     controlling 107, 109  
     operational considerations 109  
     printer component 107  
     shared printers 110  
     spooled output control 108  
     VTAMLIST definitions 108  
 printers  
     candidate 108  
     sharing 110  
 PRN (primary resource name)  
     ATTACH FM header 345

PRN (primary resource name) *(continued)*  
 message routing, ISC 246  
 SCHEDULER FM header 345  
 processor utilization, MSC 194  
 program communication block (PCB) 241, 412  
 program-to-program switch 196  
 conversational transactions 191  
 destination name 185  
 nonconversational transactions 191  
 remote destination verification 196  
 programmed symbols (PS) 467  
 protected resource, definition 363  
 protection  
 display screen 421  
 extended output component 422  
 protocol  
 output 243  
 restrictions on IMS-to-IMS sessions 257  
 PS (programmed symbols)  
 administering 467  
 definition 467  
 designing applications 469  
 determining if loaded 467  
 example of loading 467  
 how to load 467, 469  
 solving load problems 468  
 PSTOP LINK command 206  
 PSTOP state 77  
 PURGE  
 error recovery procedure 305  
 begins 305  
 DFC protocol 306  
 DFC support layer 305  
 ends 305  
 multichain 305  
 single 305  
 spanning chains 306  
 state after selective receiver purge 307  
 state after sender purge 306  
 sender ERP 316  
 PURGE state 77

## Q

QEC command 438, 439  
 QERROR state 38  
 QGET FM header  
 description 336  
 format 349  
 QGETN FM header  
 description 336  
 format 350  
 QLOCK state 39  
 QMODEL FM headers  
 CICS 530  
 formats 349, 355  
 QGET 336  
 QGETN 336  
 QPURGE 337  
 QSTATUS 337  
 QXFR 337  
 reply or output 337

QMODEL FM headers *(continued)*  
 request or input 336  
 supported by IMS 335  
 QPURGE FM header  
 format 351  
 received by IMS 337  
 QSTATUS FM header  
 format 352  
 sent by IMS 337  
 QSTOP state 77  
 qualifying an LU name 389  
 queue empty indication 309  
 Queue Manager  
 in XRF environment 47  
 planning for 47  
 queue rotation 301  
 queued subsystem 518  
 queues  
 balancing group 21  
 logical terminals 33  
 shared 15  
 quiesce-at-end-of-chain command, VTAM 438  
 QXFR FM header  
 as output 337  
 format 353, 355

## R

race, BIND 269  
 RACF (Resource Access Control Facility) 12  
 ETO security 12, 134  
 multiple system security 205  
 transaction security 394  
 unauthorized terminal use 40  
 RAP FM header  
 description 324  
 example 325  
 format 355  
 MFS 338  
 RDPN (Return Destination Process Name)  
 ATTACH FM header 346  
 message routing, ISC 246  
 SCHEDULER FM header 346  
 read flags field, output messages 447  
 read type field, output messages 447  
 ready-to-receive (RTR) command 312  
 reassigning links in MSC 207  
 receive-any buffers 71  
 recoverable input, acknowledging 424  
 recoverable-inquiry transactions, response requirements  
 IMS 455, 457  
 ISC 289, 292  
 recoverable message sent, sequence numbers 434  
 recoverable messages, ISC sync points 290  
 recoverable versus irrecoverable messages 454, 458  
 recoverable versus irrecoverable transactions 382  
 recovering transactions in APPC 382  
 recovery considerations for MSC 210  
 recovery environment  
 distributed resources, definition 364  
 local resources, definition 364

