TXSeries for Multiplatforms

CICS Intercommunication Guide

Version 6.0

SC34-6644-00

TXSeries for Multiplatforms

CICS Intercommunication Guide

Version 6.0

SC34-6644-00

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 357.

First Edition (November 2005)

This edition replaces SC09-4462-04.

Order publications through your IBM representative or through the IBM branch office serving your locality.

Contents

Figures ix

Tables xi

About this book xiii
Who should read this book xiii
Document organization xiii
Conventions used in this book xiv
How to send your comments xv

Part 1. Intercommunication planning 1

Chapter 1. Introduction to CICS intercommunication 3
Network protocols 3
Overview of the intercommunication facilities . . . 3
IBM CICS Universal Client products 4

Chapter 2. Intercommunication planning and system design 5
Designing your network configuration. 5
Communicating across TCP/IP connections . . . 5
Communicating across SNA connections 8
Mixing the communications methods 11
Summary of communication methods 13
Additional considerations 14
Ensuring that the system is still secure 14
Identifying the remote system 14
Identifying the user that initiated the request . . 15
Ensuring data integrity with synchronization support. 16
Converting between EBCDIC and ASCII data . . . 19
Performance issues for intercommunication. . . . 19
Choosing which intercommunication facility to use 20
A checklist of application requirements 21
Operational issues for intercommunication 22
Bidirectional input and display on AIX 22

Part 2. Configuring for intercommunication 23

Chapter 3. Configuring CICS for TCP/IP 25
Naming your region for a TCP/IP intercommunication environment 25
Configuring CICS for CICS family TCP/IP support 26
Configuring LD entries for CICS family TCP/IP 27
Adding a service name to the TCP/IP configuration 30
Increasing the size of the mbuf pool (AIX only) 30
Tuning TCP/IP on the Windows platform . . . 33
Configuring CD entries for CICS family TCP/IP 34
Using SMIT to configure CD entries (AIX only) 39
Configuring an IBM CICS Client to use CICS family TCP/IP. 42
Timeouts supported on TCP/IP connections between a TXSeries for Multiplatforms region and a Universal Client V3.1 or higher 43
CICS family TCP/IP Configuration examples . . . 43
CICS OS/2 configuration 44
Configuration for region cicsopen 45
Configuration for region cics 46
Configuring CICS for CICS PPC TCP/IP support. . 47
Configuring CD entries for CICS PPC TCP/IP. . 47
Summary of CICS attributes for TCP/IP resource definitions. 54
Configuring for autoinstallation of CD entries . . . 55
Configuring the default CD entry (CICS on Open Systems) 56
Configuring the default CD entry (CICS for Windows) 57
Configuring region database table sizes 58
What the supplied version of DFHCCINX does 58
The parameters passed to DFHCCINX 59
Writing your own version of DFHCCINX . . . 62
Installing your version of DFHCCINX into the region 65

Chapter 4. Configuring CICS for SNA 67
Naming CICS regions in an SNA intercommunication environment 67
Configuring CICS for local SNA support 70
Configuring CICS for local SNA from the command line 75
Configuring CICS for local SNA support by using the IBM TXSeries Administration Tool (CICS on Windows platforms only) 77
Configuring CICS for local SNA support by using SMIT (CICS for AIX only) 82
SNA Receive Timeout feature (For CICS on AIX only) 88
Configuring CICS for PPC Gateway server SNA support. 89
Configuring CICS for PPC Gateway server SNA support from the command line 95
Configuring CICS for PPC Gateway server SNA support by using the IBM TXSeries Administration Tool (CICS on Windows platforms only) 98
Configuring CICS for PPC Gateway server SNA support by using SMIT (CICS for AIX only) . . 103

Chapter 5. Configuring resources for intercommunication 109
Configuring transactions for intersystem communication. 109

Transactions over an SNA connection (CICS on Open Systems only) 109
Back-end DTP transactions over TCP/IP and an SNA connection 109
Configuring for the indoubt condition 109
Default modenames 110
SNA tuning 111
Configuring intrapartition TDQs for intercommunication 111
Terminals as principal facilities 112
Remote terminals as principal facilities 112
Remote systems as principal facilities 112
Configuring Program Definitions (PD) for DPL . . 112
Example of a PD for DPL 113
Defining remote resources for function shipping 113
Defining remote files 113
Defining remote transient data queues 114
Defining remote temporary storage queues . . 115
Defining resources for transaction routing 116
Shipping terminal definitions 117
Fully qualified terminal identifiers 117
Defining terminals to the application-owning region 118
Using the alias for a terminal in the application-owning region 118
Defining remote transactions for transaction routing 119
Defining remote transactions for asynchronous processing 120

Chapter 6. Configuring intersystem security 123

Overview of intersystem security 123
Implementing intersystem security 123
Summary of CICS intersystem security 124
Identifying the remote system 124
Authenticating systems across CICS family TCP/IP connections 124
Authenticating systems across CICS PPC TCP/IP connections 124
Authenticating systems across SNA connections 125
Authenticating systems across PPC Gateway server connections 126
CICS link security 126
CICS user security. 127
Method one:. 127
Method two: 129
Setting the RemoteSysSecurity attribute to trusted or verify 130
Setting up a CICS region to flow user IDs . . . 130
Setting up a CICS region to flow passwords . . 131
Receiving user IDs from SNA-connected systems 133
Link security and user security compared 136
How the resource definition security attributes are used 137
Examples of link and user security 139
Security and function shipping 141
Security requirements for the mirror transaction 141
Outbound security checking for function-shipping requests 141
The NOTAUTH condition 141
EXEC CICS start requests from a remote system 142
Using CRTE and CESN to sign on from a remote system 142
Intersystem security checklist 143
When link security or user security is used . . 143
When link security is used 143
When user security is used. 143
Common configuration problems with intersystem security 144
Remote user is logged in to wrong user ID . . 144
User IDs are not flowed to a remote system . . 145
Flowed user IDs are not received from a remote system 145
CD RSLKeyMask and TSLKeyMask attributes are ignored 145
Unexpected message ERZ045006W - no obvious effect on security 145
Access is unexpectedly restrictive (message ERZ045006W) 145
Access unexpectedly rejected or granted on inbound request 146

Chapter 7. Data conversion 147

Introduction to data conversion 147
SBCS, DBCS, and MBCS data conversion considerations 148
Numeric data conversion considerations . . . 149
Summary of data conversion for CICS intercommunication functions 152
Code page support 153
Data conversion for function shipping, distributed program link and asynchronous processing . . . 156
Which system does the conversion 156
How code page information is exchanged . . . 157
When TXSeries for Multiplatforms does not convert the data 157
Avoiding data conversion 158
Standard data conversion for function shipping, DPL and asynchronous processing 159
Non-standard data conversion (DFHUCNV) for function shipping, DPL and asynchronous processing 160
Data conversion for transaction routing. 164
When is DFHTRUC called 165
Parameters passed to DFHTRUC 165
Writing your own version of DFHTRUC . . . 166
Installing your version of DFHTRUC into the region 168
Data conversion for distributed transaction processing (DTP) 169
Summary of data conversion 169
Data conversion for function shipping 169
Data conversion for transaction routing. . . . 170
Data conversion for distributed transaction processing 171

Part 3. Operating an SNA intercommunication environment . 173

Chapter 8. Creating and using a PPC Gateway server in a CICS environment 175
Overview of the PPC Gateway server 175
Sharing server names between a CICS region and a PPC Gateway server 177
Process for making an intersystem request from a CICS region through a PPC Gateway server . 178
Process for making an intersystem request to a CICS region through a PPC Gateway server . . 179
Exchange log names (XLN) process 180
Choosing a management method 181
Using the CICS control program (cicscp) configuration tool to manage a PPC Gateway server 182
Creating a PPC Gateway server 182
Starting a PPC Gateway server 183
Stopping a PPC Gateway server 184
Destroying a PPC Gateway server 184
Using CICS commands to manage a PPC Gateway server 184
Creating a PPC Gateway server 184
Starting a PPC Gateway server 186
Stopping a PPC Gateway server 187
Destroying a PPC Gateway server 188
Changing the attributes of a PPC Gateway server 188
Listing the PPC Gateway servers defined on a machine 188
Viewing the attributes of a PPC Gateway server 188
Listing the PPC Gateway servers running on a machine 189
Releasing the lock of a PPC Gateway server . . 189
Using the IBM TXSeries Administration Tool to manage a PPC Gateway server 190
Creating a PPC Gateway server 190
Starting a PPC Gateway server 191
Stopping a PPC Gateway server 193
Modifying the attributes of a PPC Gateway server 193
Destroying a PPC Gateway server 194
Using SMIT to manage a PPC Gateway server . . 195
Creating a PPC Gateway server 195
Starting a PPC Gateway server 197
Stopping a PPC Gateway server 198
Viewing and changing the attributes of a PPC Gateway server. 199
Destroying a PPC Gateway server 200
Using ppcadmin commands 200
Viewing CICS configuration in the PPC Gateway server. 203
Viewing XLN process status in the PPC Gateway server. 204
Requesting the XLN process with a remote SNA system 205
Viewing pending resynchronizations in the PPC Gateway server. 206
Canceling pending resynchronizations in the PPC Gateway server 206
Viewing intersystem requests running in the PPC Gateway server 207
Viewing LUWs in the PPC Gateway server . . 208
Changing CICS intercommunication definitions for use with a PPC Gateway server 210

Chapter 9. Intersystem problem determination 213
Intersystem problem solving process 213
Step 1. Understand the failure scenario 213
Step 2. Identify the failing component 214
Step 3. Fix the problem 215
Getting further help 215
Common intercommunication errors. 216
Symptom: cannot start (acquire) SNA sessions 216
Symptom: the PPC Gateway server fails to start 217
Symptom: CICS will not configure the PPC Gateway server. 217
Symptom: CICS will not configure my transactions in the PPC Gateway server . . . 217
Symptom: CICS cannot configure the PPC Gateway server. 218
Symptom: CICS for MVS/ESA transactions abend AXFX. 219
Symptom: SNA exchange log names (XLN) fails 219
Symptom: all inbound requests fail 220
Symptom: all outbound requests fail 220
Symptom: invalid or unrecognizable data is passed to applications 221
Symptom: applications fail because of security violations. 221
Symptom: Transaction routing to CICS Transaction Server for z/OS fail with communications error 15a00007/a0000100 . . . 221
Symptom: Transaction routing causes screen errors 222
Symptom: CICS fails to detect a predefined CD entry, and autoinstalls another. 222
SNA product-specific errors 223
CICS error codes 223
PPC Gateway server problem determination . . . 228
Format of PPC Gateway server messages and trace 228
Tracing a PPC Gateway server. 230
PPC Gateway server message descriptions . . 232

Part 4. Writing application programs for intercommunication . 245

Chapter 10. Distributed program link (DPL) 247
Comparing DPL to function shipping 247
BDAM files, and IMS, DL/I, and SQL databases 248
Performance optimization for DPL 248
Restrictions on application programs that use DPL 249
Security and DPL 250
Abends when using DPL 251
Taking sync points when using DPL or function shipping 251

Multiple DPL links to the same region 251
Two ways to implement DPL in your application program 252
Implicitly specifying the remote system . . . 252
Explicitly specifying the remote system. . . . 252
Using the dynamic distributed program link user exit. 252
Information passed to the user exit 253
Initial invocation of the user exit 253
Changing the target CICS system. 254
Changing the remote program name 255
Changing the mirror transaction name 255
Changing the user ID 255
Invoking the user exit at end of routed program 255

Chapter 11. Function shipping 257
How to use function shipping. 257
Two ways to use function shipping 257
Serial connections 257
CICS file control data sets 258
Transient data 258
Local and remote names. 258
Synchronization 258
Data security and integrity 258
Application programming for function shipping 259
File control 259
Temporary storage 260
Transient data 260
Exceptional conditions 260

Chapter 12. Transaction routing . . . 263
Initiating a transaction from a terminal. 263
The initiating terminal 263
The initiated transaction. 263
Application programming for transaction routing 264
Pseudo-conversational transactions 264
The terminal in the application-owning region 264
Using the assign command in the application-owning region 265
Automatic transaction initiation (ATI) 266
Terminal definitions for ATI 266
Transaction definitions for ATI 266
Indirect links for transaction routing. 266
Dynamic transaction routing 267
Writing a dynamic transaction routing user exit 268

Chapter 13. Asynchronous processing 273
Security considerations 273
How to use asynchronous processing 273
Two ways to initiate asynchronous processing 273
Starting and canceling remote transactions . . 274
The started transaction 276
Application programming for asynchronous processing 277
Starting a transaction on a remote region . . . 277
Exceptional conditions for the EXEC CICS START command 277
Retrieving data associated with a remotely issued start request 277

Chapter 14. Distributed transaction processing (DTP). 279
Concepts of distributed transaction processing (DTP) 279
Conversations 280
Conversation states 280
Distributed processes. 281
Maintaining data integrity 282
Designing distributed processes 283
Structuring distributed transactions 283
Designing conversations. 285
Writing programs for CICS DTP 286
Conversation initiation and the front-end transaction 286
Back-end transaction initiation. 289
The back-end transaction fails to start 291
Transferring data on the conversation 291
Communicating errors across a conversation 295
Safeguarding data integrity. 296
Ending the conversation. 298
Checking the outcome of a DTP command . . 300
Testing the conversation state 303
Initial states 303
Summary of CICS commands for APPC mapped conversations 304
Using DTP to check the availability of the remote system 304

Chapter 15. Sync pointing a distributed process. 307
The EXEC CICS SYNCPOINT command 307
The EXEC CICS ISSUE PREPARE command . . . 308
The EXEC CICS SYNCPOINT ROLLBACK command 308
When a backout is required 309
Synchronizing two CICS systems. 309
SYNCPOINT in response to SYNCPOINT . . . 309
SYNCPOINT in response to ISSUE PREPARE 310
SYNCPOINT ROLLBACK in response to SYNCPOINT ROLLBACK 311
SYNCPOINT ROLLBACK in response to SYNCPOINT 311
SYNCPOINT ROLLBACK in response to ISSUE PREPARE 312
ISSUE ERROR in response to SYNCPOINT . . 312
ISSUE ERROR in response to ISSUE PREPARE 313
ISSUE ABEND in response to SYNCPOINT . . 313
ISSUE ABEND in response to ISSUE PREPARE 314
Synchronizing three or more CICS systems . . . 314
Sync point in response to EXEC CICS SYNCPOINT 314
Sync point rollback in response to EXEC CICS SYNCPOINT 316
Conversation failure and the indoubt period . . 317

Part 5. Appendixes 319

Appendix A. DFHCNV - The data conversion macros 321

Using the DFHCNV macros 321
Examples of DFHCNV macros 322
Flow of DFHCNV macro sequence 324
The DFHCNV TYPE=INITIAL macro 324
The DFHCNV TYPE=ENTRY macro. 325
The DFHCNV TYPE=KEY macro. 327
The DFHCNV TYPE=SELECT macro 327
The DFHCNV TYPE=FIELD macro 328
The DFHCNV TYPE=FINAL macro 329

Appendix B. The conversation state tables. 331
How to use the state tables. 331
The state numbers. 331
The state conversation table notes 331
Synchronization level 0 conversation state table 333
Synchronization level 1 conversation state table 335
Synchronization level 2 conversation state table 338

Appendix C. Migrating DTP applications 343
Migrating to LU 6.2 APPC mapped conversations 343
Differences between SNA and TCP/IP for DTP . . 343
Allocating a conversation 344
Use of conversation identifiers (CONVIDs) . . . 344
State after backout. 344
Terminating a synchronization level 2 conversation 344
The EXEC CICS WAIT and EXEC CICS SEND WAIT commands 345
Transaction identifiers 345
Migration considerations for function shipping . . 345

Appendix D. Data conversion tables (CICS on Windows Systems and CICS for Solaris only) 347

Bibliography. 355

Notices 357
Trademarks and service marks 358

Index 361

Figures

1. Communicating with CICS family TCP/IP support 7
2. Communicating with CICS PPC TCP/IP . . . 8
3. Using local SNA support to communicate across SNA 9
4. Using a PPC Gateway server to communicate across SNA 10
5. Communications using TCP/IP, a PPC Gateway server, and SNA. 12
6. CICS region acting as a client gateway . . . 13
7. A simple transaction 16
8. A distributed application 16
9. Acknowledgment processing. 17
10. Listener Definitions (LD) entry 29
11. SMIT panel for adding an LD entry for CICS family TCP/IP support. 33
12. General Communications Definitions (CD) panel. 37
13. TCP/IP Communications Definitions (CD) panel. 37
14. Security Communications Definitions (CD) panel. 38
15. SMIT panel for adding a CD entry for CICS family TCP/IP support. 41
16. IBM CICS Client Server and Driver examples 42
17. CICS family TCP/IP example 44
18. CICS OS/2 System Initialization Table . . . 45
19. CICS OS/2 Connection and session table 45
20. Listener Definitions (LD) entry 45
21. Communications Definitions (CD) entry 46
22. Communications Definitions (CD) entry 46
23. Listener Definitions (LD) entry 46
24. Communications Definitions (CD) entry 47
25. General Communications Definitions (CD) panel. 48
26. TCP/IP Communications Definitions (CD) panel. 48
27. Security Communications Definitions (CD) panel. 49
28. Two regions communicating across TCP/IP 50
29. Two regions communicating across TCP/IP 51
30. SMIT panel for adding a CD entry for CICS PPC TCP/IP support 53
31. An example network 69
32. An example network using local SNA support 71
33. Listener Definition window 78
34. Region Definition Properties window 79
35. Communication Definition window: General tab 80
36. Communication Definition window: SNA tab 81
37. Communication Definition window: Security tab 82
38. Add Listener SMIT panel 83
39. Show/Change Region SMIT panel (Part 1) 85
40. Show/Change Region SMIT panel (Part 2) 86
41. Add Communication SMIT panel 88
42. An example network using PPC Gateway server SNA support. 92
43. Region Definition Properties window 99
44. Communication Definition window: General tab 100
45. Communication Definition window: SNA tab 101
46. Communication Definition window: Security tab 102
47. Add Gateway SMIT panel 104
48. Show/Change Region SMIT panel 105
49. Add Communication SMIT panel. 107
50. The bind password exchange 125
51. Representation of numbers 150
52. Sample C function for halfword swap 151
53. Sample C function for fullword swap 151
54. Example record structure 152
55. Data conversion for function shipping 157
56. Avoiding data conversion 158
57. Directory structure of the PPC Gateway server (assumes an Open Systems platform) . 176
58. How server names are shared between a CICS region and a PPC Gateway server. . . 177
59. Process for making an intersystem request from a CICS region through a PPC Gateway server 179
60. Process for making an intersystem request to a CICS region through a PPC Gateway server. 180
61. Listing PPC Gateway servers with the cicsget command. 188
62. PPC Gateway server attributes for cicsgwy 189
63. Default PPC Gateway server attributes 189
64. ps -ef | grep ppcgwy command 189
65. New PPC GWY Server screen with example configuration 191
66. Start PPC GWY server screen 192
67. Stop PPC GWY server screen 193
68. PPC Gateway server Properties screen—General tab 194
69. Destroy PPC GWY Server screen 195
70. Create PPC Gateway Server screen 197
71. Auto Start a PPC Gateway Server screen 198
72. Shutdown a PPC Gateway Server screen 199
73. Show/Change PPC Gateway Server screen 200
74. Using ppcadmin commands in interactive mode 202
75. Using ppcadmin commands in command mode 202
76. ppcadmin list luentries command. 203
77. ppcadmin query luentry command 204
78. ppcadmin list xlns command 205
79. ppcadmin force xln command with successful results 205
80. ppcadmin force xln command with unsuccessful results 206
81. ppcadmin query resync command 206
82. ppcadmin cancel resync command 207

83. ppcadmin list convs command 207
84. ppcadmin query conv command 208
85. ppcadmin list luws command 209
86. ppcadmin query luw command 209
87. Example PPC Gateway server message 228
88. Distributed program link 248
89. Serially connected systems 248
90. Transferring a temporary storage queue 284
91. Starting a conversation at synchronization level 2 287
92. Startup of a back-end APPC mapped transaction at synchronization level 2 . . . 290
93. Transferring data on a conversation at synchronization level 2 292
94. Checking the outcome of an EXEC CICS SEND INVITE WAIT command 293
95. Checking the outcome of an EXEC CICS RECEIVE command 294
96. EXEC CICS SYNCPOINT in response to EXEC CICS SEND followed by EXEC CICS SYNCPOINT on a conversation 309
97. EXEC CICS SYNCPOINT in response to EXEC CICS SEND INVITE followed by EXEC CICS SYNCPOINT on a conversation . . . 310
98. EXEC CICS SYNCPOINT in response to EXEC CICS SEND LAST followed by EXEC CICS SYNCPOINT on a conversation . . . 310
99. EXEC CICS SYNCPOINT in response to EXEC CICS ISSUE PREPARE on a conversation 311
100. EXEC CICS SYNCPOINT ROLLBACK in response to EXEC CICS SYNCPOINT ROLLBACK on a conversation 311
101. EXEC CICS SYNCPOINT ROLLBACK in response to EXEC CICS SYNCPOINT on a conversation 312
102. EXEC CICS SYNCPOINT ROLLBACK in response to EXEC CICS ISSUE PREPARE on a conversation 312
103. EXEC CICS ISSUE ERROR in response to EXEC CICS SYNCPOINT on a conversation . 313
104. EXEC CICS ISSUE ERROR in response to EXEC CICS ISSUE PREPARE on a conversation 313
105. EXEC CICS ISSUE ABEND in response to EXEC CICS SYNCPOINT on a conversation . 314
106. EXEC CICS ISSUE ABEND in response to EXEC CICS ISSUE PREPARE on a conversation 314
107. A distributed sync point with all partners running on CICS 315
108. Rollback during distributed sync pointing 317
109. Sample macro source for data conversion 322
110. Record layout for VSAM99 323
111. Description for record layout for VSAM99 323
112. Example of DFHCNV macro sequence 324

Tables

1. Getting started road map. xiii
2. Conventions that are used in this book xiv
3. Road map 1
4. Road map 5
5. Summary of communication methods across TCP/IP and SNA 13
6. Synchronization level used on outbound intersystem requests 18
7. Road map 23
8. Where to next. 26
9. Road map 41
10. CICS family TCP/IP configuration example 44
11. Road map 54
12. Comparison of CICS resource definitions TCP/IP connections. 54
13. How CD attributes are set in an autoinstalled CD entry 55
14. Fields in the DFHCCINX COMMAREA structure, CICS_CCINX_Parameters 59
15. CD attributes that can be changed by DFHCCINX 62
16. Communications products for various platforms to enable local SNA support . . . 70
17. Comparison of CICS Communications Definitions (CD) for local SNA connections . . 72
18. Comparison of CICS Listener Definitions (LD) for local SNA connections. 73
19. Comparison of CICS Region Definitions (RD) for local SNA connections. 74
20. Comparison of CICS Transaction Definitions (TD) for local SNA connections 74
21. Communications products for various platforms to enable PPC Gateway server SNA support 90
22. Comparison of CICS Communications Definitions (CD) for PPC Gateway server SNA connections 93
23. Comparison of CICS Gateway Definitions (GD) for PPC Gateway server SNA connections . . 94
24. Comparison of CICS Region Definitions (RD) for PPC Gateway server SNA connections . . 95
25. Comparison of CICS Transaction Definitions (TD) for PPC Gateway server SNA connections 95
26. Security keys assigned to example systems 128
27. Security keys assigned to example users 128
28. Which user ID is flowed when local task issues a request 130
29. Comparing security options between platforms 134
30. Defining security parameters 135
31. Example resource definitions for link security 140
32. CICS Shortcodes and code pages that TXSeries for Multiplatforms supports . . . 153
33. Comparison of data conversion with other CICS systems 158
34. Resource class abbreviations and resource types 160
35. Road map 173
36. Communications products for various platforms 175
37. ppcadmin commands that are applicable to the CICS environment 201
38. The effect of updating CICS intercommunication resource definitions . . 210
39. Sources of information for following the path of a request 214
40. Required information for the service representative 215
41. Message types 228
42. Trace types 229
43. Authority required by PPC Gateway Server 243
44. Road map 245
45. Indirect links for transaction routing 267
46. Conversation states available for each synchronization level 280
47. CICS synchronization commands for use across transactions 296
48. EIB and conversation state request responses 297
49. Indications of partner response 298
50. Command sequence for synchronization levels 299
51. Interaction between some EIB fields 301
52. RECEIVE and CONFIRM flags. 302
53. Relationship of cvda codes to conversation states 303
54. CICS commands used in APPC mapped conversations 304
55. Synchronization level 0 conversation state table (states 1 to 8). 333
56. Synchronization level 0 conversation state table (states 9 to 13) 334
57. Synchronization level 1 conversation state table (states 1 to 8). 335
58. Synchronization level 1 conversation state table (states 9 to 13) 336
59. Part 1 of Synchronization level 2 conversation state table (states 1 to 8) 338
60. Part 1 of Synchronization level 2 conversation state table (states 9 to 13) 339
61. Part 2 of Synchronization level 2 conversation state table (states 1 to 8) 341
62. Part 2 of Synchronization level 2 conversation state table (states 9 to 13) 342
63. SBCS data conversion table 347
64. DBSC and mixed data conversion table 351
65. SBCS and DBCS data conversion table 354

About this book

This book describes how CICS® systems communicate with other CICS systems. It describes both how TXSeries® for Multiplatforms regions communicate with other TXSeries for Multiplatforms regions, as well as with other members of the CICS family (such as CICS Transaction Server for z/OS® and IBM CICS Universal Clients), and with any application that supports the SNA LU 6.2 protocol.

It describes how to design and implement a network configuration for a TCP/IP and an SNA network, how to configure the CICS resources, and how to design, write, and manage the application programs.

Who should read this book

The book is intended for system administrators who design, configure, and manage a network of interconnected CICS systems.

You should be familiar with the operation and configuration of a basic CICS region. You should have a understanding of network operations for either TCP/IP or SNA networks. Familiarity with the implementation and terminology used by the partner system with which the CICS region communicates is also useful

Document organization

Table 1. Getting started road map

If you want to...	**Refer to...**
Read a summary of CICS communications	Part 1, "Intercommunication planning," on page 1
Read how to configure CICS resources to enable your system to communicate with other systems	Part 2, "Configuring for intercommunication," on page 23
Read how to configure an SNA product	*TXSeries for Multiplatforms Using IBM® Communications Server for AIX® with CICS*, *TXSeries for Multiplatforms Using IBM Communications Server for Windows® Systems with CICS*, *TXSeries for Multiplatforms Using Microsoft® SNA Server with CICS*, *TXSeries for Multiplatforms Using HP-UX SNAplus2 with CICS* , and *TXSeries for Multiplatforms Using SNAP-IX for Solaris with CICS*
Read how to manage your CICS system when it is communicating with other systems	Part 3, "Operating an SNA intercommunication environment," on page 173
Read how to write CICS transaction programs that communicate with other systems	Part 4, "Writing application programs for intercommunication," on page 245

Conventions used in this book

TXSeries for Multiplatforms documentation uses the following typographical and keying conventions.

Table 2. Conventions that are used in this book

Convention	Meaning
Bold	Indicates values that you must use literally, such as commands, functions, and resource definition attributes and their values. When referring to graphical user interfaces (GUIs), bold also indicates menus, menu items, labels, buttons, icons, and folders.
`Monospace`	Indicates text that you must enter at a command prompt. Monospace also indicates screen text and code examples.
Italics	Indicates variable values that you must provide (for example, you supply the name of a file for *file_name*). Italics also indicates emphasis and the titles of books.
< >	Encloses the names of keys on the keyboard.
<**Ctrl-***x*>	Where *x* is the name of a key, indicates a control-character sequence. For example, <**Ctrl-c**> means hold down the **Ctrl** key while you press the **c** key.
<**Return**>	Refers to the key labeled with the word Return, the word Enter, or the left arrow.
`%`	Represents the UNIX® command-shell prompt for a command that does not require **root** privileges.
`#`	Represents the UNIX command-shell prompt for a command that requires **root** privileges.
`C:\>`	Represents the Windows command prompt.
>	When used to describe a menu, shows a series of menu selections. For example, ″Select **File > New**″ means ″From the **File** menu, select the **New** command.″
Entering commands	When instructed to “enter” or “issue” a command, type the command and then press <**Return**>. For example, the instruction “Enter the **ls** command” means type **ls** at a command prompt and then press <**Return**>.
[ ]	Encloses optional items in syntax descriptions.
{ }	Encloses lists from which you must choose an item in syntax descriptions.
|	Separates items in a list of choices enclosed in **{ }** (braces) in syntax descriptions.
...	Ellipses in syntax descriptions indicate that you can repeat the preceding item one or more times. Ellipses in examples indicate that information was omitted from the example for the sake of brevity.
IN	In function descriptions, indicates parameters whose values are used to pass data to the function. These parameters are not used to return modified data to the calling routine. (Do *not* include the IN declaration in your code.)
OUT	In function descriptions, indicates parameters whose values are used to return modified data to the calling routine. These parameters are not used to pass data to the function. (Do *not* include the OUT declaration in your code.)

Table 2. Conventions that are used in this book (continued)

Convention	Meaning
INOUT	In function descriptions, indicates parameters whose values are passed to the function, modified by the function, and returned to the calling routine. These parameters serve as both IN and OUT parameters. (Do *not* include the INOUT declaration in your code.)
$CICS	Indicates the full path name of the location in which the CICS product is installed; for example, **/usr/lpp/cics** on AIX. If the CICS environment variable is set to the product path name, you can use the examples exactly as shown in this book; otherwise, you must replace all instances of $CICS with the CICS product path name.
CICS on Open Systems	Refers collectively to the CICS product for all supported UNIX platforms.
TXSeries for Multiplatforms	Refers collectively to the CICS for AIX, CICS for HP-UX, CICS for Solaris, and CICS for Windows products.
CICS	Refers generically to the CICS for AIX, CICS for HP-UX, CICS for Solaris, and CICS for Windows products. Other CICS products in the CICS Family are distinguished by their operating system (for example, IBM mainframe-based CICS for the ESA, MVS™, and VSE platforms).

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information. If you have any comments about this book or any other TXSeries for Multiplatforms documentation, send your comments by e-mail to idrcf@hursley.ibm.com. Be sure to include the name of the book, the document number of the book, the version of TXSeries for Multiplatforms, and, if applicable, the specific location of the information you are commenting on (for example, a page number or table number).

Part 1. Intercommunication planning

This part introduces CICS intersystem communications. It describes the functions that are available, and helps you plan and design your network.

Table 3. Road map

If you want to...	**Refer to...**
Read about the network protocols CICS supports	"Network protocols" on page 3
Read about the intersystem communication functions that CICS supports	"Overview of the intercommunication facilities" on page 3
Read about the functions that are supported by IBM CICS clients	"IBM CICS Universal Client products" on page 4
Read about how to design your network	"Designing your network configuration" on page 5

Chapter 1. Introduction to CICS intercommunication

In a multiple system environment, TXSeries for Multiplatforms regions can communicate with other systems to:

- Provide users of the local region with services that are held on remote systems
- Provide users on remote systems with services that are held on the local region

This communication is achieved by networking the systems so that they can cooperate directly and share data and applications. The communication between the interconnected systems is referred to as *intercommunication*.

Network protocols

When two systems communicate, they need to agree on the set of rules that they use to interpret the data that they exchange. These rules are known as *network protocols* and they are defined in a *network architecture*. TXSeries for Multiplatforms intercommunication is based on the IBM Systems Network Architecture (SNA) LU 6.2 protocol. This protocol, which is often referred to as *advanced program-to-program communications (APPC)*, was developed to handle the needs of two systems that want to share data and applications. Therefore, it is ideally suited to the CICS intercommunication environment.

TXSeries for Multiplatforms supports intercommunication across an SNA network between a local region and the following:

- Other TXSeries for Multiplatforms regions
- Other CICS products such as CICS Transaction server for z/OS and CICS/400® regions
- IBM CICS Universal Clients
- Applications on systems that support the SNA LU 6.2 protocol

In addition, TXSeries for Multiplatforms can emulate SNA LU 6.2 across TCP/IP networks. This allows a local region to communicate with the following by means of TCP/IP:

- Other TXSeries for Multiplatforms regions
- IBM CICS Universal Clients

The method of connecting your TXSeries for Multiplatforms regions to SNA and TCP/IP networks is described in Chapter 2, “Intercommunication planning and system design,” on page 5.

Overview of the intercommunication facilities

The CICS intercommunication facilities that TXSeries for Multiplatforms supports simplify the operation of distributed systems. In general, this support extends the standard CICS facilities (such as reading and writing to files and queues) so that applications or users can use resources that are on remote systems without needing to know where the resources are. TXSeries for Multiplatforms supports the following CICS intercommunication facilities:

- **Distributed program link (DPL)** extends the use of the EXEC CICS LINK command to allow a CICS application program to link to a program that resides on a different CICS system.

- **Function shipping** allows an application program to access files, transient data queues, and temporary storage queues that belong to another CICS system.
- **Transaction routing** allows the execution of a transaction on a remote system. The transaction can display information on your terminal as if it were running on your local system.
- **Asynchronous processing** extends the EXEC CICS START command to allow an application to initiate a transaction to run on another CICS system. As with standard EXEC CICS START calls, the transaction that is requested in the START command runs independently of the application that is issuing the START command.
- **Distributed transaction processing (DTP)** uses additional EXEC CICS commands that allow two applications to run on different systems and pass information between themselves. These EXEC CICS commands map to the LU 6.2 mapped conversation verbs that are defined in the SNA Architecture.

  DTP is the only CICS intercommunication facility that can be used to communicate with non-CICS applications. The non-CICS applications must use *advanced program-to-program communications (APPC)* protocol.

IBM CICS Universal Client products

TXSeries for Multiplatforms provides CICS server support for the IBM CICS Universal Client products. This means that TXSeries for Multiplatforms systems can provide CICS server function to multiple workstations that are running the IBM CICS Universal Client products.

The functions that are available to the client include:

- **External call interface (ECI).** This interface enables a non-CICS application program that is running in the client workstation, to call a CICS program in the server, and run it as a subroutine.
- **External program interface (EPI).** This interface enables an application program to run in the client, and invoke a CICS transaction on the server that runs as though it was started from a 3270 terminal. The transaction returns a 3270 data stream to the client, which can capture it and present it in a graphical user interface (GUI).
- **Terminal emulation.** This enables the client workstation to function as a 3270 terminal.

CICS Family: Interproduct Communication explains how intercommunication between CICS family products is documented, and introduces the CICS intercommunication facilities. This information is necessary for anyone who is involved in the planning and implementation of communications between different CICS systems. *CICS Universal Client: Client Administration* explains the planning, configuration, and administration of the IBM CICS Universal Clients.

Chapter 2. Intercommunication planning and system design

This planning and system design chapter describes the services that are provided by TXSeries for Multiplatforms and its supporting products that enable it to communicate with remote systems.

Table 4. Road map

If you want to...	Refer to...
Read about how systems can be connected together	“Designing your network configuration”
Read about intersystem security issues	“Ensuring that the system is still secure” on page 14
Read about the integrity of data that is distributed across a network	“Ensuring data integrity with synchronization support” on page 16
Read about the conversion of data structures when they are exchanged between systems that use different ASCII or EBCDIC encoding formats	“Converting between EBCDIC and ASCII data” on page 19
Read about the performance of intersystem transactions	“Performance issues for intercommunication” on page 19
Read about the operation of your network	“Operational issues for intercommunication” on page 22
Read about bidirectional input and display	“Bidirectional input and display on AIX” on page 22

Together with information about your current systems, users and requirements, you can use these sections to plan for the implementation of network configuration and operations. Intercommunication requires cooperation between the administrators of all of systems in the network. Therefore, it is important that the administrators of the remote systems are involved in the planning process.

Designing your network configuration

TXSeries for Multiplatforms can communicate with remote systems across the following connections:
- TCP/IP networks
- Systems Network Architecture (SNA) networks

The following sections introduce the ways that your CICS region can be connected to other systems:
- “Communicating across TCP/IP connections”
- “Communicating across SNA connections” on page 8
- “Mixing the communications methods” on page 11
- “Summary of communication methods” on page 13

Communicating across TCP/IP connections

TCP/IP is a simple protocol that requires CICS to provide most of the support for intercommunications, such as the data formats that are used to send intersystem requests and the security that is needed to protect system resources.

TXSeries for Multiplatforms supports two types of communication over TCP/IP connections:

- **CICS family TCP/IP** support, which allows connectivity to TXSeries for Multiplatforms regions, and IBM CICS Universal Clients.
- **CICS PPC TCP/IP** support, which allows connectivity between CICS regions on the same machine.

"Using CICS family TCP/IP" and "Using CICS PPC TCP/IP" on page 7 give more detail on each of these types of TCP/IP communication.

Using CICS family TCP/IP

CICS family TCP/IP supports the following types of intersystem requests between TXSeries for Multiplatforms and CICS OS/2 regions:

- Function Shipping
- Transaction Routing
- Distributed Program Link (DPL)
- Asynchronous Processing

Distributed Transaction Processing (DTP) is not supported.

IBM CICS Universal Clients can also use CICS family TCP/IP to connect to a TXSeries for Multiplatforms region.

When the first request is made, a TCP/IP connection is acquired between the two systems. This connection remains acquired while both systems are active and it can carry many concurrent intersystem requests flowing in either direction.

A TXSeries for Multiplatforms region can be configured to accept TCP/IP connections on one or more TCP/IP ports in the local machine. It can also receive connection requests on one or more TCP/IP network adapters.

Note that the CICS family TCP/IP support does not provide the same level of security that is available with PPC Executive TCP/IP support. This issue is discussed in "Ensuring that the system is still secure" on page 14.

Figure 1 on page 7 shows a region that is running on a Windows platform and using CICS family TCP/IP to communicate with a CICS OS/2 system and an IBM CICS Universal Client.

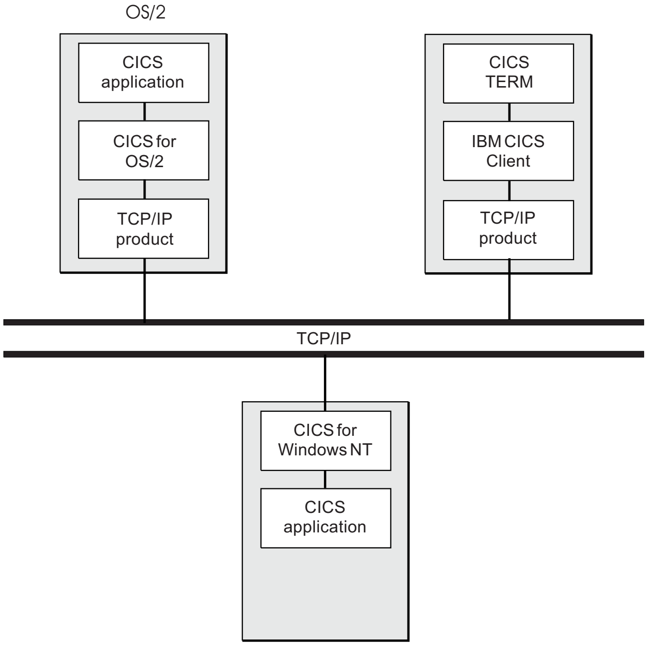


Figure 1. Communicating with CICS family TCP/IP support

Synchronization levels: All intersystem requests that are flowed on a CICS family TCP/IP connection use synchronization level 0 or 1. Synchronization levels are described in “Ensuring data integrity with synchronization support” on page 16.

Configuration information: The instructions for configuring your region to use CICS family TCP/IP are summarized in “Configuring CICS for CICS family TCP/IP support” on page 26.

Using CICS PPC TCP/IP

CICS PPC TCP/IP support allows all types of intercommunication between CICS regions on the same machine and is simple to configure.

When an intersystem request is made, CICS locates the remote region. Then, a TCP/IP connection is set up between the two regions. This connection is used exclusively by the intersystem request, and is closed down when the request has completed.

Figure 2 on page 8 shows the PPC Executive that is being used to connect applications that are running on two CICS regions on the same machine.

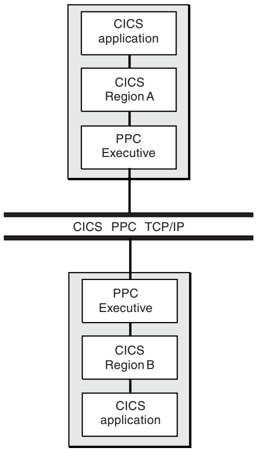


Figure 2. Communicating with CICS PPC TCP/IP

Synchronization levels: CICS PPC TCP/IP supports synchronization levels 0, 1 and 2. For more information about the different synchronization levels, refer to "Ensuring data integrity with synchronization support" on page 16.

Configuration information: The instructions for configuring a region to use CICS PPC TCP/IP are summarized in "Configuring CICS for CICS PPC TCP/IP support" on page 47.

Communicating across SNA connections

TXSeries for Multiplatforms regions can communicate across SNA with any system that supports advanced program-to-program communications (APPC). This includes CICS Transaction Server for z/OS, CICS/ESA, CICS/MVS, CICS/400, and CICS/VSE. They can communicate between all TXSeries for Multiplatforms regions. SNA can be used to communicate with IBM CICS Universal Clients. (In SNA, APPC is used synonymously with the term *LU Type 6.2*, or *LU 6.2*.)

Two methods of SNA communication are available in TXSeries for Multiplatforms:
- Local SNA support
- SNA support that uses the CICS PPC Gateway server

Both of these methods of providing an SNA connection support all the CICS intercommunication facilities to other CICS systems, and DTP is supported to non-CICS systems.

Using local SNA support to communicate across SNA

Local SNA support provides the fastest SNA connectivity that CICS offers. It enables TXSeries for Multiplatforms applications to communicate with every other member of the CICS family.

IBM CICS Universal Clients can communicate with TXSeries for Multiplatforms and use local SNA support.

Local SNA support requires an appropriate SNA product to be installed and configured on the same machine as is the TXSeries for Multiplatforms region.

TXSeries for Multiplatforms supports the following SNA products:
- **On Windows systems:** Microsoft Microsoft SNA Server and IBM Communications Server
- **On AIX systems:** IBM Communications Server

- **On Solaris:** SNAP-IX
- **On HP:**HP-UX SNAplus2

Figure 3 shows TXSeries for Multiplatforms using local SNA support to communicate with other CICS systems and with other APPC applications. The term *APPC workstations* refers to any small computer that is running APPC applications.

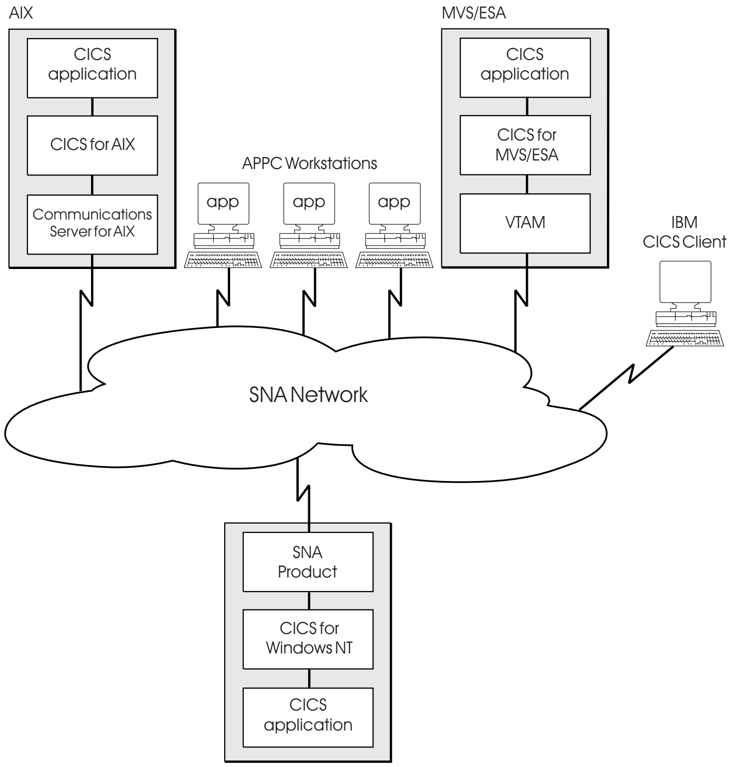


Figure 3. Using local SNA support to communicate across SNA

Synchronization levels: Local SNA support allows the use of synchronization levels 0 and 1. If you require synchronization level 2 support, you must use a PPC Gateway server. This is described in “Using a PPC Gateway server to communicate across SNA.” For information about the different synchronization levels, refer to “Ensuring data integrity with synchronization support” on page 16.

Configuration information: The instructions for configuring your region to use local SNA support are summarized in “Configuring CICS for local SNA support” on page 70.

Using a PPC Gateway server to communicate across SNA

CICS can communicate with an SNA network by using a PPC Gateway server. CICS communicates uses TCP/IP to communicate with the PPC Gateway server, and the PPC Gateway server provides a link to the SNA network.

The PPC Gateway server can be on the same machine as is your CICS region, or it can be on a different machine.

The PPC Gateway server uses an appropriate SNA product to connect to the remote SNA systems. This SNA product must be installed and configured on the machine on which the PPC Gateway server is running. For example, if the PPC

Gateway server is running on an RS/6000® machine, the SNA product is IBM Communications Server for AIX. (Check with your sales representative for full details of the supported SNA products for your PPC Gateway server machine.)

Figure 4 shows a TXSeries for Multiplatforms region using the PPC Executive to connect to a PPC Gateway server machine, which in turn connects to an SNA network.

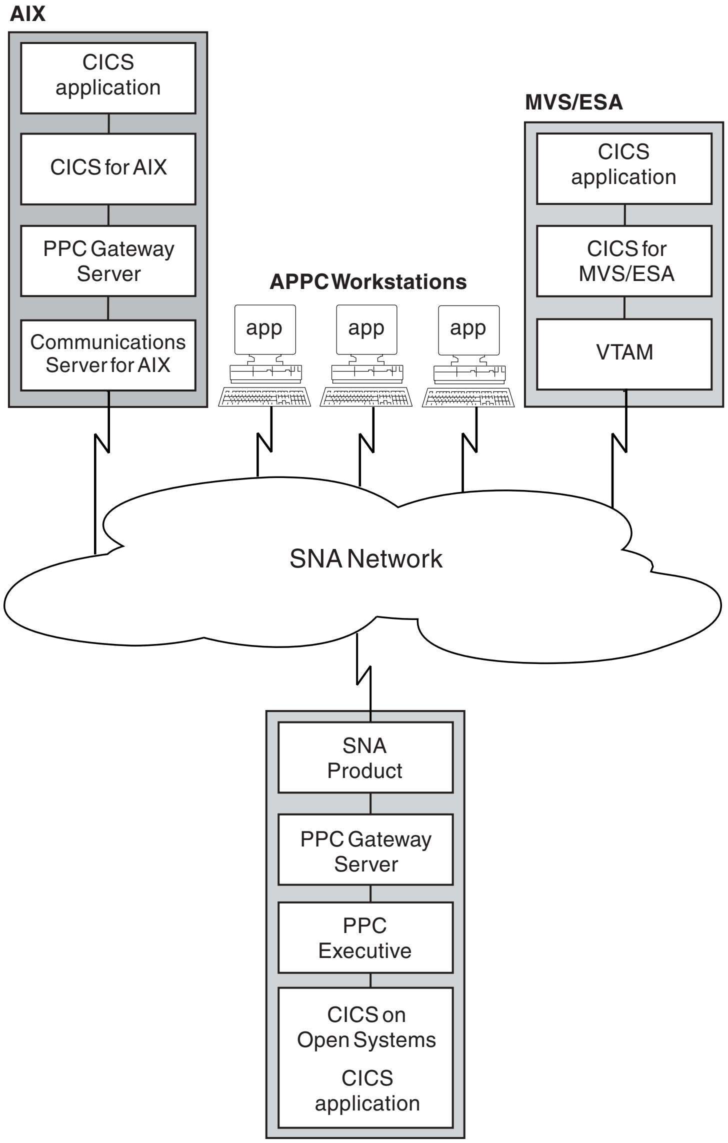


Figure 4. Using a PPC Gateway server to communicate across SNA

Using more than one PPC Gateway server with a CICS region can be useful for the following:

- To spread the network links from many remote machines across more than one gateway machine, and therefore across multiple SNA products
- To introduce some redundancy, so that if one PPC Gateway server fails, another is available as backup
- To spread the processing load across more than one PPC Gateway server

A PPC Gateway server can also be shared by a number of CICS regions. This can be desirable if your CICS regions do not make many SNA intercommunication requests. However, this setup must be used with care because the PPC Gateway server is only one operating system process and can become overwhelmed by too many intercommunication requests. In addition, problem determination can be more difficult if more than one region uses a PPC Gateway server.

Synchronization levels: A PPC Gateway server gives your region support for synchronization levels 0, 1, and 2. If you require only synchronization level 0 or 1 and you plan to put the SNA product on the machine on which your CICS region is running, you can use local SNA support. This is described in section "Using local SNA support to communicate across SNA" on page 8. For information about the different synchronization levels refer to "Ensuring data integrity with synchronization support" on page 16.

Configuration information: The instructions for configuring your region to use a PPC Gateway server are summarized in "Configuring CICS for PPC Gateway server SNA support" on page 89.

Mixing the communications methods

You can mix the communication methods that you use to connect your CICS regions. For example, a TXSeries for Multiplatforms region can use the PPC Executive to communicate with another TXSeries for Multiplatforms region over TCP/IP, while also communicating over SNA, by using both local SNA support and a PPC Gateway server.

Figure 5 on page 12 shows two CICS on Open Systems regions (Region B and Region C) that are communicating using CICS PPC TCP/IP with synchronization level 2. Region A and Region D are communicating with CICS TCP using synchronization level 1. The regions are also communicating across an SNA network to a CICS for AIX region, some APPC workstations, and a CICS Transaction Server for z/OS system. The communications with the CICS for AIX region and the workstations use local SNA support. Local SNA support was chosen because the APPC workstations do not support synchronization level 2. The communications with CICS Transaction Server for z/OS are through a PPC Gateway server because synchronization level 2 support is required. The PPC Gateway server is running on the same machine as is the CICS for AIX region and uses the same SNA product.