- recovery procedures 66
- RELQ command 438
- remote control of IMS 256
- remote destination name 185
- remote destination verification
  - MSC conversations 196
  - system integrity 191
- remote LTERMs 187
- remote systems (MSC) 177
- remote takeover
  - determining message loss after 78
  - restarting the network after 78
  - VTAM definition for Transport Manager Subsystem 79
- remote terminal operator (RTO) 74
- reports 213
  - IMS Monitor 213
  - MSC 213
- reset attached process FM header 324
- resource
  - commands to control MSC 208
  - MSC considerations 193
- Resource Access Control Facility (RACF) 205
- resource manager, definition 363
- Resource Recovery Services/MVS (RRS/MVS) 363
- response mode
  - dynamic establishment 240
  - errors
    - detection 285
    - ISC 285, 287
  - ISC
    - input 523
    - output 523
  - terminal
    - definition 413
    - design considerations 415, 499
    - effects of specifying transaction-dependent 413
    - figure 413
    - introduction 37
    - methods of termination 415
    - restrictions 415
    - sizes of message queue data sets 447
    - sizes of workstation output buffers 447
    - station-by-station 413
    - types 413
    - when activated 413
- response requests, output message 456
- response requirements
  - IMS commands and indicators 439, 454
  - inquiry transactions
    - irrecoverable 287, 457
    - recoverable 287, 457
  - irrecoverable-inquiry transactions 77
  - ISC messages 289
  - LU 6.2 application program 77
  - MFS control requests 459
  - output ISC messages 289
  - verifying IMS receipt 289, 458
  - VTAM indicators and commands 232, 439
- response time 98, 216
- restart
  - coding CICS applications 537
  - output available, ISC 273
- restart and recovery 531
- restart transaction, CICS 506
- resynchronization 424
- resynchronization, message 270
- Return Destination Process Name 346
- Return Primary Resource Name 346
- routing code, definition 21
- routing exit routines
  - MSC conversations 196
  - MSC routing 188
- routing input messages 190
- routing messages
  - destination name 185
  - ISC examples 247, 255
  - parameters 247
  - SYSIDs 185
- routing path (MSC) 181
- RPRN (Return Primary Resource Name)
  - ATTACH FM header 346
  - message routing, ISC 247
  - SCHEDULER FM header 346
- RQ\* messages, LUSTATUS command 308
- RQD\* 231, 289, 294
- RQE\* 231, 289, 294
- RQR command, SLU P 406
- RRS/MVS (Resource Recovery Services/MVS) 363
  - description 363
  - resource recovery with 363
- RSHUT command 312
- RSR (Remote Site Recovery)
  - protected conversations and 366
  - restarting the network 78
  - shared queues, planning for 56
- RSTART LINK command 206, 208
- RTO (remote terminal operator) 73
- RTR (ready-to-receive) command
  - IMS functions 462
  - protocol 312
  - resetting component status to unprotected 422
- RU (request unit) chaining in ISC 304

## S

- SBI (stop bracket initiation) command
  - CICS 516
  - IMS-CICS session 516
  - session shutdown 320
  - use in data flow control 221
- SBI command 320
- SC (session control) protocols 267
  - BIND parameters 276
  - binding sessions 268
    - negotiable versus nonnegotiable BIND 269
    - parallel session 269
    - resolving a race 269
    - single session 269
    - synchronizing sessions 270
  - message resynchronization 270
    - commands used 270

SC (session control) protocols 267 *(continued)*  
     designing procedures 271, 276  
     when required 270  
 session initiation 278  
     completing 278  
     ISC 267  
 session states 277  
 session termination 279  
     abnormal 280  
     normal 279  
 STSN flow 280  
     primary-to-secondary half session 280  
     secondary-to-primary half session 282  
 STSN format 283  
 XRF complex, establishing an ISC session 268  
 XRF complex, establishing connection 268  
 SCA (system control area), ISC support 245  
 SCHEDULER FM header  
     ATTACH 325  
     chained message support 326  
     example 489  
     format 355  
     IMS-CICS session 528  
     introduction 325, 326  
     MFS 333  
     parameter description 343, 347  
     request for asynchronous execution 238  
 scheduling  
     algorithm 29  
     Fast Path messages 20  
 scratch pad area (SPA) 31  
 screen protection, Finance Communication System  
     BID option 421  
     NOBID option 421  
 screens  
     protection 39  
     unprotected screen option 104  
 SCS (SNA character string) controls  
     format controls 487  
     function code assignments 488  
 SDF II, definition 99  
 SDT (start data traffic) command 278  
     completing session initiation 278  
     IMS 437  
 secondary logical unit  
     design considerations 234  
     first speaker in ISC 234  
 secondary logical unit type P 399, 412  
 security considerations  
     CICS 522  
     ISC 225  
     MSC 205  
     SMU in MSC 205  
 Security Maintenance Utility, MSC 205  
 security options 40  
     APPC/IMS, RACF 42  
     APPC/IMS, SAF 42  
     command authorization 41  
         DFSCCMD0 Command Authorization exit  
         routine 41  
         LTERM name 41  
     security options 40 *(continued)*  
         command authorization 41 *(continued)*  
             user ID 41  
         ETO 42  
             user ID 42  
             user LTERM name 42  
         levels of security 40  
             access to resources 40  
             access to terminals 40  
         password 42  
             LTERM name 42  
             user ID 42  
         RACF 40  
         security profile 41  
             DFSCTRN0 Transaction Authorization exit  
             routine 41  
             LTERM name 41  
             user ID 41  
         signon verification 40  
         SMU 40  
         transaction authorization 40  
             RACF 40  
             SMU 40  
         transaction command 41  
         transaction security 394  
             RACF 394  
             UACC (NONE) 394  
     selective receiver ERP  
         description 312  
         sense codes 313  
     send and receive protocol  
         bracketing  
             input 446  
             output 453  
         IMS 441  
     SEND INVITE EXEC command, CICS 501  
     SEND LAST EXEC command, CICS 502  
     send/receive and bracket protocol 441  
     SEND/RECEIVE EXEC command, CICS 501  
     sender ERP, sense codes 317  
     sending IMS commands from CICS 521  
     sense code 319  
         definition 319  
         error 319  
         received during ISC 319  
         selective receiver ERP 312, 316  
         sender ERP 317, 318  
         sent during ISC 319  
     separating input and output devices 34  
     sequence numbers  
         description 275  
         maintaining 271  
         management 283, 434  
         storage 283  
         use 276, 434, 438  
     session 9  
         binding 268  
         definition 9  
         establishing 14  
         ISC, establishing 267  
         parallel 269