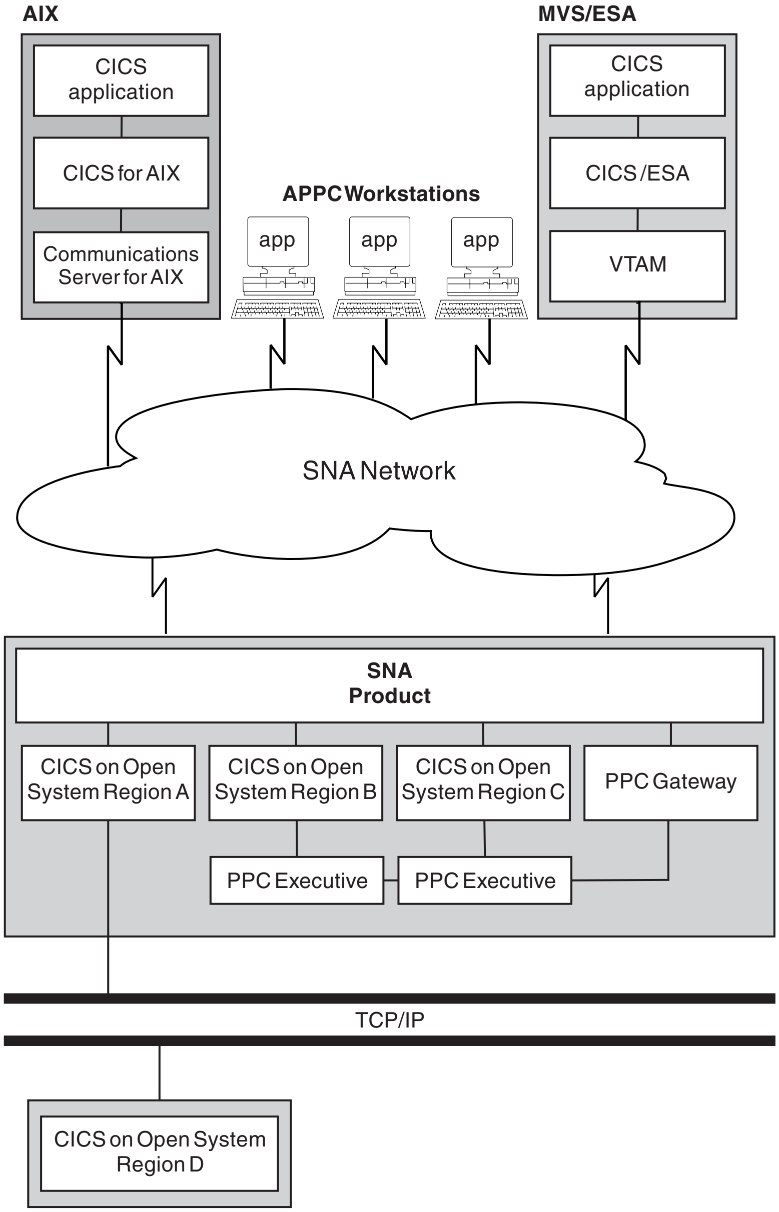


Figure 5. Communications using TCP/IP, a PPC Gateway server, and SNA

Figure 6 on page 13 shows a CICS region being used as a *client gateway*. In this setup, several IBM CICS Universal Clients are using CICS family TCP/IP to connect to the region . The region is then routing these requests across an SNA network to a mainframe CICS system.

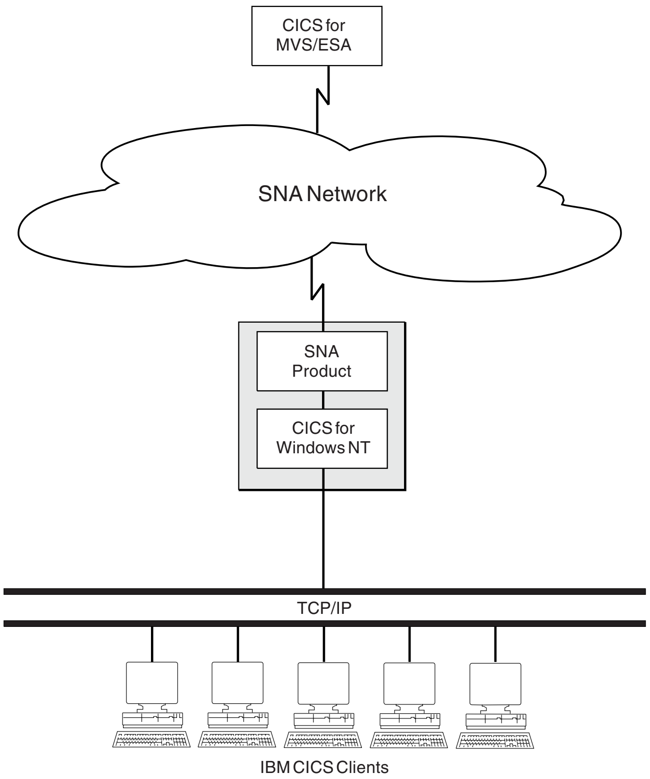


Figure 6. CICS region acting as a client gateway

For information about how to configure your region to communicate with remote systems, refer to Chapter 3, “Configuring CICS for TCP/IP,” on page 25 and Chapter 4, “Configuring CICS for SNA,” on page 67. For information about the different synchronization levels, refer to “Ensuring data integrity with synchronization support” on page 16.

Summary of communication methods

The following table summarizes the communication methods that CICS on Open Systems and CICS for Windows can use:

Table 5. Summary of communication methods across TCP/IP and SNA

Communication method	**Best for:**	**Restrictions:**
CICS family TCP/IP	Communicating at synchronization level 0 or 1 with TXSeries for Multiplatforms, IBM CICS Universal Clients, and RPC-only regions across TCP/IP	Distributed Transaction Processing (DTP) is not supported. CICS user security must be configured with care because it is not possible to reliably authenticate (identify) the remote system.
CICS PPC TCP/IP	Communicating at synchronization level 0, 1 or 2 with other TXSeries for Multiplatforms regions	Must be on the same machine.

Table 5. Summary of communication methods across TCP/IP and SNA (continued)

Communication method	Best for:	Restrictions:
Local SNA	Fast synchronization level 0 or 1 communication with remote LU 6.2 (APPC) systems. These connections can be used to connect to any CICS product.	A supported SNA product must be installed on the same machine as is the CICS region.
PPC Gateway	Synchronization level 0, 1, and 2 communication with remote LU 6.2 (APPC) systems. These connections can be used to connect to any CICS product.	When using Communications Server for AIX, Communications Server for Windows, and HP-UX SNAplus2, the PPC Gateway server must be installed on a machine along with a supported SNA product.

To make the best use of the information that is given in this chapter, consider your own installation: the systems you currently have and their capabilities, the requirements of your users, and the requirements of the systems that you want to have

If you are planning to connect your CICS region to a non-TXSeries for Multiplatforms product, refer to Chapter 3, "Configuring CICS for TCP/IP," on page 25, and Chapter 4, "Configuring CICS for SNA," on page 67.

Additional considerations

After planning your overall network configuration, review the following sections, which introduce the other factors to consider:
- "Ensuring that the system is still secure"
- "Ensuring data integrity with synchronization support" on page 16
- "Converting between EBCDIC and ASCII data" on page 19
- "Performance issues for intercommunication" on page 19
- "Operational issues for intercommunication" on page 22

Ensuring that the system is still secure

Connecting your region to other systems enables the users of these systems to share resources. CICS intercommunication provides extensions to CICS security that ensure that resources are shared only with authorized users.

The system that is receiving the request performs checking for the following purposes:
- Identifying the remote system that sent the request
- Identifying the user who initiated the request
- Controlling access to the CICS resources

Identifying the remote system

When systems connect, they pass identification information across the network. This identification information can be trusted only if the communication method provides the facilities to verify it. The process of verifying identification information is called *authentication*; it is achieved for each of the communication methods as follows:

CICS family TCP/IP

These connections provide no mechanisms for authenticating the remote system. CICS can extract the Internet Protocol (IP) address and listening port of the remote system, but no method exists for CICS to determine whether the connection request is coming from an unauthorized system that is deliberately trying to impersonate the genuine system. If a network is private and secure, this might not be a problem for you. If this potential security risk is a concern, consider using one of the other communication methods. For further information about authenticating remote systems when using CICS family TCP/IP connections, see "Authenticating systems across CICS family TCP/IP connections" on page 124.

CICS PPC TCP/IP

CICS PPC TCP/IP uses an RPC request to set up each intersystem request. As CICS PPC TCP/IP is used for communication between CICS regions on the same machine, there is no need to verify the identity of the system that is sending the request.

Local SNA

When an SNA connection is acquired, sessions are activated by using bind flows. You can configure SNA to automatically verify the identity of each system by defining a *bind password* for the two systems. Sessions are then activated only if both systems have the same bind password. For further information about authenticating remote systems when using local SNA connections, see "Authenticating systems across SNA connections" on page 125.

PPC Gateway

An SNA connection that uses a PPC Gateway server can be viewed in two parts. The SNA sessions from the PPC Gateway server to the remote system can be protected by using bind passwords, and the requests that flow between your region and the PPC Gateway server are protected because the region is on the same machine. For further information on authenticating remote systems when using PPC Gateway connections, see "Authenticating systems across PPC Gateway server connections" on page 126.

Identifying the user that initiated the request

Intersystem requests can also flow with the user ID (and optionally a password). CICS intercommunication security allows you to define the user ID that is assigned to the intersystem request, based on your knowledge of the security mechanisms that are available in the network and the remote system. It can be the user ID flowed from the remote system or a user ID that is defined locally. Transaction Security Level (TSL) keys and Resource Security Level (RSL) keys are then assigned to the request, based on the User Definition (UD) entry for the assigned user ID. You can also specify a TSL and RSL *key mask*, which restricts these keys further so that the intersystem request has less access to the system than does the local user who has the same user ID.

Intercommunication security is an extension of local security checking. Therefore, it is important that you understand how CICS implements local security. Refer to the *CICS Administration Guide* for further information about local security. Refer to Chapter 6, "Configuring intersystem security," on page 123, and "CICS user security" on page 127 for more information about how to configure CICS intercommunication security.

Ensuring data integrity with synchronization support

When designing applications that update data on more than one system, it is important to remember that network or system failures sometimes occur. These failures are difficult to manage in a distributed environment because the application is exposed to partial failures that do not occur in a single-system environment. This is best shown with an example.

Consider the case of a single system in which an application is moving records between two recoverable files, FILEA and FILEB, as shown in Figure 7. If a problem occurs, for example, data is corrupted in one of the files, a transaction abends, or a CICS region fails, CICS backs out the changes that were made before the problem occurred. Consequently, the files return to the state in which they were at the time the transaction was initially invoked.

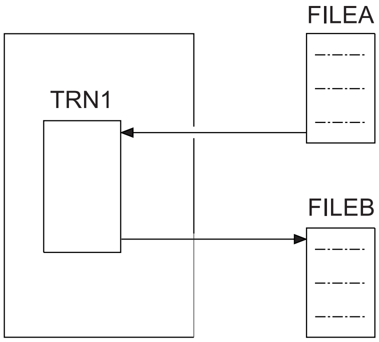


Figure 7. A simple transaction

Now consider the case where FILEB resides on a remote system, as shown in Figure 8. The work of the application is effectively split into two transactions, TRN1 and TRN2. Because the application is distributed, it is now difficult for TRN1 (in the local system) to detect problems with adding the record to FILEB (in the remote system). If TRN1 cannot detect these problems, it might delete a record from FILEA that has not been added to FILEB, and cause the record to be lost.

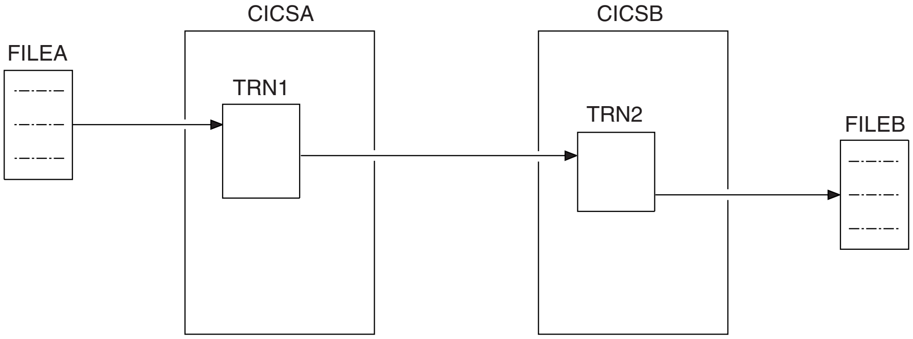


Figure 8. A distributed application

The solution is to use a form of *acknowledgment processing* so that the two transactions in the application can ensure that they each complete their task successfully. The exact form of this acknowledgment depends on the requirements of the application. For example, the application can be transferring a list of changed customer addresses between the two systems. As shown in Figure 9 on page 17 page 17, after TRN2 writes a record to FILEB, it can send a message to TRN1, indicating that the record can be deleted from FILEA.

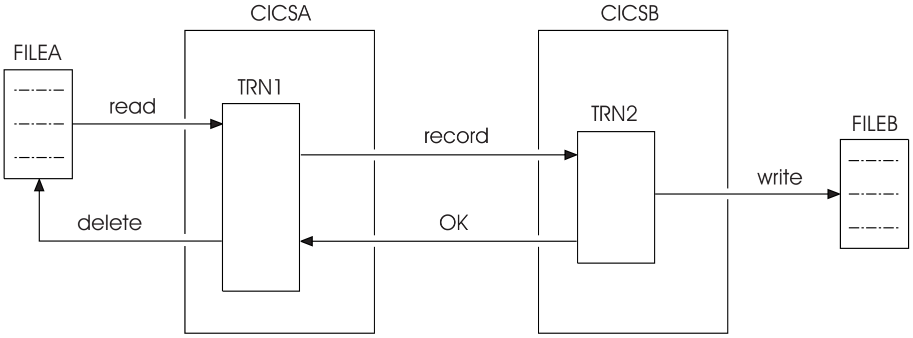


Figure 9. Acknowledgment processing

Thus a record can never be lost because TRN1 deletes the record from FILEA only when it knows that the record is written to FILEB. However, if TRN1 fails after it receives the message to delete the record, but before the record is actually deleted from FILEA, the record appears twice, once in FILEA and again in FILEB.

In this application, it probably does not matter whether a customer's address is updated once or twice. However, if the data in FILEA represents money in some form, and the act of transferring it between the systems effectively transfers the money from one bank account to another, two copies of the record cause the money to be transferred twice.

A mechanism is required to ensure that the deletion from FILEA always occurs if the write operation to FILEB is successful and never occurs if the write operation to FILEB is unsuccessful. This requires a much more sophisticated coordination between the two systems.

Because applications have different requirements, SNA defines three levels of support that allow an application to coordinate updates across a number of systems. These levels of support are called *synchronization levels*:

- **synchronization level 0 (NONE)**: SNA provides no synchronization support. The application must code its own.
- **synchronization level 1 (CONFIRM)**: SNA provides the ability to send simple acknowledgment requests.
- **synchronization level 2 (SYNCPOINT)**: SNA provides the ability for two or more CICS systems to treat the updates that are made by an application on these systems as one *logical unit of work (LUW)*. (LUW is a synonym of *unit of work (UOW)*). When the application requests a *synchronization point* (*sync point*) by using the EXEC CICS SYNCPOINT command or the final EXEC CICS RETURN command, the updates on the remote systems are *committed* (made permanent) if, and only if, updates to recoverable data that are made locally by the transaction are also committed. If a failure occurs (for example, the transaction abends) before all involved systems have agreed to commit, all the updates for the application on each of the systems are *backed out* (undone). Alternatively, all updates are backed out if the application issues the EXEC CICS SYNCPOINT ROLLBACK command.

When SNA systems first establish contact, they agree the maximum synchronization that they will use. The synchronization level for each individual task is then determined either by CICS or by the application program. TXSeries for

Multiplatforms supports all three synchronization levels across both SNA and TCP/IP. Refer to "Designing your network configuration" on page 5 for more information about these communication methods.

If the remote system and the communication network are able, TXSeries for Multiplatforms uses synchronization level 2 conversations on function shipping, asynchronous processing, and distributed program link (DPL) requests. Transaction routing requests from TXSeries for Multiplatforms always use synchronization level 1 conversations because they do not update recoverable data on the local region. However, TXSeries for Multiplatforms can receive synchronization level 2 transaction routing requests from CICS Transaction Server for z/OS, CICS/ESA, CICS/MVS, and CICS/VSE. Distributed transaction processing (DTP) uses the synchronization level that is requested on the SYNCLEVEL parameter of the EXEC CICS CONNECT PROCESS command. For further information on the CONNECT PROCESS command, refer to the *TXSeries for Multiplatforms Application Programming Reference*.

Note: For TXSeries for Multiplatforms to use synchronization level 2 with IBM mainframe-based CICS systems, they must be at these levels:
- CICS/MVS®: 2.1.2
- CICS/ESA®: 3.3 or higher
- CICS/VSE®: 2.2 or higher

The following table summarizes the synchronization level that is used on outbound intersystem requests. Refer to the following legend:

FS	Function shipping
TR	Transaction routing
AP	Asynchronous processing
DPL	Distributed program link
DTP	Distributed transaction processing
NS	Not supported

Table 6. Synchronization level used on outbound intersystem requests

Function	**FS**	**TR**	**AP**	**DPL**	**DTP**
Synchronization level chosen by:	**CICS**	**CICS**	**CICS**	**CICS**	**Application**
Request to CICS/ESA, CICS/MVS, CICS/VSE, or CICS/400 using local SNA	1	1	1	1	0 or 1
Request to CICS/ESA, CICS/MVS, CICS/VSE or CICS/400 using a PPC Gateway and SNA	2	1	2	2[1]	0, 1, or 2
Request to CICS OS/2 over SNA	1	1	1	1	0 or 1
Request to CICS OS/2, or TXSeries for Multiplatforms over CICS family TCP/IP	1	1	1	1	NS
Request to TXSeries for Multiplatforms using CICS PPC TCP/IP	2	1	2	2[1]	0, 1, or 2
Request to TXSeries for Multiplatforms over SNA when both systems are using a PPC Gateway	2	1	2	2[1]	0, 1, or 2
Request to TXSeries for Multiplatforms over SNA when one or both regions are using local SNA	1	1	1	1	0 or 1

Table 6. Synchronization level used on outbound intersystem requests (continued)

Function	FS	TR	AP	DPL	DTP
Request to non-CICS system over SNA	NS	NS	NS	NS	0, 1, or 2
Note: 1. DPL requests specifying SYNCONRETURN option use synchronization level 1 on all requests.					

Converting between EBCDIC and ASCII data

Of the systems to which your region can connect, some store data in different character encodings. An example of this is that data that is held in the Extended Binary-Coded Decimal Interchange Code (EBCDIC) format on an IBM mainframe-based CICS system, and in the American National Standard Code for Information Interchange (ASCII) format on a TXSeries for Multiplatforms system. Differences exist also between data in CICS on Windows systems and data in CICS on Open Systems, where the codes that are used for some characters vary and the method that is used for representing numbers is different. This means that if data that is transferred between two systems is to be useful, it must be converted from the format that is used on the sending system to that which is used by the receiving system.

CICS converts some data, such as the file names in function shipping requests, without user setup. For other data, such as file records, users supply resource definitions that identify the types of conversion that are to be applied to specified fields in data records. Exits and user-replaceable conversion programs are also available.

Chapter 7, "Data conversion," on page 147 explains how to configure data conversion when TXSeries for Multiplatforms is doing the conversion. The following references provide further information:

- When the data is shipped from TXSeries for Multiplatforms to IBM mainframe-based CICS and the data conversion takes place in IBM mainframe-based CICS, you also need *CICS Family: API Structure*.
- When the data is shipped from TXSeries for Multiplatforms to CICS OS/2 and the data conversion takes place on CICS OS/2, you need *CICS for OS/2® Intercommunication Guide*.
- When the data is shipped from TXSeries for Multiplatforms to CICS/400 and the data conversion takes place on CICS/400, you need *Communicating from CICS/400*.

Data conversion is described in detail in Chapter 7, "Data conversion," on page 147.

Performance issues for intercommunication

Sending data through a network takes time, so accessing resources on a remote system takes longer than accessing resources on a local system. However, the advantages of not having to replicate data on each system or requiring users to be signed on to several systems can compensate for the increased response time.

You can minimize the performance impact of distributing a resource by choosing the intercommunication facility carefully.

Choosing which intercommunication facility to use

The intercommunication facilities that CICS supports are:
- Function shipping
- Transaction routing
- Asynchronous processing
- Distributed program link (DPL)
- Distributed transaction processing (DTP)

These facilities complement one another. This means that for each application, one of the facilities is probably more efficient and easier to use than the others are. Use the following descriptions to help you choose the facility that will give you the best performance.

It is important to consider the level of support that is provided for these facilities on the remote system. You can find information about the intercommunication facilities that are supported by each of the CICS products in *CICS Family: Interproduct Communication* .

When to use function shipping

Function shipping allows an application to read from, and write to, files, temporary storage queues, and transient data queues that are on remote CICS systems. The application issues the standard EXEC CICS commands to access these data resources, and CICS packages the request and sends it to the required CICS system. This system then unpackages and executes the request as if it were issued by one of its local programs. The result of the request is sent back to the originating system and is returned to your application, again as if the resource were local.

Function shipping provides an easy way to access remote CICS data resources. However, as each request is sent separately to the remote system, it can be better to use distributed program link (DPL) or distributed transaction processing (DTP) if the application requires many accesses to the remote data. Also, function shipping is available only for accessing CICS resources, such as files and queues. If you want to access non-CICS data, such as database records, on a remote system, you must use another method.

When to use transaction routing

Transaction routing allows a user of your region to run a transaction on a remote CICS system and see the results of that transaction on a local terminal as if the transaction were running in your region. This is possible because the remote CICS system can package up all the screen displays that are generated by the routed transaction, and send them to your region, where they are unpackaged and displayed as normal on the terminal screen. Transaction routing is therefore useful if all the processing for a transaction is to take place on a single remote system.

When to use asynchronous processing

Asynchronous processing allows an application to start a transaction on a remote CICS system at a particular time. The success or failure of this remotely started transaction does not affect the application that started it. Therefore, it is useful for conditions in which it is not necessary, or desirable, to keep a local transaction waiting while the remote transaction is running.

When to use distributed program link (DPL)

Distributed program link allows an application to issue an EXEC CICS LINK to a program that resides on another CICS system. Your application passes a COMMAREA to the linked-to program that can contain instructions on the processing required. The linked-to program then executes and returns the results, again in a COMMAREA, to the original application. DPL is extremely useful if an application needs to make a large number of accesses to a remote resource, for example, to search through a file on a remote CICS system to find a particular data record, which is then displayed to the user. The local application can use DPL to connect to a program on the system that owns the data, passing details of which record is required. The linked-to program can search through the file and return the required record to the local application. Using DPL in this case is significantly more efficient that using function shipping to read each record remotely. DPL has the restriction that the COMMAREA cannot be more than 32700 bytes long. If your transaction needs to transfer large amounts of data between two CICS regions, consider using DTP.

When to use distributed transaction processing (DTP)

Distributed transaction processing is the most flexible of all the intercommunication facilities. It allows an application to send as much data as it needs, between one or more systems, at any point in its processing. However, this means it is the most complex to use because the application is responsible for setting up the communications with the remote system, sending and receiving the data, and finally closing the communications down when it is finished. It is the only intercommunication facility that is available to your application if it requires communication with a non-CICS system.

A checklist of application requirements

This checklist describes possible application requirements and summarizes which intercommunication facility is likely to be the most efficient:

- A terminal user wants to run a transaction on a remote CICS system. Use transaction routing.
- A transaction needs to read and write data that is all on a remote CICS system. Function shipping seems an obvious choice here, but it does have a higher overhead than transaction routing does. If no processing is required on the local system, transaction routing is more efficient. If some processing is required locally, you might consider distributed program link (DPL) or distributed transaction processing (DTP). As a general rule, it is often best to process data as close to its source as possible because this is likely to reduce the amount of data that is sent across the network.
- A transaction needs to read and write a small amount of data from several CICS systems. Function shipping is a good choice here. It is easy to use and it is unlikely that much scope exists for pruning the amount of data that is sent across the network by having a transaction run in each system that owns a particular piece of data.
- A transaction needs to read or write data that is stored on a remote database that is accessible by another CICS system. Use distributed program link (DPL) or distributed transaction processing (DTP) if function shipping is not supported for these data resources.
- A transaction needs to start one or more transactions in a remote CICS system and does not need to wait for the results. Use asynchronous processing.
- An application needs to search through a file on a remote CICS system and retrieve particular pieces of information. Use distributed program link (DPL).
- An application needs to transfer a large amount of data from one system to another. Use distributed transaction processing (DTP).
- An application needs to coordinate a number of related updates on several systems. Use distributed transaction processing (DTP).

- An application needs to communicate with a non-CICS system. Use distributed transaction processing (DTP).

For more information, see Part 4, “Writing application programs for intercommunication,” on page 245.

Operational issues for intercommunication

Below is a list of some of the issues that can affect you. It is worth investing some time on these issues to prevent problems when your system is in production.

- **Backup of remote data**: Ensure that site of the remote data that is used by the users of your region is backed up frequently enough and that disaster recovery procedures are in place to restore the data in the event of a failure.
- **Problem determination support**: Ensure you have a contact at the remote system site to help you track down problems with distributed applications. You need to arrange for the remote system to save transaction dumps and other problem determination aids for failed applications that are started by intersystem requests from your region.
- **System availability**: Ensure that the remote system will be available when your users require it.
- **Network monitoring**: Ensure that a clear understanding exists of who is responsible for monitoring the throughput and availability of the network.
- **Compensation for resources used**: If the remote systems that will be connecting to your region are owned by a different organization, some negotiation can be required to agree on a charging scheme for the resources that are used by remote users at each site on a day-to-day basis.

For more information, see “Performance issues for intercommunication” on page 19.

Bidirectional input and display on AIX

CICS is enabled for bidirectional input and display on local CICS terminals. Customers who are using data fields that include information expressed in Hebrew, Arabic, or other right-to-left languages can enter and display information as they normally do.

Part 2. Configuring for intercommunication

Before two systems can communicate, they need to know something about the identity and characteristics of each other, and they need information about the method that they will use to communicate. They also need to understand the resources that they will share, and the security checks that they will impose. This part describes these configuration requirements:

Table 7. Road map

If you want to...	Refer to...
Read about configuring CICS to support TCP/IP	Chapter 3, "Configuring CICS for TCP/IP," on page 25
Read about configuring CICS to support SNA	Chapter 4, "Configuring CICS for SNA," on page 67
Read about configuring the CICS resources that will be shared with other systems	Chapter 5, "Configuring resources for intercommunication," on page 109
Read about configuring CICS to enable only authorized systems and users to gain access its resources	Chapter 6, "Configuring intersystem security," on page 123
Read about configuring CICS to support data conversion when other systems represent data in a different format	Chapter 7, "Data conversion," on page 147

Chapter 3. Configuring CICS for TCP/IP

TXSeries for Multiplatforms offers you a choice of two ways of using TCP/IP protocols:

- **CICS family TCP/IP** support, which allows connectivity to other TXSeries for Multiplatforms, CICS OS/2 regions, and IBM CICS Universal Clients
- **CICS PPC TCP/IP** support, which allows connectivity to other TXSeries for Multiplatforms regions

For introductory information on these two ways of using TCP/IP, refer to "Using CICS family TCP/IP" on page 6, and "Using CICS PPC TCP/IP" on page 7.

To configure CICS for TCP/IP, perform the following steps:

For CICS family TCP/IP connections:

1. Identify a local name for your region.
2. Set up a Listener Definitions (LD) entry for your region.
3. Set up a Communications Definitions (CD) entry for each remote system with which your region is to communicate.

For CICS PPC TCP/IP connections:

1. Identify a local name for your region.
2. Set up a Communications Definitions (CD) entry for each remote system with which your region is to communicate.

The information that follows describes how each of these steps is accomplished. The references in the road map tables guide you from one step to the next.

When using CICS on AIX or Windows systems:

CICS commands are used to configure the CICS LD and CD entries. However, if you are using CICS on an AIX system, you can also use the System Management Interface Tool (SMIT) to configure these resources. If you are using CICS on a Windows system, you can use the IBM TXSeries Administration Tool.

Naming your region for a TCP/IP intercommunication environment

In an intercommunication environment, remote systems need to know the name of your CICS region in order to:

- Send your region intersystem requests
- Apply security checks and data conversion to intersystem requests that are received from your region

The name by which remote systems identify your region depends upon the communication method. The one- through eight-character name that you specify when you create your region is referred to as the *region name*, or the *APPLID*.

A remote TXSeries for Multiplatforms region or a CICS OS/2 Version 3 (or later) region that is communicating over CICS family TCP/IP also knows your region by its APPLID. However, a Version 2 CICS OS/2 region builds a name for your region based on its network adapter address and port number. For example, if your region is using the TCP/IP host name `cicsopen.cicsland.com` or `aix5.cicsland.com` and port number `1435`, your region would be known as “05G2OXPN”. This name is returned by the **cicstcpnetname** command, as follows:

```
cicstcpnetname -a aix5.cicsland.com -p 1435
05G2OXPN
```

For further information about the **cicstcpnetname** command, refer to the *TXSeries for Multiplatforms Administration Reference*.

Your CICS region also has a name that is available to local applications and resource definitions. CICS applications and resource definitions use a one- through four-character name called the *SYSID* to specify the name of the system where a resource resides. If the resource is remote, the SYSID is the one- through four-character name of a Communications Definitions (CD) entry. If the resource is local, the application or resource definition can either not specify a SYSID, or use the local SYSID. The local SYSID is defined in the **LocalSysId** attribute of your region’s Region Definitions (RD) entry. The default value is ISC0, and you can change this value when you create your region or restart it. Whatever value your region uses, ensure that it is different from the names of all CD entries and Terminal Definitions (WD) entries that are used by your CICS region’s local transactions.

When using CICS for AIX: If you are using SMIT to configure the RD entry, the SMIT field name for the **LocalSysId** command attribute is **Region system identifier (short name)**.

Table 8. Where to next

If you want to...	Refer to...
Use CICS family TCP/IP support	“Configuring CICS for CICS family TCP/IP support”
Use PPC TCP/IP support	“Configuring CICS for CICS PPC TCP/IP support” on page 47

Configuring CICS for CICS family TCP/IP support

CICS family TCP/IP support is enabled in your region by configuring:

- A Listener Definitions (LD) entry with **Protocol=TCP**

  A TCP LD entry is required for each TCP/IP port that CICS regions and IBM CICS Universal Clients can use to contact the region. While the maximum number of simultaneous connections through a single port is virtually unlimited, the system configuration must be adjusted to deal with the load effectively. As a general rule, a single listener process should not be handling more than about 500 connections (and even this might be restricted by file descriptor limits or thread limits that are imposed by operating system configuration). Above this number of connections for a single port, the TCPProcessCount attribute should be used to adjust the number of operating system processes that CICS uses to listen for connections. Determining the optimum number of connections that are simultaneously active to a single listener is dependent on workload profiles, and cannot be exactly predicted, but the above general rule should help. Configuring a LD entry is described in “Configuring LD entries for CICS family TCP/IP” on page 27.

- A Communications Definitions (CD) entry for each remote system with **ConnectionType=ppc_tcp**

  The CD entries can be defined to the region by using the standard CICS RDO commands such as **cicsadd**, or the IBM TXSeries Administration Tool (in CICS for Windows), or they can be autoinstalled when a remote system acquires the connection. The RDO technique is described in "Configuring CD entries for CICS family TCP/IP(CICS on Open Systems only)" on page 34, and the autoinstall method is described in "Configuring for autoinstallation of CD entries" on page 55.

Examples of CICS family TCP/IP connections are shown in "CICS family TCP/IP Configuration examples" on page 43.

Configuring LD entries for CICS family TCP/IP

This section describes how to configure LD entries for your system. For CICS on Open Systems, see "Configuring LD entries for CICS family TCP/IP (CICS on Open Systems)"; for CICS for Windows, see "Configuring LD entries for CICS family TCP/IP (CICS for Windows only)" on page 28.

Configuring LD entries for CICS family TCP/IP (CICS on Open Systems)

CICS family TCP/IP support requires at least one Listener Definitions (LD) entry with **Protocol=TCP**. The **TCPAddress** and **TCPService** attributes are used to define the network adapters and port number on which CICS is to accept connection requests.

TCPAddress defines the network adapter addresses. It can be specified in the following ways:

- The Internet Protocol (IP) address in *dotted decimal* notation. For example, `1.23.45.67`. Do not use leading zeros when specifying an address in dotted decimal notation. CICS interprets such an entry as octal.
- The Internet Protocol (IP) address in *dotted hexadecimal* notation. For example, `0x01.0x17.0x2D.0x43`.
- The host name defined in the Internet name service. For example, `aix5.cicsland.com`.

The **TCPService** attribute specifies a TCP/IP service name. This service name is configured in the TCP/IP configuration file called **/etc/services** and defines a TCP/IP port number. How to add entries to **/etc/services** is described in "Adding a service name to the TCP/IP configuration" on page 30.

The following example shows **TCPAddress** and **TCPService** in a CICS family TCP/IP LD entry called CICSTCP. Notice also that **ActivateOnStartup=yes** as CICS will activate listeners only during region start up. Any LD entries that are added while the region is running will be used by the CICS region only when the region is restarted.

```
CICSTCP:   ActivateOnStartup=yes
                   Protocol=TCP
                   TCPAddress="aix5.cicsland.com"
                   TCPService="cicstcp"
```

The command to add LD entry CICSTCP to the CICS region's permanent database is:

```
cicsadd -r cics -P -c ld CICSTCP TCPAddress ="aix2.cicsland.com"
        TCPService="cicstcp"
```

The **ActivateOnStartup** and **Protocol** attributes are not specified because the default values are being used. (The command **cicsget -r** *regionName* **-c ld ""** will display the default attributes for the LD entries).

Alternatively, the **TCPAddress** and **TCPService** attributes can be left blank. A blank **TCPAddress** indicates that CICS can use the address of any of the TCP/IP network adapters on the machine. A blank **TCPService** attribute means that CICS will listen for connections on port 1435.

```
  CICSTCP:   ActivateOnStartup=yes
                   Protocol=TCP
                   TCPAddress=""
                   TCPService=""
```

CICS does not need an entry in /etc/services if TCPService="". However, it is recommended that you add an entry for the 1435 port to document that your region is using it.

Note: If you have multiple network adapters, and you configure a LD entry with **TCPAddress=""**), that listener cannot be used for CICS family TCP/IP connections with other CICS server regions. However, such a listener can be used to support CICS family TCP/IP connections from the CICS server to IBM CICS Universal Clients. If you want to use CICS family TCP/IP support between CICS regions, you must ensure that your LD entry defines only one network adapter address.

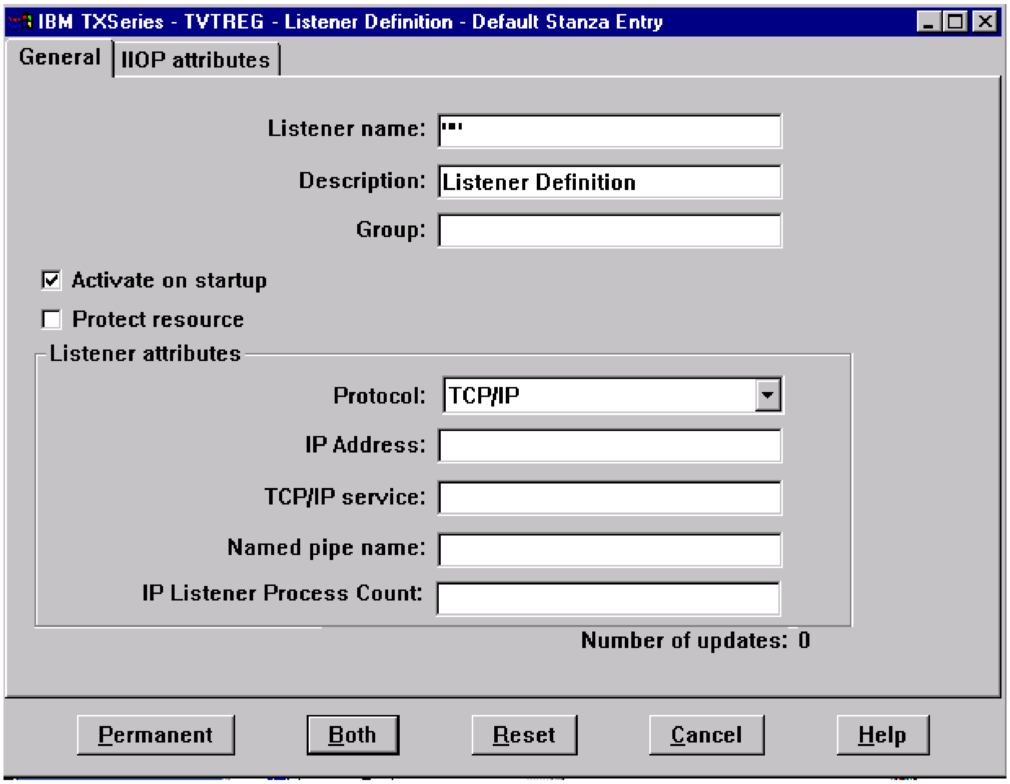
Configuring LD entries for CICS family TCP/IP (CICS for Windows only)

A Listener Definitions (LD) entry describes how your CICS region should connect to a network. For CICS family TCP/IP support, a LD entry is required for each TCP/IP port that will be used to receive requests.

To define a LD entry for CICS family TCP/IP, you select your region name from the main panel of the IBM TXSeries Administration Tool. Then select the **Listener** resource from the **Resources** option of the **Subsystem** menu.

This action displays the list of LD entries that you have already defined.

Select the **New** option of the **Listeners** menu, and the properties notebook window is displayed, as shown in Figure 10 on page 29.


IBM TXSeries - TVTREG - Listener Definition - Default Stanza Entry
General
IIOP attributes

Listener name: ""

Description: Listener Definition

Group:

☑ Activate on startup

☐ Protect resource

Listener attributes

Protocol: TCP/IP

IP Address:

TCP/IP service:

Named pipe name:

IP Listener Process Count:

Number of updates: 0

Permanent
Both
Reset
Cancel
Help

Figure 10. Listener Definitions (LD) entry

Enter a name for the LD entry in **Listener name**. This can be up to eight characters in length, and is a local name that your CICS region uses to identify the LD entry in CICS messages. It must be different from the names of all other LD entries that are in your region. However, it does not have to be unique within the network and does not have to relate to any other names that are used in the network.

The **Protocol** should be TCP/IP.

The **IP address** is the TCP/IP address of the machine on which your CICS region is running. It can be expressed in any of the following ways:

- The Internet Protocol (IP) address in *dotted decimal* notation. For example, `1.23.45.789`. Do not use leading zeros when specifying an address in dotted decimal notation. CICS interprets such an entry as octal.
- The Internet Protocol (IP) address in **dotted hexadecimal** notation. For example, `0x01.0x17.0x2D.0x315`.
- The host name defined in the Internet name service. For example, `cicsopen.cicsland.com`. If a host name is used, it ***must*** map to only one IP address.

Alternatively, the IP address can be left blank to indicate that CICS can use any of the TCP/IP network adapters that are on the machine.

The **TCP/IP service** attribute specifies a TCP/IP service name that, in turn, defines a TCP/IP port number. Adding service entries is described in “Adding a service name to the TCP/IP configuration” on page 30.

The default value for the **TCP/IP service** attribute is blank, which requests that CICS uses the **1435** port that is the port assigned to CICS by the Internet Assigned Number Authority (IANA).

When you have filled in all the attributes, select either **Permanent** or **Both** to create the LD entry.

If **Permanent** is selected, the CICS command that the IBM TXSeries Administration Tool will issue for the example shown above is:

```
C:\ cicsadd -r cicswint -P -c ld CICSTCP               \
                           Protocol=TCP                                \
                           TCPAddress="wint127.cicsland.com" \
                           TCPService="cicstcp"
```

The **cicsadd** command is passed the name of each attribute with its required value. This attribute name can be displayed by placing the mouse pointer over the description of the attribute on the IBM TXSeries Administration Tool page.

You must restart your region in order to use this listener. You should perform a cold start if you selected **Permanent**, or auto start if you selected **Both**.

Refer to the *TXSeries for Multiplatforms Administration Reference* for a description of the **cicsadd** command.

Adding a service name to the TCP/IP configuration

The service name that is specified in the CICS family TCP/IP Listener Definitions (LD) entry is defined in the TCP/IP configuration file /etc/services. The example below shows three service name entries from /etc/services. Each entry begins with a service name (cicstcp1). This is followed by the TCP/IP port number (8595) and /tcp. You can also add a comment to the end of the line. This must begin with a hash character (#).

```
cicstcp1    8595/tcp        # TCP listener1 for region regionName
cicstcp2    8596/tcp        # TCP listener2 for region regionName
cicstcp3    8597/tcp        # TCP listener3 for region regionName
```

Note: The allocation of port number to services is not enforced. System administrators can set up the /etc/services file as they choose, including using port 1435 for a service other than CICS. Therefore always check whether you have chosen a port number that is unique to your machine, and that is not being used by any other CICS region, listener, or TCP application. If your region is using the CICS default port number 1435, it is recommended that you add an entry to /etc/services for this port number to document that it is in use. This might prevent the systems administrator from configuring another application or CICS region with this port. For example:

```
cicstcp      1435/tcp        # TCP Listener used by region regionName
```

Increasing the size of the mbuf pool (AIX only)

If your local machine is managing many TCP/IP connections, it might run out of space in the *mbuf pool*. This is an area of memory that TCP/IP allocates when AIX is initialized. The default size is 2 MB, which will manage up to about 800 CICS family TCP/IP connections.

You can display size of the mbuf pool, along with other network options, by using the **no -a** command. The mbuf pool size is called *thewall*. It is expressed in multiples of 1024 bytes, so in the example below, the mbuf pool size is 2048 x 1024 bytes (2 MB).

```
% no -a | grep thewall
                thewall = 2048
```

The **no -o thewall** command allows you to change the mbuf pool size. The example below sets the mbuf pool size to 4096 x 1024 bytes (4 MB). (You will need to be the **root** user to issue this command.)

```
% no -o thewall 4096
```

The **no** command operates only on the current running kernel, so it must be rerun each time your machine is started up. To configure your machine to run this command automatically on startup, add the command to the */etc/rc.net* file.

Attention: Be careful when you use the **no** command because it performs no range checking on the parameters that you specify. If used incorrectly, the **no** command can cause your system to become inoperable.

Using SMIT to configure LD entries (AIX only)

When using CICS for AIX, you can use SMIT to configure the Listener Definitions (LD) entries. To support CICS family TCP/IP, you must configure an LD entry with the **Protocol type** field set to **TCP**. You will need an LD entry for each TCP/IP port that CICS regions and IBM CICS Universal Clients use to contact the region. The optimum number of connections simultaneously active through a single port is about 200 and the maximum is around 450 connections.

To add an LD entry for CICS family TCP/IP to your region:

- Ensure that you are logged on to AIX with enough privileges to change the region database. (For example, log onto AIX as the **root** user.)
- Optionally set the environment variable **CICSREGION** to the name of your CICS region. For example:

  ```
  % export CICSREGION=cics
  ```
- Enter `smitty cicsregion` to start SMIT.
- Use option **Change Working CICS Region** to select your CICS region. (This is required only if you have not set up **CICSREGION** before starting SMIT.)
- Select options:

  ```
  ► Define Resources for a CICS Region
      ► Manage Resource(s)
          ► Listeners
              ► Add New
  ```

  This displays the **Add Listener** panel. Enter a model Listener Definition (LD) entry name. This could be the name of an LD entry that you have defined already. Alternatively, press the Enter key to use the default.

  Figure 11 on page 33 shows an example of a SMIT Add Listener panel.
- Type in the name of the LD entry in the **Listener Identifier** field, and the values that you require for **TCP adapter address** and **TCP service name**.
  - The **TCP adapter address** field defines the network adapter addresses of your machine. It can be specified in the following ways:

    - The Internet Protocol (IP) address in dotted decimal notation. For example, `1.23.45.67`. Do not use leading zeros when specifying an address in dotted decimal notation. CICS interprets such an entry as octal.
    - The Internet Protocol (IP) address in dotted hexadecimal notation. For example, `0x01.0x17.0x2D.0x43`.
    - The host name that is defined in the Internet name service. For example, `aix5.cicsland.com`.
  - The **TCP service name** field specifies the service name that CICS uses when starting TCP/IP. This service name is configured in the TCP/IP configuration file called **/etc/services** and defines a TCP/IP port number.

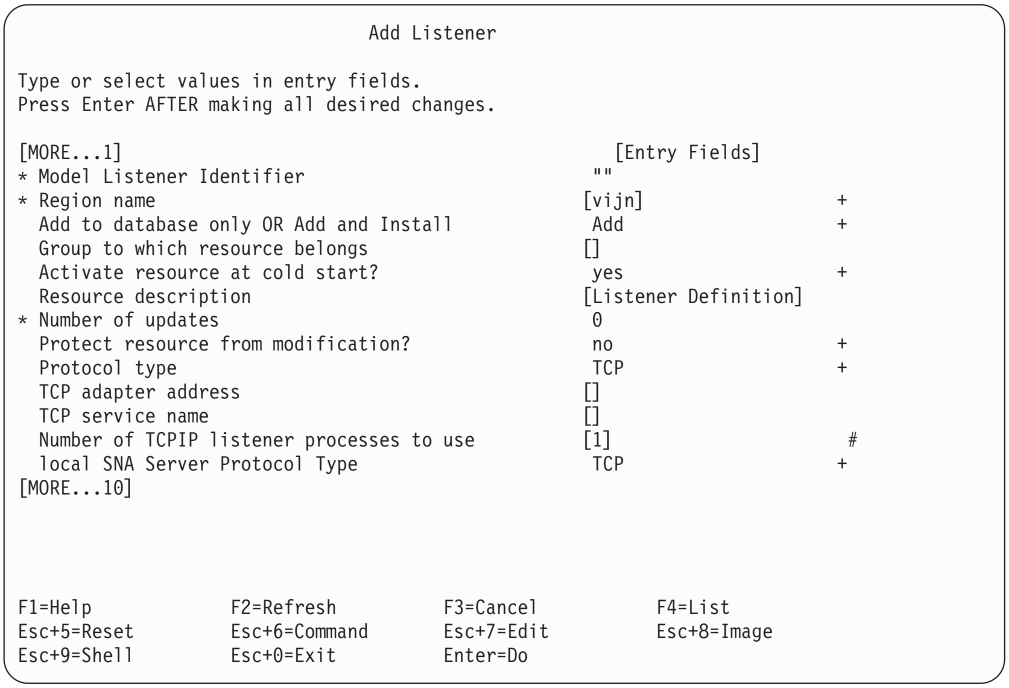
    This is described further in "Adding a service name to the TCP/IP configuration" on page 30.

  The **TCP adapter address** and **TCP service name** attributes can remain blank. A blank **TCP adapter address** indicates that CICS can use the address of any of the TCP/IP network adapters that are on the machine. A blank **TCP service name** attribute means that CICS will listen for connections on port **1435**.

  CICS does not need an entry in **/etc/services** if **TCP service name=""**. However, it is recommended that you add an entry for the 1435 port to document that your region is using it.

  Note: If you have multiple network adapters, and you configure a LD entry with **TCP adapter address=""**), that listener cannot be used for CICS family TCP/IP connections with other CICS server regions. However, such a listener can be used to support CICS family TCP/IP connections from the CICS server to IBM CICS Universal Clients. If you want to use CICS family TCP/IP support between CICS regions, you must ensure that your LD entry defines only one network adapter address.
- Then press the Enter key to create the LD entry.

Figure 11 on page 33 shows an example of the SMIT panel for adding a new LD entry.



```
                               Add Listener

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[MORE...1]                                            [Entry Fields]
* Model Listener Identifier                          ""
* Region name                                       [vijn]                +
  Add to database only OR Add and Install            Add                  +
  Group to which resource belongs                   []
  Activate resource at cold start?                   yes                  +
  Resource description                              [Listener Definition]
* Number of updates                                  0
  Protect resource from modification?                no                   +
  Protocol type                                      TCP                  +
  TCP adapter address                               []
  TCP service name                                  []
  Number of TCPIP listener processes to use         [1]                    #
  local SNA Server Protocol Type                     TCP                  +
[MORE...10]




F1=Help            F2=Refresh         F3=Cancel          F4=List
Esc+5=Reset        Esc+6=Command      Esc+7=Edit         Esc+8=Image
Esc+9=Shell        Esc+0=Exit         Enter=Do
```

Figure 11. SMIT panel for adding an LD entry for CICS family TCP/IP support

Tuning TCP/IP on the Windows platform

The default settings for the KeepAliveTime and TcpTimedWaitDelay registry settings can cause delays if a network failure occurs. This section discusses what you must consider for these registry settings when you are connecting Windows Client terminals (**cicslterm.exe**) that are hard-named to a Windows region over TCP/IP. A terminal or a connection is hard-named when you give it a specific name in the DFHCCINX (for connections), or in the DFHCHATX (for terminals).

TCP/IP's KeepAliveTime registry setting

The KeepAliveTime value sets the time that can elapse without a communication from an endpoint connection before the system checks whether the endpoint connection is still active. The default value is 7,200,000 milliseconds (2 hours). If a network failure occurs while this registry setting is set to this default setting, two hours can elapse before the failed endpoint connection is cleared and able to be reused.

You can lower the value for the KeepAliveTime registry setting, which increases network activity on idle connections, but the activity on active connections is not affected. However, if the KeepAliveTime value is set too low, active connections can terminate incorrectly because of latency on the network. The appropriate value for the KeepAliveTime registry setting differs for each installation. Some installations run efficiently by using a value as low as five seconds.

You can change the KeepAliveTime registry setting in the following Windows registry location:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
```

Microsoft SNA Server's KeepAlive registry setting: If you are using the Microsoft SNA Server over encapsulated IP and you have applied the

KeepAliveTime registry setting change in the Windows registry, you possibly need to configure the Microsoft SNA Server not to use TCP/IP's KeepAlive facility.

For example, suppose that the implementation has a Local Area Network (LAN) with IBM Universal Clients connecting to a local server that is using SNA Server to connect to a mainframe. In this configuration, if the value for TCP's KeepAliveTime registry setting is set low (to detect terminal disconnections quickly), configure the SNA Server not to use TCP/IP's KeepAlive facility. However, if the value of TCP's KeepAliveTime registry setting is set high enough to allow the external link to be correctly registered, the SNA Server can be configured to use the SNA KeepAlive registry setting.

To configure the Microsoft SNA Server not to use TCP/IP's KeepAlive facility, set the value for SNA's KeepAlive registry setting to **NO** in the Windows Registry.

You can change the registry settings for the Microsoft SNA Server configuration in the following Windows registry location:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\SnaBase\Parameters\SnaTcp
```

TcpTimedWaitDelay settings

The Microsoft Windows implementation of TCP/IP uses a default value for the TcpTimedWaitDelay registry setting of 240 seconds (four minutes). If a network failure occurs, it is possible that CICS will not detect the failure of an active session (as opposed to an idle session) for four minutes. The TcpTimedWaitDelay setting is by Microsoft SNA Server with encapsulated IP.

The value for the TcpTimedWaitDelay is derived from the Maximum Segment Length (MSL), which is the maximum time that an IP packet can exist. The setting for the TcpTimedWaitDelay registry setting must be twice the value of the MSL to allow enough time for the final packet to be delivered and the response to be received. For example, if a TCP/IP implementation has an average of 30 seconds for an MSL, set the TcpTimedWaitDelay value at 1 minute to allow enough time for packet delivery and response.

Attention: Reducing the value of the TcpTimedWaitDelay registry setting on a congested network can result in spurious failures. Before making any changes to this registry setting, ensure that the packet delivery times for your implementation are stable.