- session 9 (*continued*)
  - remote control 256
  - requirements to bind ISC 268
  - single 269
- session initiation 75
  - bind parameters
    - Finance Communication System 433
    - SLU P system 433
  - IMS-CICS 515
  - ISC 267
  - possible session states 277
  - requested by
    - CICS 515
    - master terminal operator, IMS 267, 432
    - network operator, MVS VTAM 267, 432
    - system definition 267, 432
    - workstation 432, 446
  - step-by-step explanation 432
  - transmission sequence 432
  - ways to request 76
- session local flag, ATTACH FM header 347
- session parameters
  - establishing connection 267, 407
  - use 267, 407
- session termination 279
  - abnormal 280, 516
  - BID option specified, effects 461
  - CICS 516
  - conditional versus unconditional 234
  - definition 279, 439
  - immediate
    - definition 439, 516
    - master terminal operator, IMS 279, 439
    - network operator, MVS VTAM 279, 439
    - station 279
    - workstation 439
  - normal 279, 516
  - orderly
    - definition 440
    - initiated 320, 440
    - requested by CICS 516
    - sequence 321, 440
  - summary (figure) 439
  - symmetrical (SBI/BIS) in ISC 320
- set-and-test-sequence-numbers (STSN) command 280
- shared queues 15
  - benefits of using 16
  - definition 15
  - environment 15
    - components of, illustration 17
    - operating in, overview 15
    - required components of 17
  - MVS system log, role of 17
  - planning for 43
- sharing printers
  - between systems 110
  - between users 110
- shutting down an IMS network 266, 441
- SIGNAL command
  - protocol 320
  - VTAM 463
- signing off, definition 14
- signing on, definition 14
- signon and static terminals 72
- single session, ISC bind requirements (IMS-to-IMS) 255
- SINGLE1 parameter 244, 300
- SINGLE2 parameter 244, 300
- slave-master MSC relationship 173
- SLU P 399
  - application program
    - converting from Finance 404
    - functions available 402
    - MFS considerations 402
    - XRF considerations 403
  - bracket and send/receive protocols 408, 441
  - IMS facilities 411
  - input component definition 412
  - MFS DPM option 449
  - network
    - major parts 400
    - operating 400
  - sample configuration 400
  - SCAN/NOSCAN 404
  - session parameters 471
  - terminals supported 400
  - VTAM
    - commands and indicators 405
    - facilities 404
    - workstations 401
    - XRF complex, establishing connections 408
- small buffer devices 106
- SNA character string (SCS) controls 487
- SNA commands 232
- SNA QUIESCE 39
- SPA (scratch pad area) 31
  - characteristics 31
  - MSC conversations 195
  - size 191, 195
  - transaction code field 31, 37
- splitting databases, MSC 194
- SPOOL command
  - output control 108
- spooled output control 108
- standard application programs and MSC 370
- START command
  - making IMS ready 75
  - starting a SLU P network 431
- start data traffic (SDT) command 278
- start lists, VTAM 74
- START/RETRIEVE EXEC commands, CICS 505
- starting a SLU P network 75
- starting an IMS network 75
  - making IMS ready
    - /START command 266, 431
    - results 266, 431
  - prerequisites 75, 431
  - results 75
- starting CQS 86
- starting MSC 206
- starting workstations 75
- station, user 400