You can change the TcpTimedWaitDelay registry setting in the following Windows registry location:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
```

Configuring CD entries for CICS family TCP/IP

This section describes how to configure CD entries for your system. For CICS on Open Systems, see “Configuring CD entries for CICS family TCP/IP(CICS on Open Systems only)”; for CICS for Windows, see “Configuring Communications Definitions (CD) entries for CICS family TCP/IP (CICS for Windows only)” on page 36.

Configuring CD entries for CICS family TCP/IP(CICS on Open Systems only)

To define a CICS family TCP/IP connection in your region, configure a Communications Definitions (CD) entry that has **ConnectionType=ppc_tcp**.

To configure the location of the remote system in a CICS family TCP/IP CD entry, use the **RemoteTCPAddress** and **RemoteTCPPort** attributes.

RemoteTCPAddress specifies the machine, or more specifically the network adapter card on the machine, on which the remote system is running. It can be specified in the following ways:

- The Internet Protocol (IP) address in **dotted decimal** notation. For example, `1.23.45.67`. Do not use leading zeros when specifying an address in dotted decimal notation. CICS interprets such an entry as octal.
- The Internet Protocol (IP) address in **dotted hexadecimal** notation. For example, `0x01.0x17.0x2D.0x43`.
- The host name that is defined in the Internet name service. For example, `aix5.cicsland.com`. If a host name is used, it **must** map to only one IP address. You can check this by using the **host** command. For example:

```
$ host aix5.cicsland.com
aix5.cicsland.com is 1.23.45.67
```

The **RemoteTCPPort** attribute must be set to the port number that the remote system is using to listen for connection requests. The default value for **RemoteTCPPort** is **1435**, which is the port that is assigned to CICS by the Internet Assigned Number Authority (IANA).

The **ListenerName** attribute must be set to the name of a Listener Definitions (LD) entry that is defined in the local region that has **Protocol=TCP**. This LD entry defines the IP address and port number that the remote system must use to contact the local region. CICS requires the **ListenerName** attribute to be correctly configured, even if this connection is to be used only for outbound requests, because a **cicsip** process, which is started as a result of the LD entry, is required to open the TCP/IP connection. ("Configuring LD entries for CICS family TCP/IP" on page 27 describes how to set up an LD entry.)

The **RemoteLUName** attribute should be set to the remote region name (that is, the APPLID), unless the remote system is a version 2 CICS OS/2 or version 2 CICS for Windows system, when it should be the netname that is returned by the **cicstcpnetname** command:

```
cicstcpnetname -a <RemoteTCPAddress> -p <RemoteTCPPort>
```

The subject of how regions are named is discussed further in "Naming your region for a TCP/IP intercommunication environment" on page 25. For further information about the **cicstcpnetname** command, see the *TXSeries for Multiplatforms Administration Reference*.

The following attributes are not required for a CICS family TCP/IP connection and can remain as the default value:

- **RemoteNetworkName**
- **SNAConnectName**
- **GatewayName**
- **AllocateTimeout**
- **RemoteSysEncrypt**

Refer to "Data conversion for transaction routing" on page 164 for information about how to configure the **RemoteCodePageTR** attribute.

The following example shows a command to add a CD entry called TOSF to the CICS region's permanent database. Only some of the attributes are specified. The attributes that are not specified are set to their default values. You can view the default values for a CD entry for your region by using **cicsget -r** *regionName* **-c cd ""**.

```
cicsadd -r cics -P -c cd TOSF   \
                       ConnectionType=ppc_tcp  \
                       RemoteLUName="cicsosf1"   \
                       RemoteTCPAddress="digital.cicsland.com" \
                      RemoteTCPPort=1435    \
                       ListenerName="CICSTCP"  \
                       LinkUserId="LINKTOSF"
```

Refer to Chapter 6, "Configuring intersystem security," on page 123 for information about how to configure the following security attributes:

- **OutboundUserIds**
- **RemoteSysSecurity**
- **LinkUserId**
- **TSLKeyMask**
- **RSLKeyMask**

Configuring Communications Definitions (CD) entries for CICS family TCP/IP (CICS for Windows only)

A Communications Definitions (CD) entry describes details of a remote system and how your local region should communicate with it. To define a CD entry for a remote system, select your region name from the main panel of the IBM TXSeries Administration Tool. Then select the **Communication** resource from the **Resources** option of the **Subsystem** menu.

This displays the list of CD entries that you have already defined.

Select the **New** option of the **Communications** menu. The properties notebook window is displayed.

The attributes for a CD entry are grouped into four pages:

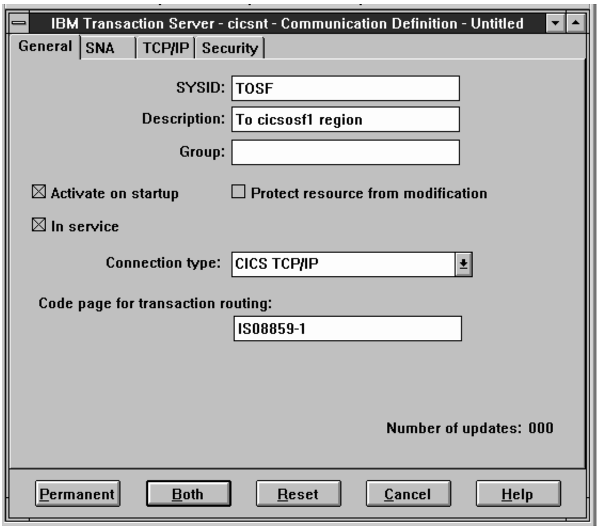
General
Attributes that are required by all CD entries.

SNA Attributes that describe how the remote system communicates over a SNA network. Attributes on this page are not applicable to a CICS family TCP/IP CD entry.

TCP/IP
Attributes that describe how the remote system connected to the TCP/IP network.

Security
Attributes that define how security will be managed.

The **General** page is shown in Figure 12 on page 37.


IBM Transaction Server - cicsnt - Communication Definition - Untitled
General
SNA
TCP/IP
Security
Permanent
Both
Reset
Cancel
Help

SYSID: TOSF

Description: To cicsosf1 region

Group:

☒ Activate on startup

☐ Protect resource from modification

☒ In service

Connection type: CICS TCP/IP

Code page for transaction routing: IS08859-1

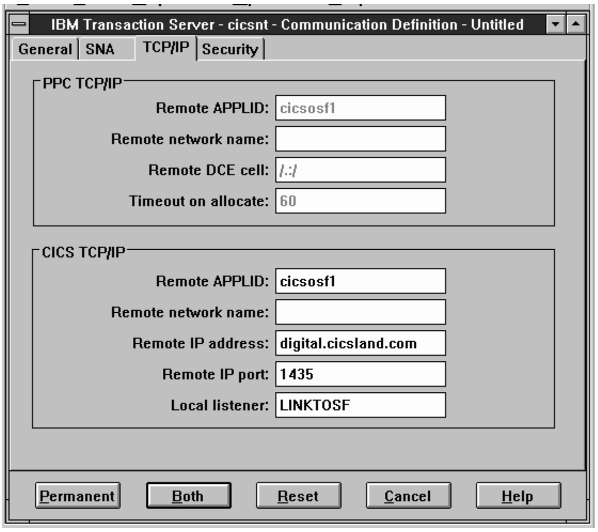
Number of updates: 000

Figure 12. General Communications Definitions (CD) panel

Enter the four-character SYSID for this CD entry. This is the CD entry's key and must be different from that which is used by other CD entries in your region. However, the name is used only by local resources and applications. It does not have to be unique within the network, and it does not relate to any other values in the remote system.

The **Connection type** should be changed to **CICS TCP/IP**. The **Code page for transaction routing** should be the code page that is used to flow transaction routing data across the network. The correct code page depends on the national language of your local region and the type of the remote system. Chapter 7, "Data conversion," on page 147 describes how to determine the value.

Select the **TCP/IP** tab to display the TCP/IP attributes, which is shown in Figure 13. This panel is used to describe the remote CICS region and its location in the TCP/IP network.


IBM Transaction Server - cicsnt - Communication Definition - Untitled
General
SNA
TCP/IP
Security
Permanent
Both
Reset
Cancel
Help

PPC TCP/IP

Remote APPLID: cicsosf1

Remote network name:

Remote DCE cell: /.:/

Timeout on allocate: 60

CICS TCP/IP

Remote APPLID: cicsosf1

Remote network name:

Remote IP address: digital.cicsland.com

Remote IP port: 1435

Local listener: LINKTOSF

Figure 13. TCP/IP Communications Definitions (CD) panel

For most CD entries the **Remote APPLID** is the name of the remote CICS region. The exceptions are when the remote system is either:

- CICS OS/2 version 2
- CICS for Windows

For these systems the APPLID must be generated by using the **cicstcpnetname** utility. This is described in the *TXSeries for Multiplatforms Administration Reference*.

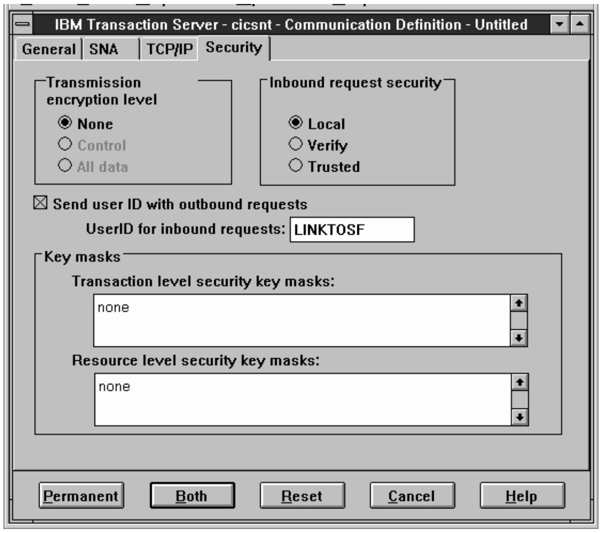
The **Remote IP address** is the TCP/IP address of the machine on which the remote CICS region is running. It can be expressed in one of the following ways:

- The Internet Protocol (IP) address in **dotted decimal** notation. For example, `1.23.45.67`. Do not use leading zeros when specifying an address in dotted decimal notation. CICS interprets such an entry as octal.
- The Internet Protocol (IP) address in **dotted hexadecimal** notation. For example, `0x01.0x17.0x2D.0x43`.
- The host name that is defined in the Internet name service. For example, `aix5.cicsland.com`. If a host name is used, it **must** map to only one IP address.

The **Remote IP port** attribute must be set to the port number that the remote system will be using to listen for TCP/IP connection requests. The default value is **1435**, which is the port that is assigned to CICS by the Internet Assigned Number Authority (IANA).

The **Local Listener** attribute must be set to the name of a TCP Listener Definitions (LD) entry that is defined in the local region. This LD entry defines the IP address and port number that the remote system must use to contact the local region. CICS requires the **Local Listener** attribute to be correctly configured even if this CD entry is only to be used for outbound requests, because the **cicsip** process, which is started as a result of the LD entry, is required to open the TCP/IP connection. Further information about how to configure the LD is given in “Configuring LD entries for CICS family TCP/IP” on page 27.

Select the **Security** tab to display the security attributes, which are shown in Figure 14. These attributes describe the security checking that applies to all intersystem requests that use this CD entry.


IBM Transaction Server - cicsnt - Communication Definition - Untitled
General
SNA
TCP/IP
Security
Permanent
Both
Reset
Cancel
Help

Transmission encryption level
● None
○ Control
○ All data

Inbound request security
● Local
○ Verify
○ Trusted

☒ Send user ID with outbound requests

UserID for inbound requests: LINKTOSF

Key masks

Transaction level security key masks: none

Resource level security key masks: none

Figure 14. Security Communications Definitions (CD) panel

Selecting the **Local** option for **Inbound request security** indicates that CICS should run all incoming intersystem requests from the remote system under the user ID that is specified in **UserID for inbound requests**. The User Definitions (UD) entry for this user ID determines which resources these intersystem requests can access. This type of security is called *Link Security*, and is described in “CICS link security” on page 126.

Selecting either **Verify** or **Trusted** causes CICS to apply *User Security* to incoming intersystem requests. This means CICS uses the security information (such as user ID and password) that is sent with the intersystem request. CICS also uses the user ID that is specified in **UserID for inbound requests** to restrict the resources that inbound intersystem requests can access. For more information about User Security, see “CICS user security” on page 127.

Security information that is sent with outbound requests is controlled by the **Send user ID with outbound requests** options. “Setting up a CICS region to flow user IDs” on page 130 and “Setting up a CICS region to flow passwords” on page 131 describe how these options work.

After you have filled in all the attributes, select either **Permanent** or **Both** to create the CD entry.

If **Permanent** is selected, the CICS command that the IBM TXSeries Administration Tool will issue for the example shown above is:

```
cicsadd -r cics -P -c cd TOSF                        \
       ConnectionType=ppc_tcp                        \
       RemoteLUName="cicsosf1"        \
       RemoteTCPAddress="aix.cicsland.com"    \
       RemoteTCPPort=1435                              \
       ListenerName="CICSTCP"          \
       LinkUserId="LINKTOSF"
```

The **cicsadd** command is passed the name of each attribute with its required value. You can display this attribute name by placing the mouse pointer over the description of the attribute on the IBM TXSeries Administration Tool panel. Any attribute that is not specified on the **cicsadd** command is set to its default value. How to view and change the default values for CD entry attributes is described in “Configuring the default CD entry (CICS on Open Systems)” on page 56.

Refer to the *TXSeries for Multiplatforms Administration Reference* for a description of the **cicsadd** command.

Using SMIT to configure CD entries (AIX only)

When using CICS for AIX, you can use the AIX System Management Interface Tool (SMIT) to configure the Communications Definitions (CD) entries. To support CICS family TCP/IP, you must configure a CD entry with the **Connection type** field set to **ppc_tcp**.

To add a CD entry for CICS family TCP/IP to your region:

- Ensure that you are logged on to AIX with enough privileges to change the region database. (For example, log onto AIX as the **root** user.)
- Optionally set the environment variable **CICSREGION** to the name of your CICS region. For example:

```
$ export CICSREGION=cics
$
```

- Enter `smitty cicsregion` to start SMIT.
- Use option **Change Working CICS Region** to select your CICS region. (This is required only if you have not set up **CICSREGION** before starting SMIT.)
- Select options:

  ```
  ► Define Resources for a CICS Region
      ► Manage Resource(s)
          ► Communications
              ► Add New
  ```

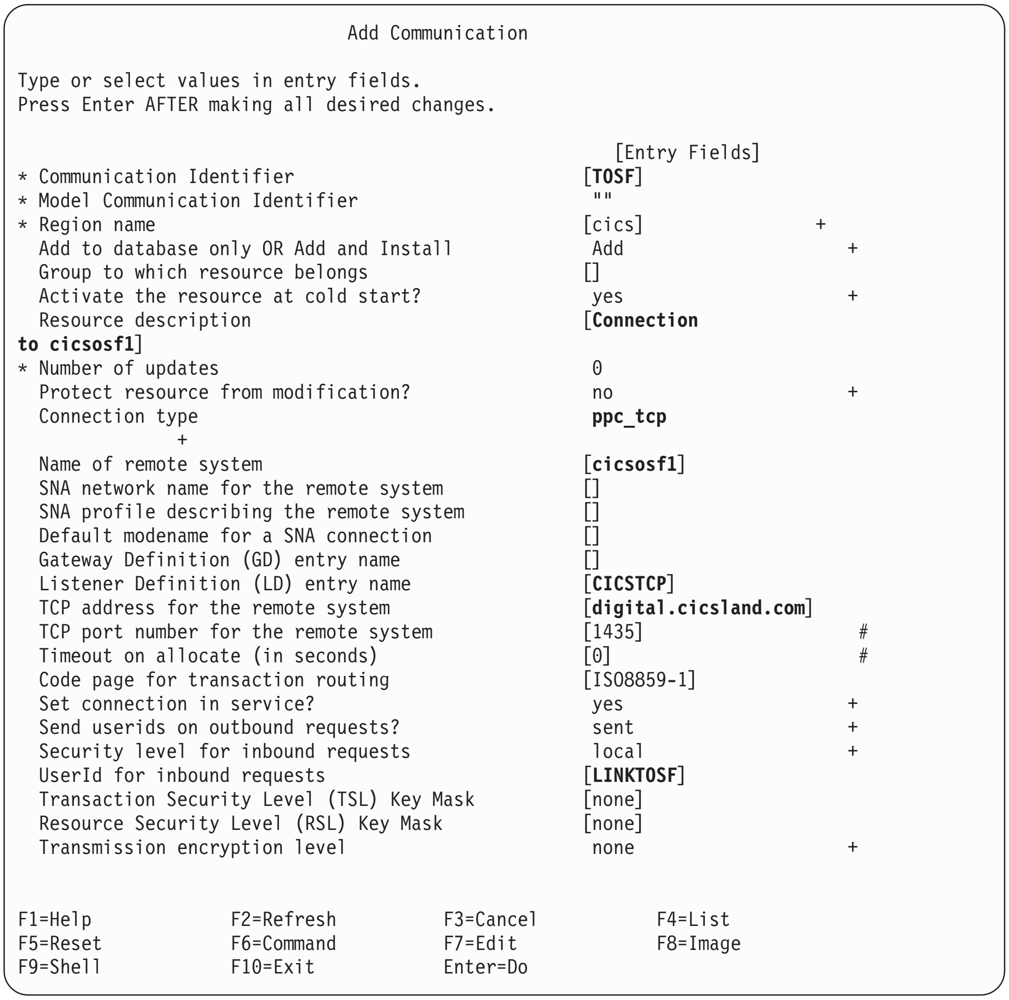
  This displays the **Add Communication** panel. Enter a model Communications Definition (CD) entry name. This could be the name of a CD entry that you have defined already. Alternatively, press the Enter key to use the default.

  Figure 15 on page 41 shows an example of a SMIT Add Communication panel.
- The **Communication Identifier** field is the name of the CD entry. This name is used in the SYSID option of CICS commands.
- The **Connection type** field specifies **ppc_tcp** to indicate that this is a CD entry for CICS family TCP/IP.
- You configure the location of the remote system by using the **TCP address for the remote system** and **TCP port number for the remote system** fields.
  - The **TCP address for the remote system** specifies the machine, or more specifically the network adapter card on the machine, on which the remote system is running. It can be specified in the following ways:
    - The Internet Protocol (IP) address in **dotted decimal** notation. For example, `1.23.45.67`. Do not use leading zeros when specifying an address in dotted decimal notation. CICS interprets such an entry as octal.
    - The Internet Protocol (IP) address in **dotted hexadecimal** notation. For example, `0x01.0x17.0x2D.0x43`.
    - The host name that is defined in the Internet name service. For example, `aix5.cicsland.com`. If a host name is used, it **must** map to only one IP address. You can check this by using the **host** command.
  - The **TCP port number for the remote system** field must be set to the port number that the remote system is using to listen for connection requests. The default value for **TCP port number for the remote system** is **1435**, which is the port assigned to CICS by the Internet Assigned Number Authority (IANA).
- The **Listener Definition (LD) entry name** field must be set to the name of an LD entry that is defined in the local region that has **Protocol type** of **TCP**. This LD entry defines the IP address and port number that the remote system is to use to contact the local region. CICS requires the **Listener definition (LD) entry name** field to be correctly configured, even if this connection is to be used only for outbound requests, because a **cicsip** process, which is started as a result of the LD entry, is required to open the TCP/IP connection. ("Using SMIT to configure LD entries (AIX only)" on page 31 describes how to set up a LD entry.)
- The security levels that your CICS region will use are configured with:
  - **Send userids on outbound requests?**
  - **Security level for inbound requests**
  - **UserId for inbound requests**
  - **Transaction Security Level (TSL) Key Mask**
  - **Resource Security Level (RSL) Key Mask**

  Chapter 6, "Configuring intersystem security," on page 123 describes how CICS security is configured.

- Refer to “Data conversion for transaction routing” on page 164 for information about how to configure the **Code page for transaction routing** field.
- The following fields are not required for CICS family TCP/IP connections, and can remain as the default value:
  - **SNA network name for the remote system**
  - **SNA profile describing the remote system**
  - **Gateway Definition (GD) entry name**
  - **Timeout on allocate (in seconds)**
  - **Transmission encryption level**
  - **Default modename for a SNA connection**

Figure 15 shows an example of the SMIT panel for adding a new CD entry.



```
                               Add Communication

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                     [Entry Fields]
* Communication Identifier                         [TOSF]
* Model Communication Identifier                    ""
* Region name                                      [cics]             +
  Add to database only OR Add and Install           Add                  +
  Group to which resource belongs                  []
  Activate the resource at cold start?              yes                  +
  Resource description                             [Connection
to cicsosf1]
* Number of updates                                 0
  Protect resource from modification?               no                   +
  Connection type                                   ppc_tcp
               +
  Name of remote system                            [cicsosf1]
  SNA network name for the remote system           []
  SNA profile describing the remote system         []
  Default modename for a SNA connection            []
  Gateway Definition (GD) entry name               []
  Listener Definition (LD) entry name              [CICSTCP]
  TCP address for the remote system                [digital.cicsland.com]
  TCP port number for the remote system            [1435]                 #
  Timeout on allocate (in seconds)                 [0]                    #
  Code page for transaction routing                [ISO8859-1]
  Set connection in service?                        yes                  +
  Send userids on outbound requests?                sent                 +
  Security level for inbound requests               local                +
  UserId for inbound requests                      [LINKTOSF]
  Transaction Security Level (TSL) Key Mask        [none]
  Resource Security Level (RSL) Key Mask           [none]
  Transmission encryption level                     none                 +


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 15. SMIT panel for adding a CD entry for CICS family TCP/IP support

Table 9. Road map

If you want to...	Refer to...
Review examples and a summary of CICS configurations	“CICS family TCP/IP Configuration examples” on page 43, and “Summary of CICS attributes for TCP/IP resource definitions” on page 54

Table 9. Road map (continued)

If you want to...	Refer to...
Read about configuring CICS resources	Chapter 5, “Configuring resources for intercommunication,” on page 109
Read about intercommunication security	Chapter 6, “Configuring intersystem security,” on page 123
Read about data conversion	Chapter 7, “Data conversion,” on page 147

Configuring an IBM CICS Client to use CICS family TCP/IP

This section describes how to configure an IBM CICS Client to connect to a CICS region through CICS family TCP/IP. For detailed information about how to configure a CICS Client, refer to *CICS Universal Client: Client Administration*.

The client initialization file needs a Server section for each region it will communicate with, and a Driver section for each Protocol used. Example Server and Driver sections are shown in Figure 16.

```
Server = cicsopen
    NetName = cicsopen.cicsland.com
    Protocol = TCPIP
    Description = CICS on cicsopen region
    UpperCaseSecurity = N
    InitialTransid = CESN
    ModelTerm = ibm-cics-client
    Port = 1435
Driver = TCPIP
     DriverName = CCLIBMIP
```

Figure 16. IBM CICS Client Server and Driver examples

The parameters are as follows:

Server
The name that the Client uses for the region.

NetName
The character or numeric TCP/IP identifier for the host. This can be an IP address (like 1.23.45.67), or a hostname (for example, cicsopen, or cicsopen.cicsland.com). A hostname is looked up in the Client’s HOSTS file, or its name server.

Protocol
This must match a Driver entry in the Client’s Driver section.

Description
An optional description of the server that is returned in some programming functions on the Client.

UpperCaseSecurity
It is recommended that this be set to `N` for a TXSeries for Multiplatforms region. This prevents user IDs and passwords being converted to uppercase by the Client (which it does by default). Unless all the user IDs and passwords on the region are in uppercase only, this conversion could cause errors.

InitialTransid
This parameter is optional. If present, it is used as the first transaction (with any parameters following it) run when the client terminal emulator connects to the server.

ModelTerm
This parameter is optional. When a Client terminal is autoinstalled into a TXSeries for Multiplatforms region, the model Terminal Definition used will have a DevType of this value. See the *TXSeries for Multiplatforms Administration Reference* for further details. The default value is `ibm-cics-client`.

Port
The port on which the CICS for Windows region listens for connections. If the parameter is omitted (or set to 0), the Client's TCP/IP SERVICES file is searched for a CICS entry. If this is not found, the default of 1435 is used.

Driver
Any 1- through 8-character name.

DriverName
The client device driver for the TCP/IP product that the Client uses.

Timeouts supported on TCP/IP connections between a TXSeries for Multiplatforms region and a Universal Client V3.1 or higher

A TXSeries for Multiplatforms region can time out transactions that are running over TCP/IP connections to an IBM CICS Universal Client V3.1 and higher based on the value that is specified in the region's Transaction Definitions (TD) **Timeout** attribute. When a Universal Client V3.1 or higher connects to a TXSeries for Multiplatforms region, it identifies itself as a client that can support timeout functions. In this case, the value that is set in the region's Transaction Definitions (TD) **Timeout** attribute automatically overrides any value that is set in the Region Definitions (RD) **XPRecvTimeout** attribute. If the region waits for a response from the client longer than the time that is specified in its Transaction Definitions (TD) **Timeout** attribute, the region abends the transaction and flows the abend to the client.

The following conditions can then occur:

- If the region receives a response from the client, the connection is maintained and the client can submit further transactions.
- If the region receives no response from the client, the TCP/IP connection to the client is checked. If still no response is received from the client, the connection closes and all transactions that are running over it are terminated.
- If the region cannot contact the client, the connection closes and all transactions that are running over it are terminated.

An IBM CICS Universal Client V3.1 and higher can time out External Call Interface (ECI) programs based on the value that is in the **eci_timeout** field in the ECI parameter block. If the client times out a transaction in this case, the transaction is purged on the server, and returns an **A147** abend code.

CICS family TCP/IP Configuration examples

The examples that follow show a CICS for Windows region called `cicswint` communicating with a CICS on Open Systems region, called `cics`, and with a CICS OS/2 region called `CICSOS2`. Table 8 on page 26 shows the TCP/IP host names and ports that the CICS regions use, and Figure 17 on page 44 summarizes the configuration that is discussed in this section.

Table 10. CICS family TCP/IP configuration example

	cicswint	cics	CICSOS2
Local TCP/IP host	wint127.cicsland.com	aix.cicsland.com	warp3.cicsland.com
Listening port	1435	1435	5566

When using CICS for AIX

The examples in this section show CICS commands being used to configure the resources. If you are using CICS for AIX, you could use the System Management Interface Tool (SMIT) to configure the resources.

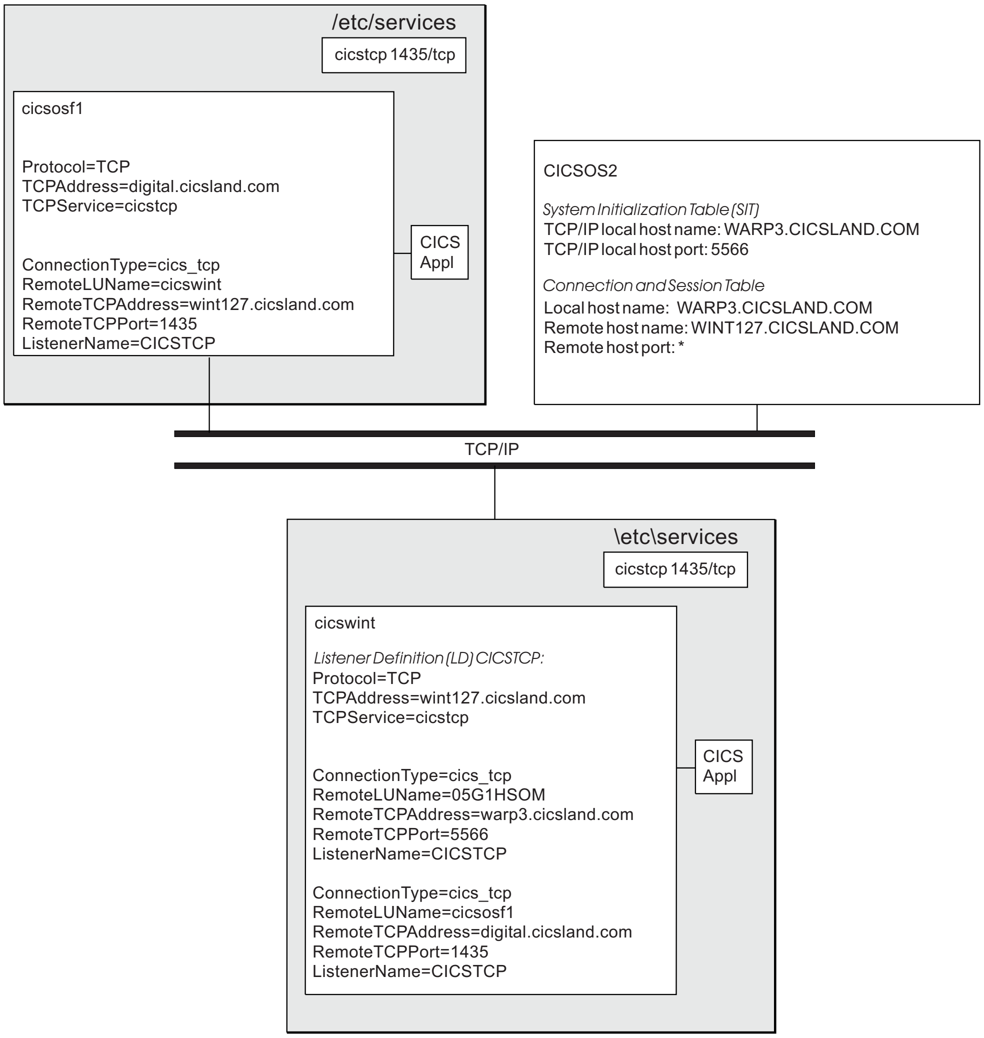
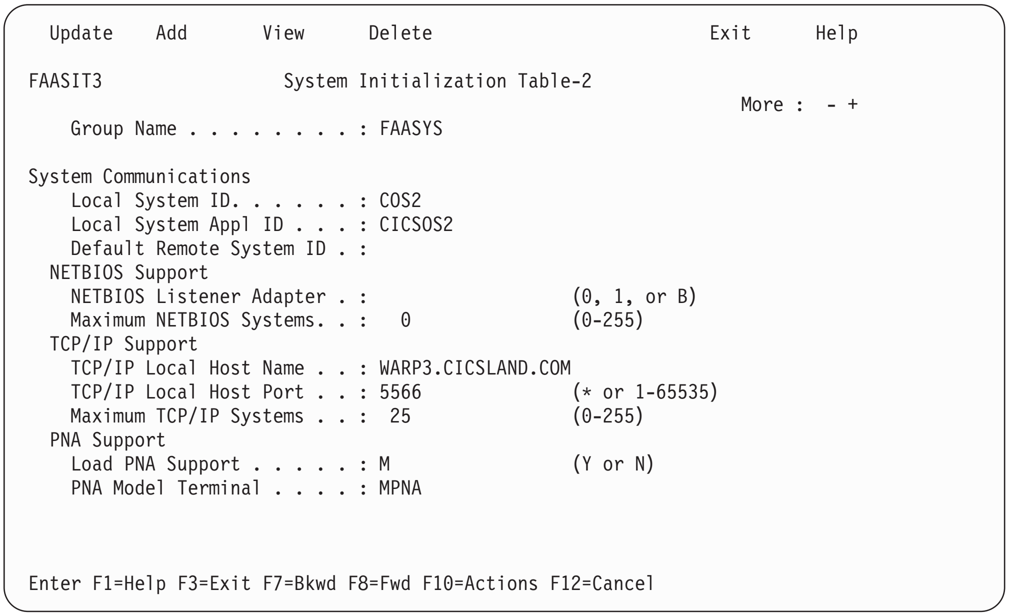


Figure 17. CICS family TCP/IP example

CICS OS/2 configuration

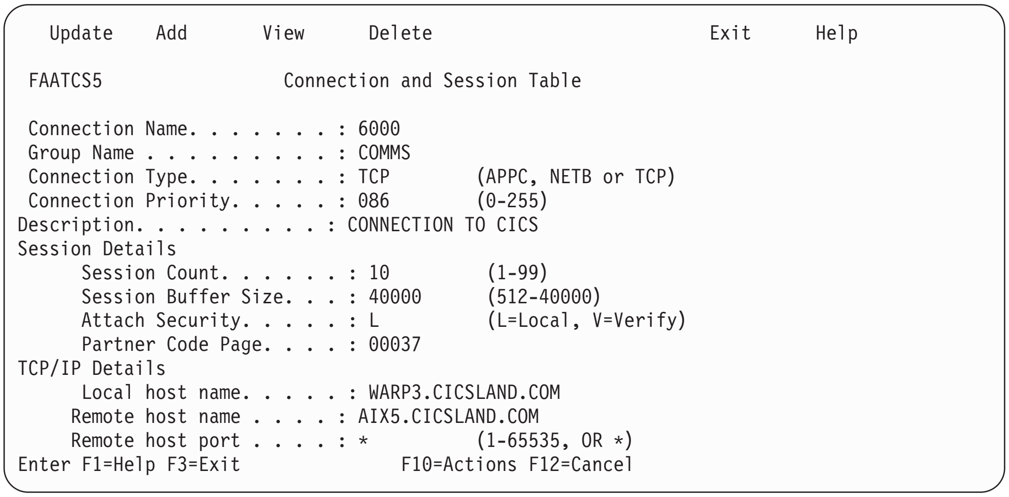
You define the CICS OS/2 listener in the System Initialization Table (SIT) by using the CEDA transaction. The host name of the local machine, `warp3.cicsland.com` and the listening port of `5566` is shown in Figure 18 on page 45.



```
  Update    Add       View      Delete                          Exit      Help

FAASIT3               System Initialization Table-2
                                                                    More :  - +
    Group Name . . . . . . . . : FAASYS

System Communications
    Local System ID. . . . . . : COS2
    Local System Appl ID . . . : CICSOS2
    Default Remote System ID . :
  NETBIOS Support
    NETBIOS Listener Adapter . :                  (0, 1, or B)
    Maximum NETBIOS Systems. . :   0              (0-255)
  TCP/IP Support
    TCP/IP Local Host Name . . : WARP3.CICSLAND.COM
    TCP/IP Local Host Port . . : 5566             (* or 1-65535)
    Maximum TCP/IP Systems . . :  25              (0-255)
  PNA Support
    Load PNA Support . . . . . : M                (Y or N)
    PNA Model Terminal . . . . : MPNA



Enter F1=Help F3=Exit F7=Bkwd F8=Fwd F10=Actions F12=Cancel
```

Figure 18. CICS OS/2 System Initialization Table

Figure 19 shows the CICS OS/2 definition of the connection from the `CICSOS2` region to the `cics` region. You define it in the Connection and Session Table (TCS®) by using the CEDA transaction. Note that CICS OS/2 expands the **Remote host port** value of "*" to 1435.



```
  Update    Add       View      Delete                          Exit      Help

FAATCS5               Connection and Session Table

Connection Name. . . . . . . : 6000
Group Name . . . . . . . . . : COMMS
Connection Type. . . . . . . : TCP        (APPC, NETB or TCP)
Connection Priority. . . . . : 086        (0-255)
Description. . . . . . . . . : CONNECTION TO CICS
Session Details
      Session Count. . . . . . : 10        (1-99)
      Session Buffer Size. . . : 40000     (512-40000)
      Attach Security. . . . . : L         (L=Local, V=Verify)
      Partner Code Page. . . . : 00037
TCP/IP Details
      Local host name. . . . . : WARP3.CICSLAND.COM
     Remote host name . . . . : AIX5.CICSLAND.COM
     Remote host port . . . . : *          (1-65535, OR *)
Enter F1=Help F3=Exit               F10=Actions F12=Cancel
```

Figure 19. CICS OS/2 Connection and session table

Configuration for region cicsopen

Figure 20 shows the Listener Definitions (LD) entry that enables `cicsopen` to receive CICS family TCP/IP connection requests from remote systems.

```
CICSTCP:
   ActivateOnStartup=yes
   Protocol=TCP
   TCPAddress="cicsopen.cicsland.com"
   TCPService="cicstcp"
```

Figure 20. Listener Definitions (LD) entry

The **TCPService** name is configured in the **/etc/services** file.

```
cicstcp            1435/tcp    # TCP Listener used by region cicsopen
```

Figure 21 shows the Communications Definitions (CD) entry for the connection from `cicsopen` to `CICSOS2`. The CICS OS/2 region is version 2, so you set the **RemoteLUName** attribute by using the **cicstcpnetname** command:

```
% cicstcpnetname -a warp3.cicsland.com -p 5566
05G1HSQM
```

```
TOS2:
  ConnectionType=cics_tcp
  RemoteLUName="05G1HSQM"
  RemoteTCPAddress="warp3.cicsland.com"
  RemoteTCPPort=5566
  ListenerName="CICSTCP"
  RemoteCodePageTR="IBM-037"
```

Figure 21. Communications Definitions (CD) entry

Figure 22 shows the CD entry for the connection from `cicsopen` to `cics`. Note that CICS on Open Systems regions use their region name for the **RemoteLUName** attribute.

```
TOSF:
  ConnectionType=ppc_tcp
  RemoteLUName="cics"
  RemoteTCPAddress="aix.cicsland.com"
  RemoteTCPPort=1435
  ListenerName="CICSTCP"
  RemoteCodePageTR="ISO8859-1"
```

Figure 22. Communications Definitions (CD) entry

Configuration for region cics

Figure 23 shows the Listener Definitions (LD) entry that enables `cicsosf1` to receive CICS family TCP/IP connection requests from remote systems.

```
CICSTCP:
   ActivateOnStartup=yes
   Protocol=TCP
   TCPAddress="aix.cicsland.com"
   TCPService="cicstcp"
```

Figure 23. Listener Definitions (LD) entry

The **TCPService** name is configured in the /etc/services file as follows:

```
cicstcp            1435/tcp    # TCP Listener used by region cics
```

Figure 24 on page 47 shows the Communications Definitions (CD) entry for the connection from `cics` to `cicswint`.

```
TWIN:
  ConnectionType=ppc_tcp
  RemoteLUName="cicswint"
  RemoteTCPAddress="wint127.cicsland.com"
  RemoteTCPPort=1435
  ListenerName="CICSTCP"
  RemoteCodePageTR="ISO8859-1"
```

Figure 24. Communications Definitions (CD) entry

Configuring CICS for CICS PPC TCP/IP support

To configure a connection for CICS PPC TCP/IP support you must:

- Configure a Communications Definitions (CD) entry for each remote region with the attribute **ConnectionType=ppc_tcp**.

  This is described in “Configuring CD entries for CICS PPC TCP/IP.”

Configuring CD entries for CICS PPC TCP/IP

A Communications Definitions (CD) entry describes details of a remote system and how your local region should communicate with it. To define a CD entry for a remote system, select your region name from the main panel of the IBM TXSeries Administration Tool. Then select the **Communication** resource from the **Resources** option of the **Subsystem** menu.

This displays the list of CD entries that you have already defined.

Select the **New** option of the **Communications** menu. The properties notebook window will be displayed.

The attributes for a CD entry are grouped into four pages:

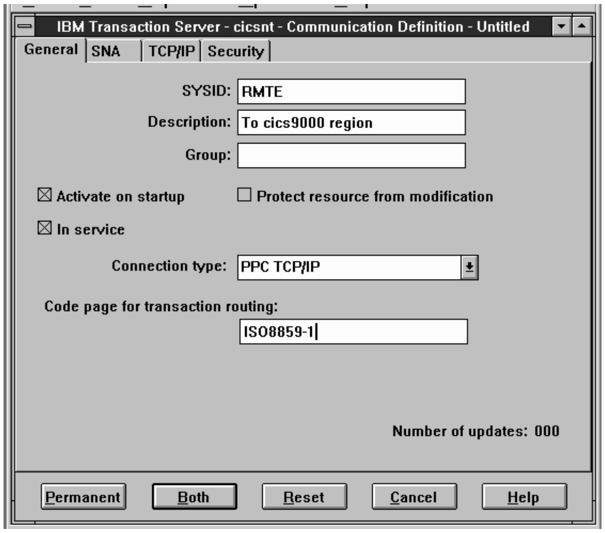
General
Attributes that are required by all CD entries.

SNA
Attributes that describe how the remote system communicates over an SNA network. Attributes on this page are not applicable to an CICS PPC TCP/IP CD entry.

TCP/IP
Attributes that describe how the remote system is connected to the TCP/IP network.

Security
Attributes that define how security is to be managed.

The **General** page is shown in Figure 25 on page 48.


IBM Transaction Server - cicsnt - Communication Definition - Untitled
General
SNA
TCP/IP
Security
Permanent
Both
Reset
Cancel
Help

SYSID: RMTE

Description: To cics9000 region

Group:

☒ Activate on startup

□ Protect resource from modification

☒ In service

Connection type: PPC TCP/IP

Code page for transaction routing: ISO8859-1

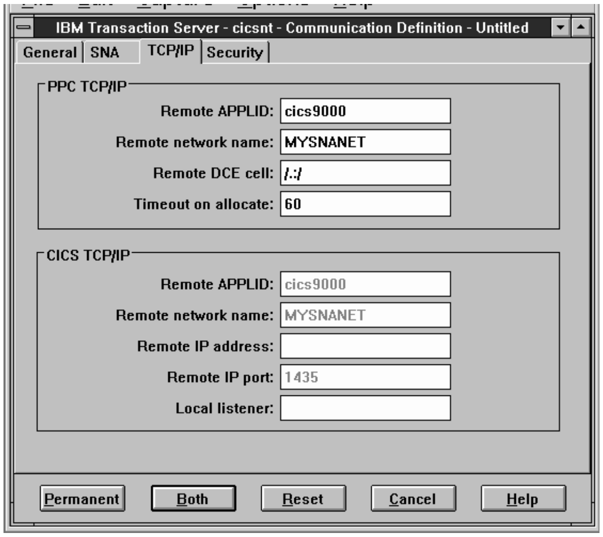
Number of updates: 000

Figure 25. General Communications Definitions (CD) panel

Enter the four-character SYSID for the CD entry. This is the CD entry's key, so it must be different from that which is used by all other CD entries in your region. However, the name is used only by local resources and applications, and does not have to be unique within the network. It does not have to relate to any other names in the remote system.

The **Connection type** should be changed to be **PPC TCP/IP**, and the **Code page for transaction routing** should be the code page that is used to flow transaction routing data across the network. The correct code page depends on the national language of your local region and the type of the remote system. Chapter 7, "Data conversion," on page 147 describes how to determine this value.

Select the **TCP/IP** tab to display the TCP/IP attributes, as shown in Figure 26. These attributes describe the remote system.


IBM Transaction Server - cicsnt - Communication Definition - Untitled
General
SNA
TCP/IP
Security
Permanent
Both
Reset
Cancel
Help

PPC TCP/IP

Remote APPLID: cics9000

Remote network name: MYSNANET

Remote DCE cell: /.:/

Timeout on allocate: 60

CICS TCP/IP

Remote APPLID: cics9000

Remote network name: MYSNANET

Remote IP address:

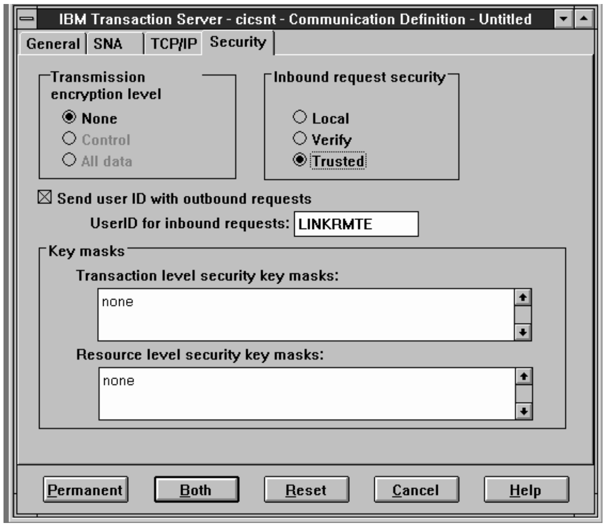
Remote IP port: 1435

Local listener:

Figure 26. TCP/IP Communications Definitions (CD) panel

If the remote system is a CICS region, the **Remote APPLID** is the name of this CICS region.

Select the **Security** tab to display the security attributes, which are shown in Figure 27. These attributes describe the security checking that is applied to all intersystem requests that use this CD entry.


IBM Transaction Server - cicsnt - Communication Definition - Untitled

Transmission encryption level

● None
○ Control
○ All data

Inbound request security

○ Local
○ Verify
● Trusted

☒ Send user ID with outbound requests

UserID for inbound requests: LINKRMTE

Key masks

Transaction level security key masks: none

Resource level security key masks: none

Figure 27. Security Communications Definitions (CD) panel

Selecting the **Local** option for **Inbound request security** indicates that CICS should run all incoming intersystem requests from the remote system under the user ID that is specified in **UserID for inbound requests**. The User Definitions (UD) entry for this user ID will determine which resources these intersystem requests can access. This type of security is called *Link Security*, and is described in “CICS link security” on page 126.

Selecting either **Verify** or **Trusted** results in CICS applying *User Security* to incoming intersystem requests. This means that CICS uses the security information (such as user ID and password) that is sent with the intersystem request. CICS also uses the user ID that is specified in **UserID for inbound requests** to restrict the resources that inbound intersystem requests can access. For more information about User Security, see “CICS user security” on page 127.

Security information that is sent with outbound requests is controlled by the **Send user ID with outbound requests** options. “Setting up a CICS region to flow user IDs” on page 130 and “Setting up a CICS region to flow passwords” on page 131 describe how these options work.

After you have filled in all the attributes, select either the **Permanent** or **Both** button to create the CD entry.

If **Permanent** is selected, the CICS command that the IBM TXSeries Administration Tool will issue for the example shown above is:

```
cicsadd -r cicsopen -P -c cd RMTE                                 \
                             ConnectionType=ppc_tcp                 \
                             RemoteLUName="cics9000"                \
                             LinkUserid="LINKRMTE"                  \
                             RemoteSysSecurity=trusted
```

The **cicsadd** command is passed the name of each attribute with its required value. You can display this attribute name by placing the mouse pointer over the description of the attribute on the IBM TXSeries Administration Tool page. Any attribute that is not specified on the **cicsadd** command is set to its default value. How to view and change the default values for CD entry attributes is described in "Configuring the default CD entry (CICS on Open Systems)" on page 56.

Refer to the *TXSeries for Multiplatforms Administration Reference* for a description of the **cicsadd** command.

Figure 28 illustrates how the RD and CD attributes in two CICS regions should correspond if they are to communicate.

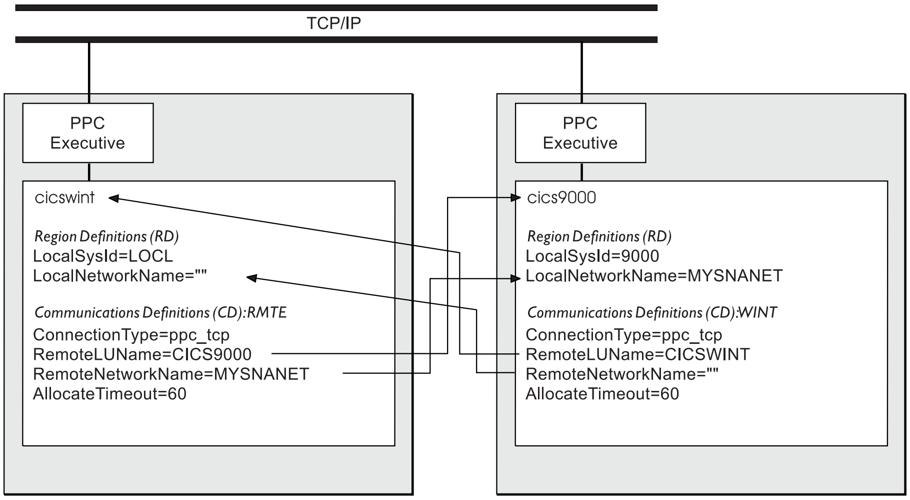


Figure 28. Two regions communicating across TCP/IP

Note that the **RemoteNetworkName** is given a value in one of the regions. This attribute is not required for a CICS PPC TCP/IP connection. However, if it is specified, it ***must*** match the **LocalNetworkName** value that is in the other region.

For CICS on Open Systems only:

You define a CICS PPC TCP/IP connection in your region by using a Communications Definitions (CD) entry that has **ConnectionType=ppc_tcp**.

The location of the remote system is configured in the CD entry with the **RemoteLUName** and **RemoteNetworkName** attributes.

If the remote system is a CICS on Open Systems and CICS for Windows region, you should set the **RemoteLUName** attribute to the name of the remote region, and the **RemoteNetworkName** to the value that is coded in the remote region's Region Definitions (RD) attribute **LocalNetworkName**.

The **AllocateTimeout** attribute defines, in seconds, how long CICS should wait for the remote system to accept an intersystem request. The default value is 0, which means "wait forever". You should set this timeout attribute to a value such as 60 in order to activate the timeout process. Then, if the remote system becomes overloaded, or fails while accepting an intersystem request, the CICS transaction that is issuing the intersystem request is not left hanging.

The following attributes are not required for a CICS PPC TCP/IP connection and can be left as the default value:

- **SNAConnectName**
- **GatewayName**
- **ListenerName**
- **RemoteTCPAddress**
- **RemoteTCPPort**
- **RemoteSysEncrypt**
- **DefaultSNAModeName**

Refer to "Data conversion for transaction routing" on page 164 for information about how to configure the **RemoteCodePageTR** attribute.

Refer to Chapter 6, "Configuring intersystem security," on page 123 for information about how configure the following security attributes:

- **OutboundUserIds**
- **RemoteSysSecurity**
- **LinkUserId**
- **TSLKeyMask**
- **RSLKeyMask**

Figure 29 shows a CICS on Open Systems or a CICS for Windows region named cicsopen communicating with a CICS on Open Systems region named cics9000, and shows the attributes that their CD entries use to define the connection between them.

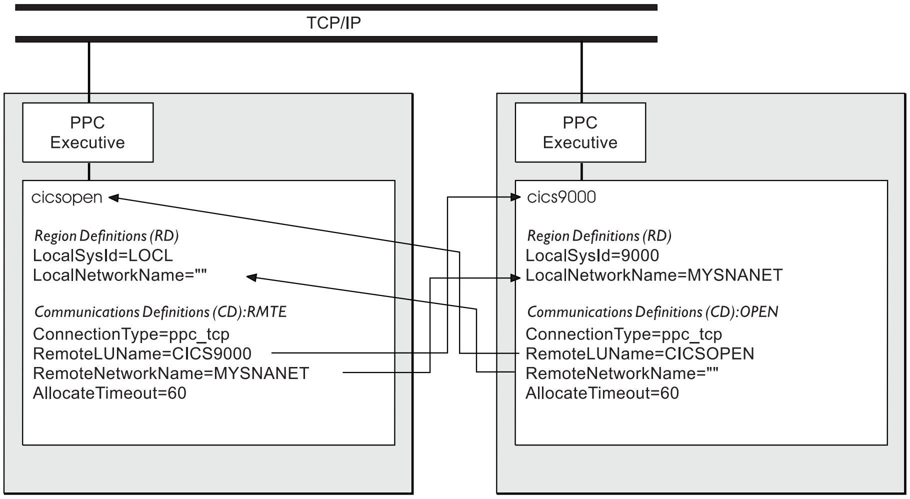


Figure 29. Two regions communicating across TCP/IP

The command shown below adds the RMTE CD entry to the region cicsopen's permanent database:

```
cicsadd -r cicssopen -P -c cd RMTE                                \
                             ConnectionType=ppc_tcp                    \
                             RemoteLUName="cics9000"      \
                             RemoteNetworkName="MYSNANET"  \
                             AllocateTimeout=60
```

The attributes that are not specified on the **cicsadd** command are set to their default values. You can view the default values for a CD entry by using the command `cicsget -r` *regionName* `-c cd "" .`

Using SMIT to configure CD entries (AIX only)

When using CICS for AIX, you can use the AIX System Management Interface Tool (SMIT) to configure the Communications Definitions (CD) entries. To add a CD entry for CICS PPC TCP/IP to your region, you must:

- Ensure you are logged on to AIX with enough privileges to change the region database. (For example, log onto AIX as the **root** user.)
- Optionally set the environment variable **CICSREGION** to the name of your CICS region. For example:

  ```
  % export CICSREGION=cics
  %
  ```

- Enter `smitty cicsregion` to start SMIT.
- Use option **Change Working CICS Region** to select your CICS region. (This is required only if you have not set up **CICSREGION** before starting SMIT.)
- Select options:

  ```
  ► Define Resources for a CICS Region
     ► Manage Resource(s)
        ► Communications
           ► Add New
  ```

  This displays the **Add Communication** panel. Enter a model Communications Definition (CD) entry name. This could be the name of a CD entry that you have defined already. Alternatively, press the Enter key to use the default.