- station, user 400 *(continued)*
  - definition of as logical unit 400
  - device selection 400
  - terminology 401
- Statistical Analysis Utility and MSC 212
- stop bracket initiation command 320
- STOP command 38
- STOP state 38, 78
- stopped transactions in MSC 211
- stopping
  - MSC 206
  - workstations 279
- structure
  - creation and deletion 119
  - list
    - definition 17
    - overflow, definition 17
    - primary, definition 17
    - pair, definition 17
- structure recovery data set (SRDS)
  - overview 17
- STSN command 280
  - action code 438
  - action code format 283, 285, 435
  - action required 276, 435
  - controller sequence number, verification 435
  - Finance Communication Systems 435, 438
  - flow 280
  - message resynchronization 435, 437
  - response 276, 437
    - requirements 437
    - response requirements 438
    - summary 437
    - summary (figure) 276
  - sync points 275
    - VTAM sequence number, verification 435
- STSN function, requirements for ETO 167
- STSN support for ETO devices 168
- SUBPOOL macro 240, 261
- subpools users
  - dynamic allocation 240
  - VTAM 240
- subsystem
  - direct-control 518
  - external, definition 12
  - queued 518
- suspending output from IMS 438
- sync point
  - input messages, ISC 288
  - ISC
    - CICS 522
    - input 292
    - output 293, 294
    - requirements 289
- synchronization, ISC half sessions 287
- synchronous processing
  - ATTACH FM header, ISC 323
  - CICS 501
  - definition 238
- SYS1.VTAMLST 70
  - ATCCONxx member 74
- SYS1.VTAMLST 70 *(continued)*
  - ATCSTRyy member 74
  - defining IMS as an application node 70
  - VTAM nodes 71
- SYSID keyword, MSC 192
- SYSIDs 184
- SYSMSG FM header
  - ATTACH FM header 326
  - format 356
  - sending 326
  - types 327
- Sysplex environment 15
  - definition 15
  - shared queues in 15
- SYSSTAT FM header, format 356
- system
  - definition
    - IMS ISC sample 257, 263
    - messages, length 327
  - system control area (SCA) 245
  - system definition 197
    - macros 197, 198
    - MSC
      - exit routines 201
      - general considerations 197
      - implications 203
      - local 198
      - macros 198
      - partner 198
      - setting link priorities 200
      - verifying 203
  - system identification for logical link paths 192
  - system identifier (MSC) 184
  - system log 212, 215
  - system log, MVS
    - shared queues and 17

## T

- terminal
  - physical, defining 125
- TERMINAL macro 262
  - defining an ISC session 262
  - MODETBL= keyword 73
- terminal modes 36
  - conversation mode 37
  - exclusive 38
  - lock mode 38
  - response mode 37
  - SNA QUIESCE 39
- Terminal Routing exit routine 188
- terminal states 38
- terminals
  - attached through VTAM 24
    - Class 1 with XRF 26
    - Class 2 with XRF 26
  - COMPINOP state 39
  - component protection state 39
  - conversation mode 37
  - definition 8
  - device class control 106

- terminals (*continued*)
  - documenting requirements 24
  - ETO and exclusive mode 38
  - exclusive mode 38
  - INOP state 39
  - lock mode 38
  - logical
    - chains 33
    - master terminal 35
    - network design 32
    - queues 33
    - relationship to physical terminals 32
  - LU 6.2 terminals and Fast Path 62
  - modes and states 36, 39
  - nonswitched communications network 35
  - page protection state 39
  - profiles 69, 71
  - QERROR state 38
  - QLOCK state 39
  - response mode 37
  - screen protection state 39
  - separating input and output devices 34
  - SNA QUIESCE 39
  - STOP state 38
  - support for, IMS 24
  - test mode 38
- terminating communications, MSC conversations 197
- terminating ISC Extension conversations 287
- termination, session 279
- test mode 38, 237
- traffic between two IMS systems 240
- TRANSACT macro
  - EDIT=ULC 105
  - PRTY= keyword 200
  - translation to uppercase 101
- transaction 76
  - abends 536
  - code, definition 21
  - codes, unique 28
  - MSC statistics 215
  - multiple systems 173
  - states 77
- Transaction Analysis utility 212
- transaction code (remote destination) 185
- transaction sync point relationships 288, 523
- transaction types
  - commands 458
  - definitions 76, 456
  - inquiry
    - definition 455
    - recoverable or irrecoverable 457, 458
  - ISC session, during 236
  - message switches
    - IMS 458
    - ISC 341
    - ISC examples 248, 252
    - supported by ISC, list 224
    - test mode, in ISC 237
    - update 456
- transparency option 103

- tuning
  - buffers 213
  - MSC environment 193
  - MSC tuning and monitoring 212
- two-phase commit process, definition 363
- TYPE macro 25