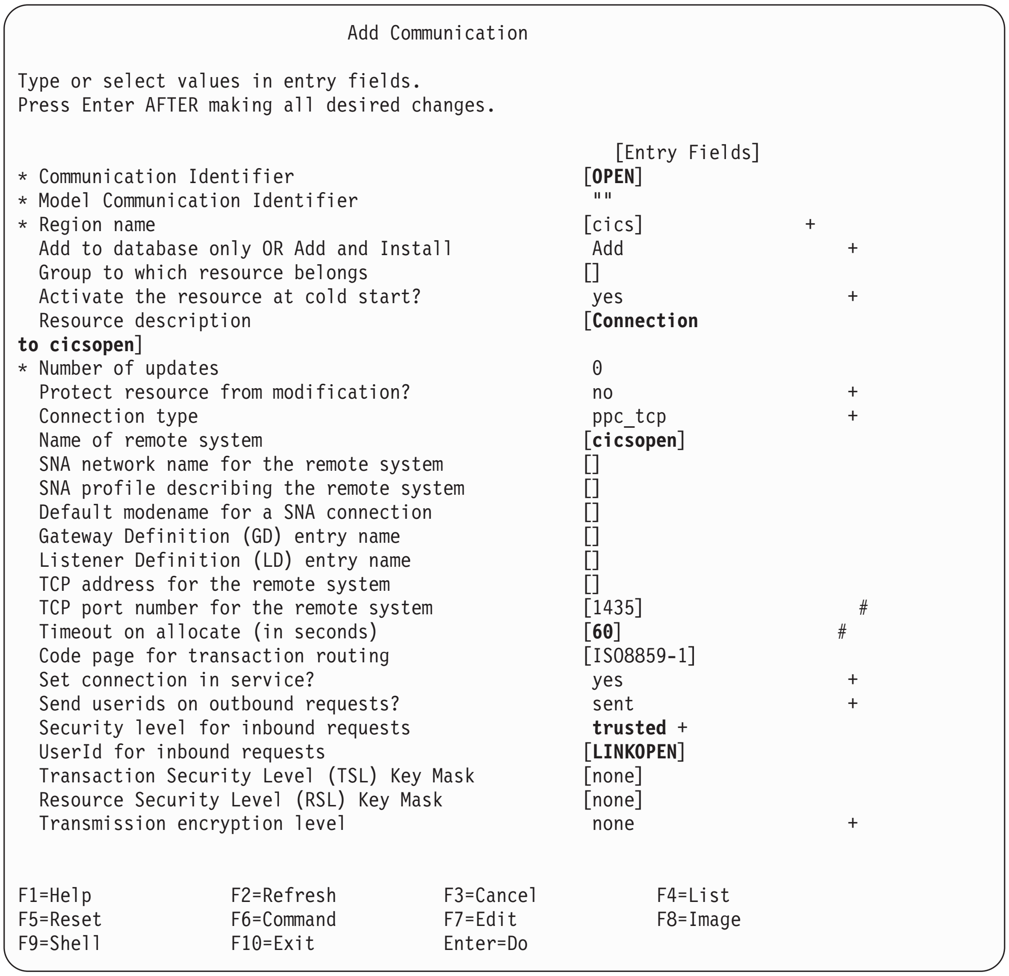
  Figure 30 on page 53 shows an example of a SMIT Add Communication panel.
- **Communication Identifier** is the name of the CD entry.
- **Name of remote system** is the LU name of the remote system, and **SNA network name for the remote system** is the name of the network that to which the remote system is attached.

  If the remote system is a CICS on Open Systems or CICS for Windows region, you should set the **Name of remote system** attribute to the name of the remote region, and the **SNA network name for the remote system** to the value that is coded in the remote region's Region Definitions (RD) attribute **Network name to which local region is attached**.
- **Timeout on allocate (in seconds)** defines how long your CICS region should wait for the remote system to accept requests. A value of 60 seconds is suggested. The default value is 0, which means "wait forever". You should set this timeout field to a value such as 60 in order to activate the timeout process. Then, if the remote system becomes overloaded, or fails while accepting an intersystem request, the CICS transaction that is issuing the intersystem request is not left hanging.
- The security levels that your CICS region will use are configured with:
  - **Send userids on outbound requests?**
  - **Security level for inbound requests**
  - **UserId for inbound requests**
  - **Transaction Security Level (TSL) Key Mask**
  - **Resource Security Level (RSL) Key Mask**

Chapter 6, “Configuring intersystem security,” on page 123 describes how CICS security is configured.

- Refer to “Data conversion for transaction routing” on page 164 for information about how to configure the **Code page for transaction routing** field.
- The following fields are not required for a CICS PPC TCP/IP connection and can remain as the default value:
  - **SNA profile describing the remote system**
  - **Gateway Definition (GD) entry name**
  - **Listener Definition (LD) entry name**
  - **TCP address for the remote system**
  - **TCP port number for the remote system**
  - **Transmission encryption level**
  - **Default modename for a SNA connection**
- Press the Enter key to create the CD entry.

Figure 30 shows an example of the SMIT panel for adding a new CD entry.



```
                               Add Communication

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                       [Entry Fields]
* Communication Identifier                          [OPEN]
* Model Communication Identifier                     ""
* Region name                                       [cics]              +
  Add to database only OR Add and Install            Add                    +
  Group to which resource belongs                   []
  Activate the resource at cold start?               yes                    +
  Resource description                              [Connection
to cicsopen]
* Number of updates                                  0
  Protect resource from modification?                no                     +
  Connection type                                    ppc_tcp                +
  Name of remote system                             [cicsopen]
  SNA network name for the remote system            []
  SNA profile describing the remote system          []
  Default modename for a SNA connection             []
  Gateway Definition (GD) entry name                []
  Listener Definition (LD) entry name               []
  TCP address for the remote system                 []
  TCP port number for the remote system             [1435]                   #
  Timeout on allocate (in seconds)                  [60]                   #
  Code page for transaction routing                 [ISO8859-1]
  Set connection in service?                         yes                    +
  Send userids on outbound requests?                 sent                   +
  Security level for inbound requests                trusted +
  UserId for inbound requests                       [LINKOPEN]
  Transaction Security Level (TSL) Key Mask         [none]
  Resource Security Level (RSL) Key Mask            [none]
  Transmission encryption level                      none                   +


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 30. SMIT panel for adding a CD entry for CICS PPC TCP/IP support

For more information, see the *TXSeries for Multiplatforms Administration Reference*.

Table 11. Road map

If you want to...	Refer to...
Read a summary of CICS configurations	"Summary of CICS attributes for TCP/IP resource definitions"
Read about configuring CICS resources	Chapter 5, "Configuring resources for intercommunication," on page 109
Read about controlling intersystem security	Chapter 6, "Configuring intersystem security," on page 123
Read about intersystem data conversion	.Chapter 7, "Data conversion," on page 147

Summary of CICS attributes for TCP/IP resource definitions

Table 12 lists the important resource definition attributes that are used for intersystem communications for TCP/IP connections.

Table 12. Comparison of CICS resource definitions TCP/IP connections

Resource Definition	CICS family TCP/IP connections	CICS PPC TCP/IP connections
Communications Definition (CD) **AllocateTimeout**	Not applicable.	Wait time in seconds for an intersystem request to be started in the remote system.
Communications Definition (CD) **ConnectionType**	Set to **ppc_tcp**.	
Communications Definition (CD) **GatewayName**	Not required.	Not required.
Communications Definition (CD) **ListenerName**	Set to the name of a locally defined Listener Definition (LD) entry that has **Protocol=TCP**.	Set to blank ("").
Communications Definition (CD) **RemoteLUName**	Set to the region name (APPLID) of the remote system, or netname returned by the **cicstcpnetname** command.	Set to the region name (APPLID) of the remote system.
Communications Definition (CD) **RemoteNetworkName**	Not required. Set to blank ("").	Not required. Set to blank ("") or the value from the **LocalNetworkName** attribute in the remote system's Region Definition (RD) entry.
Communications Definition (CD) **RemoteTCPAddress**	Set to the host name or Internet address of the remote network adapter.	Not required.
Communications Definition (CD) **RemoteTCPPort**	Set to the number of the port that the remote system is listening on.	Not required.
Communications Definition (CD) **SNAConnectName**	Not required.	Not required.
Region Definition (RD) **LocalLUName**	Not required.	Not required.

Table 12. Comparison of CICS resource definitions TCP/IP connections (continued)

Resource Definition	CICS family TCP/IP connections	CICS PPC TCP/IP connections
Region Definition (RD) **LocalNetworkName**	Not required. Set to blank ("") or the name of the local SNA network.	Not required. Set to blank ("") or the name of the local SNA network.
Transaction Definition (TD) **TPNSNAProfile**	Not required.	Not required.
Listener Definition (LD) entry	At least one LD entry is required with **Protocol=TCP**. The **TCPAddress** and **TCPService** attributes must also be configured.	Not required.
Gateway Definition (GD) entry	Not required.	Not required.

Configuring for autoinstallation of CD entries

Note on network protocols

CICS can autoinstall CD entries for IBM CICS Universal Clients that are connected over either TCP/IP or SNA networks. The information in this section applies to both network protocols.

CICS automatically creates (*autoinstall*) a Communications Definitions (CD) entry for an IBM CICS Client when it connects to your region. It also autoinstalls a CD entry if a remote CICS region acquires a CICS family TCP/IP connection to your local region and a suitable CD entry is not already defined in your region.

Irrespective of whether the remote system is an IBM CICS Client or a CICS region, the attributes that are assigned to an autoinstalled CD entry are set from:

- Information that is extracted from the network.
- Information that is received from the remote system. This information is received by the CICS transaction **CCIN**.
- Information from the CICS supplied program **DFHCCINX**. This program is called each time an autoinstall takes place. It is passed information about the remote system and can veto the install or change some of the attributes that are assigned to the CD entry. You can customize this program.
- Information that is configured in the default CD entry called **""**.

Table 13 shows how this information is used to set each of the CD attributes.

Table 13. How CD attributes are set in an autoinstalled CD entry

CD attribute	Value
SYSID (key of the CD entry)	An initial value is proposed by CICS that is unique in the region. The DFHCCINX program can override this. However, the autoinstall fails if DFHCCINX changes the SYSID to the name of an existing CD entry.

Table 13. How CD attributes are set in an autoinstalled CD entry (continued)

CD attribute	Value
RemoteLUName	An initial value is extracted from the network. The DFHCCINX program can override this. However, DFHCCINX should not change it when the network is between two CICS regions because this causes some intersystem requests to fail.
RemoteNetworkName **SNAConnectName** **GatewayName** **ListenerName** **RemoteTCPAddress** **RemoteTCPPort**	Values for these attributes are extracted from the network and from information that is received by the CCIN transaction.
InService **AllocateTimeout** **OutboundUserIds** **RemoteSysEncrypt** **DefaultSNAModeName**	Values for these attributes are retrieved from the default CD entry called **""**.
RemoteCodePageTR	An initial value can be received by the CCIN transaction and this is passed to DFHCCINX. DFHCCINX can use the received code page name, the **RemoteCodePageTR** attribute from the default CD entry (**""**), or supply a new value.
RemoteSysSecurity **LinkUserId**	Initial values for these attributes are taken from the default CD entry called **""**. DFHCCINX can override these values. DFHCCINX can also control how passwords that are received from CICS Clients are managed. This is explained further in "Controlling the management of passwords" on page 64.
TSLKeyMask **RSLKeyMask**	These attributes are always set to `all` in an autoinstalled CD entry. Therefore, if you want to restrict access to your region for autoinstalled CD entries, use the **LinkUserId** attribute.

The sections that follow describe how to configure your region so that autoinstalled CD entries have the correct attributes.

Configuring the default CD entry (CICS on Open Systems)

As shown in Table 13 on page 55, the default CD entry, **""**, is used as a template for autoinstalled CD entries. Therefore, it is important that it contains the attributes that you want to be assigned to autoinstalled CD entries. You can view the current settings of the default CD entry by using the **cicsget** command.

```
cicsget -c cd -r cicsunix ""

GroupName=""
ActivateOnStartUp=yes
ResourceDescription="Communications Definition"
AmendCounter=0
Permanent=no
ConnectionType=local_sna
RemoteLUName=""
RemoteNetworkName=""
SNAConnectName=""
GatewayName=""
ListenerName=""
RemoteTCPAddress=""
RemoteTCPPort=""
AllocateTimeout=0
RemoteCodePageTR="ISO8859-1"
InService=yes
OutboundUserIds=sent
RemoteSysSecurity=local
LinkUserId=""
TSLKeyMask=none
RSLKeyMask=none
RemoteSysEncrypt=none
DefaultSNAModeName=""
```

Use the information that is given in Table 13 on page 55 and “What the supplied version of DFHCCINX does” on page 58 to determine which attributes need changing. The **RemoteCodePageTR** attribute often needs changing in a DBCS environment. The following sections explain which CD attributes are important for each type of connection:

- “Configuring CD entries for CICS family TCP/IP(CICS on Open Systems only)” on page 34,
- “Configuring CD entries for CICS PPC TCP/IP” on page 47,
- “Configuring CICS for PPC Gateway server SNA support” on page 89

When you have determined which values you require for the default CD entry, use the **cicsupdate** command to change the attributes. For example, to change the **LinkUserId** attribute in the default CD entry:

```
% cicsupdate -c cd -r cicsunix "" LinkUserId="LINKAUTO"
```

When the required attributes are set in the default CD entry, restart your region to allow it to read the new values.

Configuring the default CD entry (CICS for Windows)

To view and possibly change the default attributes for a Communications Definitions (CD) entry, select your region name from the main panel of the IBM TXSeries Administration Tool. Then select the **Communication** resource from the **Resources** option of the **Subsystem** menu.

This displays the list of CD entries that you have already defined.

Select the **Defaults** option of the **Communications** menu. A properties notebook window is displayed showing the existing default values.

From this window you can change any of the default values. For example, if your region is running in a DBCS environment, you might want to change the **Code**

page for transaction routing to the code page that is used by your CICS clients, so that the supplied version of DFHCCINX sets up the correct code page when these clients connected.

Select **Permanent** to save any changes that you have made.

Configuring region database table sizes

The Region Definitions (RD) attribute **ClassTableSize** indicates how much storage is assigned to the internal hash tables that are used to access each resource definition entry. For maximum efficiency, the value that is supplied for each type of resource definition entry is normally twice the number of entries that are defined in the region database. So, for example, if you had five Communications Definitions (CD) entries defined in the region database, you would code a value of 10 for the CD.

When you are using autoinstall, the number of CD entries in the region database is changing. Therefore, when you code **ClassTableSize**, remember to include the number of expected autoinstall CD entries in your calculation.

Note: The hash table storage is allocated from the region pool size. Therefore, if you increase the amount of storage that is used by the hash tables, you might also need to increase the size of region pool that is configured in the RD attribute **MaxRegionPool**.

What the supplied version of DFHCCINX does

The supplied version of DFHCCINX accepts all installation requests. In addition, it does the following:

- Uses the values that are supplied by CICS for the CD entry name (SYSID) and RemoteLUName
- Checks the code page that is received from the remote system. If the remote system requested a code page, DFHCCINX checks whether a **iconv** conversion template is between the local code page and the remote code page. If a conversion template exists, the received code page is used for the **RemoteCodePageTR** attribute in the autoinstalled CD entry. If no conversion template exists, or if the remote system did not supply a code page, the code page from the **RemoteCodePageTR** attribute in the default CD entry **""** is used.
- Sets up client security. If the CD entry is for an IBM CICS Client that is using a CICS family TCP/IP connection ECI request, or a **cicslterm** on Windows, **RemoteSysSecurity** is set to **verify** and passwords are checked and discarded. Otherwise **RemoteSysSecurity** is set to the value from the default CD entry. The **LinkUserId** is set to the value from the default CD entry for all autoinstalled CD entries.
- Log a message to the CCIN log. DFHCCINX writes a message to the Transient Data queue, CCIN, which is defined to write to a file called:

  ```
  /var/cics_regions/regionName/data/CCIN.out
  ```

  For example:

```
date time CD entry 'TN03' for CICS server 'TNX43W03' has been
            installed with code page 'ISO8859-1'
```

DFHCCINX cannot prevent an uninstall (delete) of a CD entry. So for an uninstall request, DFHCCINX only logs a message to the CCIN log. For example:

```
 date time CD entry '@RP1' for CICS client '@RP1AAAA' has been deleted
```

DFHCCINX can be changed. If the default version of DFHCCINX does not match the needs of your region, refer to “The parameters passed to DFHCCINX” and “Writing your own version of DFHCCINX” on page 62, which describe the information that is passed to DFHCCINX, and how to use this information to write your own version.

The parameters passed to DFHCCINX

The parameters for DFHCCINX are passed a COMMAREA structure called **CICS_CCINX_Parameters**, which contains fields shown in Table 14

Table 14. Fields in the DFHCCINX COMMAREA structure, CICS_CCINX_Parameters

C datatype	Field name	Size
cics_sshort_t	Length	2-byte signed integer
cics_ubyte_t	RequestType	1-byte unsigned integer
cics_ubyte_t	SystemType	1-byte unsigned integer
cics_sshort_t	ApplIdLength	2-byte signed integer
cics_char_t	ApplId	9-byte character array
cics_sshort_t	SysIdLength	2-byte signed integer
cics_char_t	SysId	5-byte character array
cics_sshort_t	RemoteCodePageLength	2-byte signed integer
cics_char_t	RemoteCodePage	80-byte character array
cics_sshort_t	LocalCodePageLength	2-byte signed integer
cics_char_t	LocalCodePage	80-byte character array
cics_sshort_t	DefaultCodePageLength	2-byte signed integer
cics_char_t	DefaultCodePage	80-byte character array
cics_ubyte_t	RemoteSysSecurity	1-byte unsigned integer
cics_sshort_t	LinkUserIdLength	2-byte signed integer
cics_char_t	LinkUserId	9-byte character array
cics_ubyte_t	ConnectionType	1-byte unsigned integer
cics_ulong_t	RemoteTCPAddress	4-byte unsigned integer
cics_ushort_t	RemoteTCPPort	2-byte unsigned integer
cics_char_t	RemoteSNALUName	9-byte character array
cics_ubyte_t	ReturnCode	1-byte unsigned integer
cics_ulong_t	ConnectionProtection	4-byte unsigned integer

The fields in the COMMAREA for an install request are:

Length
The size of the CICS_CCINX_Parameters structure.

RequestType
Indicates whether it is an install or an uninstall request. For an install request this would be set to *0*.

SystemType
Indicates whether the install request is for a client or a server. For a client, this field is set to *0*; for a server, its value is *1*.

ApplIdLength
Length of the NETNAME in the Netname field.

ApplId
NETNAME of the remote system padded on the right with NULLs (0x00). The value that is returned in this field is set in the RemoteLUName attribute of the installed CD entry. Do not change this value if the CD entry is for a CICS server.

SysIdLength
The length of the SYSID in the SysId field.

SysId The four-character key for the CD entry padded on the right with NULLs (0x00).

RemoteCodePageLength
The length of the code page in RemoteCodePage.

RemoteCodePage
This is the code page that is received from the remote system, padded on the right with NULLs (0x00). The value in this field when DFHCCINX returns to CICS is used to set the RemoteCodePageTR attribute in the installed CD entry.

LocalCodePageLength
The length of the code page in LocalCodePage.

LocalCodePage
This is the code page for the local region, padded on the right with NULLs (0x00).

DefaultCodePageLength
The length of the code page in DefaultCodePage.

DefaultCodePage
This is the code page from the RemoteCodePageTR attribute of the default CD entry "", padded on the right with NULLs (0x00).

RemoteSysSecurity
This indicates the type of security that CICS should use when receiving requests from the remote system. This is the value from the RemoteSysSecurity attribute of the default CD entry, "". The value that DFHCCINX returns in this field is used to set the RemoteSysSecurity field in the installed CD entry. Use:

- *0* for **RemoteSysSecurity=local**
- *1* for **RemoteSysSecurity=verify**
- *2* for **RemoteSysSecurity=trusted**

If the remote system is an IBM CICS Client, this field is used to determine whether the password should be kept or discarded. Possible values of this field are:

- **0x00** to indicate that the password should be checked, then discarded
- **0x10** to indicate that the password should not be checked, and should be discarded
- **0x20** to indicate that the password should be checked, then kept
- **0x30** to indicate that the password should not be checked, but should be kept

The use of this field is explained in “Controlling the management of passwords” on page 64, and “Setting up a CICS region to flow passwords” on page 131.

LinkUserIdLength
The length of the user ID in LinkUserId.

LinkUserId
When DFHCCINX is called, this field contains the LinkUserId attribute (padded on the right with NULLs (0x00)) from the default CD entry, "". The value that is returned to CICS by DFHCCINX is set in the LinkUserId attribute of the installed CD entry.

ConnectionType
This indicates the type of connection on which the CCIN request was received. It is set to:

- *1* for **ConnectionType=local_sna**
- *2* for **ConnectionType=cics_tcp** or for a local client (**cicslterm**)
- *4* for **ConnectionType=ppc_gateway**
- *8* for **ConnectionType=ppc_tcp**
- *10* for **ConnectionType=cics_ipc**

RemoteTCPAddress
If ConnectionType=2 (cics_tcp, or a **cicslterm**), this field contains the TCP/IP address of the TCP connected client or system. Alternatively, if it is set to "12345678" (network order), it is a **cicslterm**.

RemoteTCPPort
If ConnectionType=2 (cics_tcp, or **cicslterm**), this field contains the TCP port number of the remote system. If it is set to zero (0), it is a **cicslterm**.

RemoteSNALUName
If ConnectionType=1 (local_sna) or if ConnectionType=4 (ppc_gateway), this field contains the SNA logical unit (LU) name of the remote system.

ReturnCode
Set this field to indicate whether CICS is to proceed with the installation request. Use:

- *0* to indicate that the install can proceed.
- *1* to indicate that the install can proceed, but a response is to be sent to the remote system to indicate that a problem exists with the code page.
- **0x10** to veto the install.

The fields in the COMMAREA for an uninstall request are:

Length
The size of the CICS_CCINX_Parameters structure

RequestType
Indicates whether it is an install or an uninstall request. For an uninstall request, this field is set to *1*.

SystemType
Indicates if the install request is for a client or a server. For a client, this field is set to *0*; for a server, it is set to *1*.

ApplIdLength
Length of the NETNAME in the Netname field.

ApplId
The value that is returned in this field is from the RemoteLUName attribute of the CD entry.

SysIdLength
The length of the SYSID in the SysId field.

SysId The four-character key for the CD entry.

The remainder of the fields in the uninstall COMMAREA are set to NULLs (Ox00).

Writing your own version of DFHCCINX

The source for the supplied version of DFHCCINX (cics_ccinx.ccs), along with a Makefile to build it, are in directory:

```
prodDir/samples/ccinx
```

In addition, a header file that describes the parameters that are passed to DFHCCINX, is provided in file *prodDir*/include/cics_ccinx.h. cics_ccinx.ccs is written in the C programming language, so the program begins at the **main** function. "What the supplied version of DFHCCINX does" on page 58 describes what this program does. The sections below suggest some changes to the supplied version. When you have created your own version, "Installing your version of DFHCCINX into the region" on page 65 describes how to install it in your region.

Note: If you want to change the functions in the supplied cics_ccinx.ccs file, copy cics_ccinx.ccs to your own directory before modifying it so that you can refer to the original version if necessary. In CICS on Open Systems, if you want to use the sample Makefile, copy that to you own directory also, and modify the PROGRAM. variable to the path name to which your compiled DFHCCINX should be written. The value of PROGRAM in the supplied Makefile overwrites the supplied DFHCCINX in *prodDir*/bin.

What DFHCCINX can do

DFHCCINX is passed a parameter structure in a COMMAREA that contains information about the CD entry that is to be installed or uninstalled. "The parameters passed to DFHCCINX" on page 59 describes this parameter structure.

For an install request, the parameter structure contains the values that CICS is using for some of the CD attributes. It also contains information about the remote system and the type of connection that is used to contact the region. Table 15 shows the values that can be changed by DFHCCINX. Alternatively, DFHCCINX can veto the installation request by setting the ReturnCode field to 0x10. DFHCCINX can also use EXEC CICS commands to perform any additional set up for the remote system. However, it must ***not*** access the principal facility because this is reserved for the use of CICS only.

Table 15. CD attributes that can be changed by DFHCCINX

COMMAREA field names	Corresponding CD attribute	Restrictions
SysId (and SysIdLength)	CD entry key	If this is changed to the name of a CD entry that is already defined in the region, the install fails.
ApplId (and ApplIdLength)	RemoteLUName	Do not change this field if the remote system is a CICS region. Otherwise, transaction routing and asynchronous processing requests that involve the remote region might fail.

Table 15. CD attributes that can be changed by DFHCCINX (continued)

COMMAREA field names	Corresponding CD attribute	Restrictions
RemoteCodePage (and RemoteCodePageLength)	RemoteCodePageTR	None
RemoteSysSecurity	RemoteSysSecurity	None
LinkUserId (and LinkUserIdLength)	LinkUserId	None

DFHCCINX cannot prevent an uninstall (delete) of a CD entry. It is called so that it can maintain private data or log messages.

Logging messages to the CCIN log

The supplied version of DFHCCINX logs information to the CCIN transient data queue each time it is called. This allows you to keep track of the remote systems that are connecting to your region. The function in the supplied version of DFHCCINX that writes the log messages is called **CCINX_LogMessage**. You can change this routine to log different information. Alternatively, if you do not need this information, you could remove the calls to CCINX_LogMessage from the **main** function of DFHCCINX. Removing this logging enables DFHCCINX to run faster, and saves disk space, especially if you have many IBM CICS Clients connecting to the region.

Assigning a value to RemoteCodePageTR

The supplied version of DFHCCINX validates the code page that is received from the remote system, by attempting to access the data conversion template that translates between the local code page and the code page that is requested by the remote system. This method works in all environments. However, it is an inefficient method for a particular installation in which the conversion templates available are static. Therefore, you can improve the performance of DFHCCINX by hardcoding the code page that is assigned to an autoinstall CD entry. For example, if all remote systems were to use the code page that was configured in the default CD entry, **""**, the function **CCINX_CheckCodePage** in the supplied DFHCCINX could be changed to copy the DefaultCodePage into the RemoteCodePage field. Alternatively, DFHCCINX could have a static table that assigns code pages based on the information about remote system, or the received code page.

Preventing autoinstall

For security reasons, you might want to prevent some or all CD entries from autoinstalling in your region. For example, you might want to disable autoinstall completely, or for a particular connection type. Alternatively, you might choose to allow autoinstall only for particular remote systems.

CICS does not autoinstall a CD entry if DFHCCINX returns `ReturnCode=0x10` in its parameter structure. Therefore, you can use the information that is passed to DFHCCINX to restrict when autoinstall is to proceed. In the example below, DFHCCINX is preventing all autoinstall requests for CICS family TCP/IP connections. This would mean that IBM CICS Clients could not connect to your region by using CICS family TCP/IP because they require an autoinstalled CD entry. A remote CICS region could connect across CICS family TCP/IP only if you had explicitly defined a CD entry for it.