## U

- Unaccessed ETO User Control Blocks 159
- UNBIND command, stopping session initiation 278
- unconditional bracket termination, IMS error handling 459, 461
- unit of recovery
  - definition 364
  - in-doubt, definition 364
  - in-flight, definition 364
- UNTYPE keyword on TYPE macro 65
- unprotected screen option 104
- unscheduled reassignments, MSC 207
- USER 236
- user descriptor 144
- user descriptor, definition 118
- user workstation
  - API (application program interface) 497
  - CICS asynchronous 497, 518
  - CICS synchronous 497, 518
  - use 233
  - bracket protocol 296
  - input messages, ISC 296
  - output messages, ISC 299
- USERVAR 268
- USSTAB option defined in VTAM 73
- USTOPPED state 78

## V

- verification
  - IMS terminal support 24
  - remote destinations 191
  - transaction definitions across systems 203
- vertical partitioning 176
- VLVB records 342
- VTAM (Virtual Telecommunications Access Method)
  - ACBNAME parameter 199
  - APPL statement 199
  - attached terminals 24
  - BB indicator 243, 308
  - BID command 295, 416
  - binary synchronous communications (BSC) 24
  - BIS command 320
    - LU 6.1 half sessions 320
    - use with CICS 516
  - bracket protocol 408
    - CD (change direction) 408
    - Finance Communication System 408
    - SLU P 408
  - bracket protocols 238
  - buffering 79
  - CANCEL command 304, 462
  - CD indicator 243, 406
  - CHASE command 305

VTAM (Virtual Telecommunications Access Method)  
*(continued)*  
 COMM macro 199  
 commands and indicators 232, 405  
 definition for Transport Manager Subsystem 79  
 devices with MFS support 24  
 EB indicator 243, 309  
 establishing sessions 407  
   Finance Communication System 407  
   SLU P 407  
 facilities 232  
   commands and indicators 232, 405  
   data transmission 232  
   Finance Communication System 404  
   IMS 405  
   ISC 232  
   SLU P 404  
   used by ISC 499, 518  
 generation 70  
   IMS as host subsystem 70  
   LOGON MODE identifiers 73  
   NCP buffer pool values 72  
   storage requirements 71  
   VTAM buffer pool values 71  
   VTAM configurations 74  
   VTAM nodes 71  
 half-duplex protocol, ISC 296  
 IMS, relationship to 10  
 macros 234  
   SEND 234  
   TERMSESS 234  
 missing interrupt handler (MIH) 79  
 MSC linking 173  
 node as chosen name 25  
 output buffers 71  
 pacing 79  
 parallel sessions 199  
 RQR command 406, 463  
 RTR command 462  
   Fast Path 462  
   summary 312, 462  
 SBI command 320  
   LU 6.1 half sessions 320  
   use with CICS 516  
 SDLC using VTAM 199  
 SDT command 278  
 SESSION parameter 199  
 SIGNAL command 320, 463  
 synchronous data link control (SDLC) 24  
 Transport Manager Subsystem, definition for 79  
 UNBIND command 278  
 VTAM 210

## W

work unit  
 backed out 270, 531  
 CICS 531  
 committed 270, 531  
 definition 270  
 example 271

work unit *(continued)*  
 pending, unilateral decisions 272  
 status at session initiation 277  
 workload distribution, MSC 194  
 workstation, user  
   definition as logical unit 240  
   logical unit definition 10  
   terminology 240  
 workstations  
   Finance Communication System 401  
   SLU P 401  
   terminology 400

## X

XRF (extended recovery facility)  
 SLU P application program 403  
 XRF (Extended Recovery Facility) 26  
   Class 1 terminals 67  
   Class 2 terminals 66  
   Class 3 terminals 66  
   class of service 67  
   definition 64  
   establishing communication 268  
     Finance Communication System 408  
     ISC 268  
     SLU P 408  
     system takeover considerations 408  
   local queue manager data sets 47  
   master terminals 35  
   takeover considerations 408  
     Finance Communication System 408  
     SLU P 408  
   terminal support 26, 67  
   terminals in 65  
   USERVAR 268

## Z

Z2 field in message prefix 106



---

# Readers' Comments — We'd Like to Hear from You

IMS/ESA  
Administration Guide:  
Transaction Manager

Publication No. SC26-8731-04

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<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"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<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

International Business Machines Corporation  
Department BWE/H3  
P.O. Box 49023  
San Jose, CA 95161-9945



Fold and Tape

Please do not staple

Fold and Tape





Program Number: 5655-158



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SC26-8731-04



Spine information:



IMS/ESA

IMS/ESA V6 Admin Guide: TM

Version 6