```
   EXEC CICS ADDRESS COMMAREA(Parameters);

   if ((Parameters->RequestType == CICS_CCINX_REQUESTTYPE_INSTALL) &&;amp;
       (Parameters->ConnectionType == CICS_CCINX_CTYPE_CICSTCP))
 {
       Parameters->ReturnCode = CICS_CCINX_RETURNCODE_REJECT;
   }
```

Note: The constants that begin `CICS_CCINX_` are defined in the C language header file **cics_ccinx.h**, which is supplied with DFHCCINX.

Controlling the management of passwords

In some conditions, it might be necessary for the local CICS region to send a password and user ID to a remote system. This can occur if, for example, your CICS region is acting as a client gateway to a CICS for MVS/ESA™ host and you want to control all security with RACF® at the host. It is also needed when your SNA product does not support sending already_verified userids (such as the Microsoft Microsoft SNA Server for Windows), and you want to implement user security.

CICS does not normally save passwords that it receives when a user signs on. However, if you require CICS to pass the password on to a remote system, it must save it. The DFHCCINX exit supplies options that can be used to override entries in the default CD entry so that passwords that are received from CICS on Open Systems client ECI requests, or from CICS for Windows **cicslterm**, are kept so that they can later be passed on to a remote system.

Note: When CICS saves the password in storage, the password is encrypted. However when passwords are sent over an SNA network, they are sent in plain text. This is a requirement of the SNA architecture. Creating a DFHCCINX to save passwords does mean the overall system security is lowered. The default DFHCCINX, which does not save passwords, should therefore be used wherever possible.

The sending of passwords between clients and the local CICS region is controlled by the **RemoteSysSecurity** field in the DFHCCINX COMMAREA. The client can send a password when this field is set to **CICS_CCINX_SECURITYTYPE_VERIFY**. You can control whether CICS should save the password with the following values in the **RemoteSysSecurity**:

CICS_CCINX_PSWD_CHECK_AND_DROP (0x00)

Indicates that when CICS has verified a user ID and a password that it has received from the client or remote system, it should discard it. This means that if the request is routed on to another system, the password cannot accompany the request.

This value is used either of it is set explicitly, or if **CICS_CCINX_SECURITYTYPE_VERIFY** is **not** set (regardless of the setting of other values of the CICS_CCINX_PSWD_ field).

CICS_CCINX_PSWD_CHECK_AND_KEEP (0x20)

Indicates that when CICS has successfully verified a user and a password that it has received from the client or remote system, it should save it. This means that both the user ID and the password can accompany the request if it is routed on to another system. (The attribute **OutboundUserIds** in the CD entry for the system to which the request is routed is used to determine whether the password is actually sent.)

This value is ignored unless CICS_CCINX_SECURITYTYPE_VERIFY is also requested.

CICS_CCINX_PSWD_IGNORE_AND_DROP (0x10)
Indicates that when CICS has received a user ID and a password with a request, it is not to verify the password but should handle the user ID as if it were accompanied by a valid password. The password is discarded, which means that if the request is later routed on to another system, the password cannot accompany the request.

This value is ignored unless CICS_CCINX_SECURITYTYPE_VERIFY is also requested.

When this option is chosen, password validation for the client is disabled, including signon with the CESN transaction.

CICS_CCINX_PSWD_IGNORE_AND_KEEP (0x30)
Indicates that if CICS receives a user ID and a password with a request, it is not to verify the password but should handle the user ID as if it were accompanied by a valid password. The password should be saved so it is available to be forwarded to another system in that same way as the CICS_CCINX_PSWD_CHECK_AND_KEEP option is.

This value is ignored unless CICS_CCINX_SECURITYTYPE_VERIFY is also requested.

When this option is chosen password validation for the client is disabled, including signon with the CESN transaction.

If the passwords are kept, they can be sent from the local CICS region to a remote CICS region. This is set in the **OutboundUserIds** attribute in the CD. Relevant values of the **OutboundUserIds** attribute are:

sent_only_with_pswd
Send user ID with its password. If the password is not available, do not send a user ID.

sent_maybe_with_pswd
Send user ID with its password. If the password is not available, send the user ID already verified.

For examples of the use of these parameters, refer to "Setting up a CICS region to flow passwords" on page 131.

Installing your version of DFHCCINX into the region

The location of DFHCCINX is specified in the DFHCCINX Program Definition (PD) entry. The **PathName** attribute of this entry is set to **DFHCCINX**, which means that CICS uses the first version of the file DFHCCINX that it finds in the directories that are specified in the region's PATH. To install your own version of DFHCCINX either:

- Copy your version of DFHCCINX to a directory that is specified before *prodDir*/bin in your region's PATH. For example, `/var/cics_regions/`*`regionname`*`/bin`. This means that CICS finds your version of DFHCCINX before the supplied version in *prodDir*/bin.
- Alternatively, change the PD entry for DFHCCINX so that it specifies the location of your version of DFHCCINX in the **PathName** attribute. This PD entry is protected (**Permanent=yes**), so you must switch it to unprotected (**Permanent=no**) while you update it. The example below shows the PD entry for DFHCCINX being updated in both the permanent and running database of

CICS region **cicsopen**. The existing version is deleted from the running region. Then, the new value of **PathName** is set along with **Permanent=no**. Finally, the process is repeated to switch the PD entry back to **Permanent=yes**.

```
cicsdelete -c pd -r cicsopen -R DFHCCINX
cicsupdate -c pd -r cicsopen -B DFHCCINX PathName=cicsbin/DFHCCINX \
                  Permanent=no
cicsdelete -c pd -r cicsopen -R DFHCCINX
cicsupdate -c pd -r cicsopen -B DFHCCINX Permanent=yes
```

When the PD entry points to the location of your version of DFHCCINX, either restart your region, or use CEMT SET PROGRAM(DFHCCINX) NEW to bring the new DFHCCINX into use.

For more information, see the following sections:
- “IBM CICS Universal Client products” on page 4
- “Summary of CICS attributes for TCP/IP resource definitions” on page 54

Also refer to the description of Communications Definitions in *TXSeries for Multiplatforms Administration Reference*.

Chapter 4. Configuring CICS for SNA

This chapter describes how to configure a CICS region to use the Systems Network Architecture (SNA) protocol. TXSeries for Multiplatforms regions can use SNA to communicate with any other system that supports this protocol, including CICS Transaction Server for z/OS, CICS/ESA, CICS/MVS, CICS/400, CICS/VSE, and CICS OS/2.

To enable SNA communications, you must first identify your CICS region in the SNA network. The various names that identify a CICS region in the network are discussed in “Naming CICS regions in an SNA intercommunication environment.” Because the underlying SNA services are provided by a separate SNA product, you must then configure the SNA product. For information about how to configure a SNA product for CICS, see one of the following documents:

- *TXSeries for Multiplatforms Using IBM Communications Server for AIX with CICS*
- *TXSeries for Multiplatforms Using IBM Communications Server for Windows Systems with CICS*
- *TXSeries for Multiplatforms Using Microsoft SNA Server with CICS*
- *TXSeries for Multiplatforms Using HP-UX SNAplus2 with CICS*
- *TXSeries for Multiplatforms Using SNAP-IX for Solaris with CICS*

Each of these documents presents example configurations of TXSeries for Multiplatforms regions that are communicating with mainframe CICS and other TXSeries for Multiplatforms regions.

After your region is named and the SNA product is configured, you can configure your CICS region to implement SNA communications in two ways:

- Through local SNA support, discussed in “Configuring CICS for local SNA support” on page 70
- Through PPC Gateway server SNA support, discussed in “Configuring CICS for PPC Gateway server SNA support” on page 89

To use either of these implementation scenarios, you must configure various CICS resource definitions from the command line, by using the IBM TXSeries Administration Tool on Windows systems, or by using the AIX System Management Interface Tool (SMIT). Configuring resource definitions by using each of these configuration methods is outlined within both implementation scenario sections.

Autoinstallation of Communications Definitions (CD) entries

CICS can automatically install a Communications Definitions (CD) entry in the local region for an IBM CICS Client that is connected over an SNA network. CICS uses the same process to do this task as it does to install CD entries automatically for clients that are connected over a TCP/IP network. See “Configuring for autoinstallation of CD entries” on page 55 for more information.

Naming CICS regions in an SNA intercommunication environment

A CICS region can be known by several names in a SNA network. The following list identifies some of the most common:

Region name This one- through eight-character name is specified when you create the region. It is the name that is used in administrative commands. It is also known as the *APPLID*.

Local Logical Unit (LU) name

This one- through eight-character name is specified in the local region’s Region Definitions (RD) **LocalLUName** attribute. It identifies the local region to remote systems that are communicating with it through local SNA support. It must be unique within the SNA network. The first character of this name must be an uppercase alphabetic character (A through Z), and the subsequent characters must be either uppercase alphabetic or numeric characters.

For example, the following values are valid for the **LocalLUName** attribute:

- A
- CICS1
- CICSAIX
- MYLU

The following values are not valid:

- 9: Begins with a number
- 1CICS: Begins with a number
- CICSaix: Contains lowercase alphabetic characters
- MY-LU: Contains the unsupported character ’-’

If this attribute is left blank (""), CICS sets it to the region name and translates it to uppercase characters.

Gateway LU name

This one- through eight-character name is specified in the local region’s Gateway Definitions (GD) **GatewayLUName** attribute. It identifies the local region to remote systems that are communicating with the region through a PPC Gateway server. It must be unique within the SNA network. The first character of the this name must be an uppercase alphabetic character (A through Z), and the subsequent characters must be either uppercase alphabetic or numeric characters.

For example, the following values are valid for the **GatewayLUName** attribute:

- A
- CICS1
- CICSAIX
- MYLU

The following values are not valid:

- 9: Begins with a number
- 1CICS: Begins with a number
- CICSaix: Contains lowercase alphabetic characters
- MY-LU: Contains the unsupported character ’-’

If this attribute is left blank (""), CICS sets it to the value that is in the Region Definitions (RD) **LocalLUName** attribute. If the Region Definitions (RD) **LocalLUName** attribute is blank, CICS sets it to the region name and translates it to uppercase characters.

Local system identifier

This one- through four-character name is specified in the Region

Definitions (RD) **LocalSysId** attribute. It is referred to as the *SYSID*. Transactions that are running on the local system can obtain this name or use it to perform functions on the local region. By convention, a remote system can also use this same value as a Communications Definitions (CD) entry to identify the local region. Because this name can be used as a Communications Definitions (CD) entry in a remote system, ensure that it is different from the names of all CD entries that are used by your CICS region's local transactions.

Local network name

This one- through eight-character value for the Region Definitions (RD) **LocalNetworkName** attribute defines the network to which the region belongs. Although the local network name is not a name for the region, it can be combined with the value for the **LocalLUName** attribute to uniquely identify the region. The resulting name is referred to as the region's *network-qualified* or *fully qualified* LU name. For example, if a region that has CICSA as its value for the **LocalLUName** attribute belongs to a network called NETWORK1 (that is, the value for the region's Region Definitions (RD) **LocalNetworkName** attribute is NETWORK1), the region's fully qualified name is NETWORK1.CICSA. Typically, this name is used to refer to the region in the SNA communications product.

When assigning the values of the **LocalLUName** and **GatewayLUName** attributes to the region, ensure that these LU names are unique within the SNA network. The LU namesthat are used within the region also cannot conflict with one another. Some networks use naming conventions that help prevent name clashes. Consult your network administrator for help in choosing LU names.

Figure 31 shows a CICS region that is using both local SNA support and a PPC Gateway server.

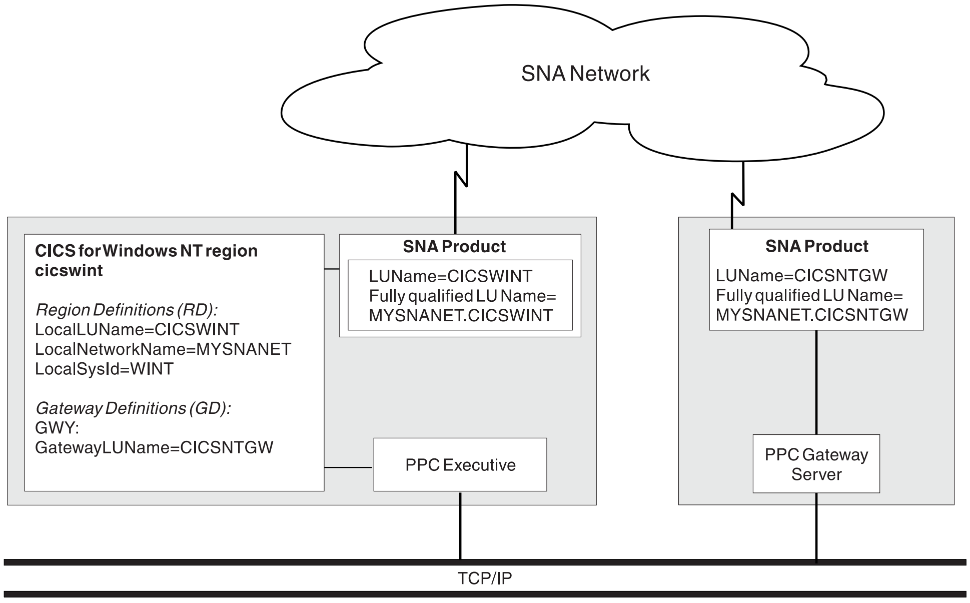


Figure 31. An example network

In Figure 31, remote systems that contact the region by way of local SNA support identify the CICS region by its Region Definitions (RD) **LocalLUName** attribute value CICSWINT. A remote system that contacts the region by way of a PPC Gateway server identifies the CICS region by its Gateway Definitions (GD) **GatewayLUName** attribute value CICSNTGW.

Note: If a **GatewayLUName** value is not specified, CICS uses the **LocalLUName** attribute value as the region identifier for remote systems. However, the same **LocalLUName** attribute value cannot be shared between local SNA support and a PPC Gateway server or among multiple PPC Gateway servers. Be sure to create a unique Gateway Definitions (GD) entry for each system that is connecting through a PPC Gateway server.

Each LU name that is used in your region must also be configured in the SNA product. Refer to that product's documentation for more information. If the system to which your region connects uses VTAM®, the name must be configured as a local LU name in VTAM.

Configuring CICS for local SNA support

Local SNA support lets TXSeries for Multiplatforms regions communicate with every other member of the CICS family. Synchronization levels 0 and 1 are supported. If you require synchronization level 2, you must use PPC Gateway server SNA support, as described in "Configuring CICS for PPC Gateway server SNA support" on page 89. (For descriptions of synchronization levels, see "Ensuring data integrity with synchronization support" on page 16.)

Local SNA support requires that an appropriate SNA product is installed and configured on the same machine as is the local TXSeries for Multiplatforms region. Such products are shown in Table 16.

Table 16. Communications products for various platforms to enable local SNA support

Platform	Communications product
Windows	IBM Communications Server or Microsoft SNA Server
AIX	IBM Communications Server
Solaris	SNAP-IX
HP-UX	SNAplus2

To use local SNA support, you must configure the following:

- A Listener Definitions (LD) entry, which describes how a CICS region connects to a network. An SNA LD entry indicates that a CICS region uses local SNA. The Listener Definition causes a Listener process to be started at region startup.
- Two Region Definitions (RD) attributes, **LocalLUName** and **LocalNetworkName**, which identify the LU name of the local CICS region and the network to which it belongs, respectively. Optionally, it is recommended that a short name for the local region be specified by using the **LocalSysId** attribute.
- A Communications Definitions (CD) entry for each remote system with which the local region is to communicate.

Note: You possibly need to configure the Transaction Definitions (TD) **TPNSNAProfile** attribute if both of the following conditions apply:

- The local CICS region is a CICS for AIX region.
- Remote systems will request transactions that reside in the local region.

See the *TXSeries for Multiplatforms Administration Reference* for more information.

Figure 32 shows the key configuration attributes. It shows a CICS region that has a local LU name of `CICSWINT` connected through local SNA to a remote system that has a local LU name of `CICSOS2`.

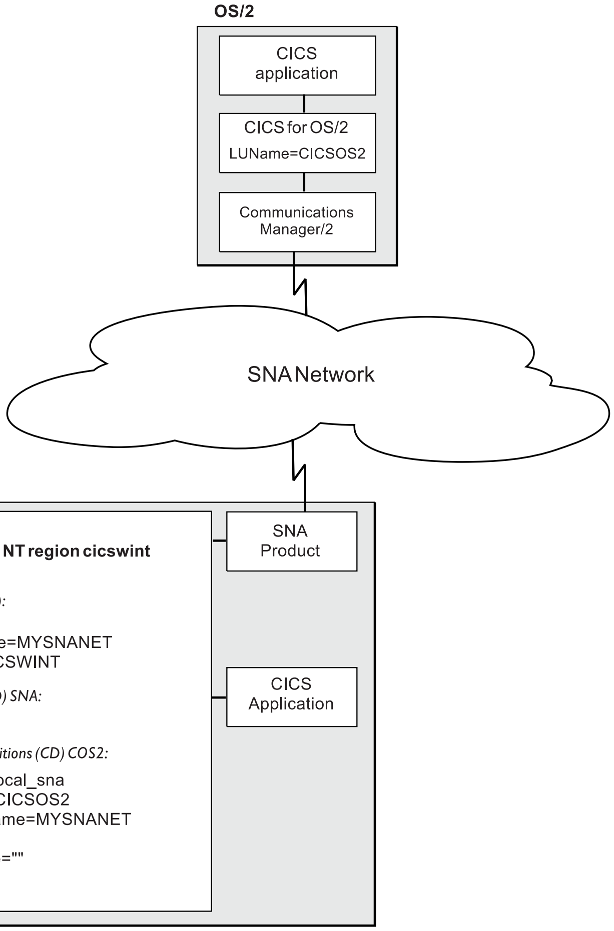


Figure 32. An example network using local SNA support

The CICS region has a Region Definitions (RD) entry that has the **LocalSysId** attribute identifying the short name for the local region, the **LocalNetworkName** attribute defining the name of the network to which the local region belongs, and the **LocalLUName** attribute defining the LU name of the local region. It has a Listener Definitions (LD) entry that has the **Protocol** attribute set to `SNA` to define local SNA support. It also has a Communications Definitions (CD) entry called `COS2`, which defines the remote CICS OS/2 system to the local region. The Communications Definitions (CD) **ConnectionType** attribute identifies the connection as `local_sna`, the **RemoteLUName** attribute defines the LU name of the remote region, and the **RemoteNetworkName** attribute defines the name of the network to which the remote region belongs. Leaving the **GatewayName** attribute blank ("") indicates that no PPC Gateway server is involved in the communications. Leaving the **SNAConnectName** attribute blank ("") indicates that no alias exists for the partner LU name in the remote system.

Note: The values of the attributes **GatewayName** and **SNAConnectName** are ignored when the region's Communications Definitions (CD) **ConnectionType** attribute identifies the connection as `local_sna`.

Figure 32 on page 71 shows just a few of the resource definition attributes that can be configured for SNA communications. For more complete lists of attributes that are commonly used in communications, see Table 17, Table 18 on page 73, Table 19 on page 74, and Table 20 on page 74. They list and define the resource definition attributes that are used for local SNA communications that are covered in this chapter. Command-line attribute names are given, along with their equivalent names in the IBM TXSeries Administration Tool and SMIT. The *TXSeries for Multiplatforms Administration Reference* and the *CICS Administration Guide* provide complete information about all CICS commands and definitions.

Note: The terminology in this chapter uses the attribute names of the CICS resource definitions that are referenced when CICS commands are used to configure a region. If you are using a CICS on Windows platform, you can use the IBM TXSeries Administration Tool instead of the CICS commands to configure the resources. Likewise, if you are using CICS for AIX, you can use SMIT to configure the resources.

Table 17. Comparison of CICS Communications Definitions (CD) for local SNA connections

Communications Definitions (CD) attributes	Equivalent IBM TXSeries Administration Tool resource definition attributes	Equivalent SMIT resource definition attributes	Use in local SNA connections
<Key>	SYSID	Communication Identifier	Specifies the name of the Communications Definitions (CD) entry.
ConnectionType	Connection type	Connection type	Specifies the type of the connection.
DefaultSNAModeName	Default SNA mode name	Default modename for a SNA connection	Specifies the SNA modegroup that is to be used for intersystem requests when an SNA modename is not specified in either the PROFILE option of the EXEC CICS ALLOCATE command or in the **SNAModeName** attribute of the Transaction Definitions (TD) entry.
LinkUserId	UserID for inbound requests	UserId for inbound requests	Specifies a locally defined user ID to associate with inbound requests.
ListenerName	The value is taken from the **Listener Name** attribute value.	Listener Definition (LD) entry name	Specifies the name of the Listener Definition.
OutboundUserIds	Sent user ID	Send user IDs on outbound requests?	Specifies whether a user ID with or without a password is sent with outbound requests.

Table 17. Comparison of CICS Communications Definitions (CD) for local SNA connections (continued)

Communications Definitions (CD) attributes	Equivalent IBM TXSeries Administration Tool resource definition attributes	Equivalent SMIT resource definition attributes	Use in local SNA connections
RemoteCodePageTR	Code page for transaction routing	Code page for transaction routing	Specifies the code page for transaction-routing data that is flowing between the local and remote regions.
RemoteLUName	Remote LU name	Name of remote system	Specifies the LU name of the remote system.
RemoteNetworkName	Remote network name	SNA network name for the remote system	Specifies the network name of the remote system.
RemoteSysSecurity	Inbound request security	Security level for inbound requests	Specifies how CICS is to process security information received with inbound requests.
RSLKeyMask	Resource level security key masks	Resource Security Level (RSL) Key Mask	Contains the list of resource link security keys that CICS uses to control access to resources from transactions.
SNAConnectName	SNA LU alias	SNA profile describing the remote system	Specifies the name of the remote system's partner LU alias.
TSLKeyMask	Transaction level security key masks	Transaction Security Level (TSL) Key Mask	Contains the list of transaction link security keys that CICS uses to control access to transactions.

Table 18. Comparison of CICS Listener Definitions (LD) for local SNA connections

Listener Definitions (LD) attributes	Equivalent IBM TXSeries Administration Tool resource definition attributes	Equivalent SMIT resource definition attributes	Use in local SNA connections
<Key>	Listener name	Listener Identifier	Specifies the name of the Listener Definitions (LD) entry. At least one LD entry is required with **Protocol=SNA**.
Protocol	Protocol	Protocol type	Specifies the communications protocol.

Table 19. Comparison of CICS Region Definitions (RD) for local SNA connections

Region Definitions (RD) attributes	Equivalent IBM TXSeries Administration Tool resource definition attributes	Equivalent SMIT resource definition attributes	Use in local SNA connections
LocalLUName	Local LU name	Local LU name	Specifies the local LU name of the region.
LocalNetworkName	Local network name	Network name to which local region is attached	Specifies the name of the local SNA network.
LocalSysId	Local SYSID	Region system identifier (short name)	Specifies the short name of the CICS region (as used in the SYSID option of several CICS commands).

Table 20. Comparison of CICS Transaction Definitions (TD) for local SNA connections

Transaction Definitions (TD) attributes	Equivalent IBM TXSeries Administration Tool resource definition attributes	Equivalent SMIT resource definition attributes	Use in local SNA connections
SNAModeName	SNA mode name	SNA modename for this transaction	Specifies the modename on an allocate request to a remote system that is connected by SNA.
TPNSNAProfile	Not applicable	SNA TPN profile for APPC listener program	**(AIX SNA only)** Specifies the name of an AIX SNA TPN profile that is configured in an SNA communications product, for example, `CICSTPN`.

For information about how to configure these attributes, see the following sections. Each section presents an example local SNA system configuration:

- "Configuring CICS for local SNA from the command line" on page 75 configures a local CICS region that is running on a Windows system to communicate with a remote CICS OS/2 region.
- "Configuring CICS for local SNA support by using the IBM TXSeries Administration Tool (CICS on Windows platforms only)" on page 77 configures a local CICS region that is running on a Windows system to communicate with a remote CICS OS/2 region.
- "Configuring CICS for local SNA support by using SMIT (CICS for AIX only)" on page 82 configures a local CICS for AIX region to communicate with a remote CICS OS/2 region.

Many attribute values must match associated values that are used in related communications products, such as IBM Communications Server for Windows or IBM Communications Server for AIX. See the communications product's administration documentation for more information.

Configuring CICS for local SNA from the command line

This section uses the CICS commands **cicsadd** and **cicsupdate** to add or update intercommunication resources. For more information about the use of these commands, see the *TXSeries for Multiplatforms Administration Reference*.

To configure CICS for local SNA, perform the following procedure (assume that default values are accepted for any attributes that are not discussed):

1. Configure a Listener Definitions (LD) entry by issuing the **cicsadd** command, as shown in the following example:

   ```
   cicsadd -r cicswint -P -c ld LOCALSNA Protocol=SNA
   ```

   In this example, a Listener Definitions (LD) entry called `LOCALSNA` with the attribute `Protocol=SNA` is added to the permanent database of the region cicswint. (If you add an LD entry to the region while it is running, local SNA support is not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the entry is added only to the permanent database, the region must be cold-started.)
2. Update the **LocalLUName**, **LocalNetworkName**, and **LocalSysId** Region Definitions (RD) attributes by issuing the **cicsupdate** command, as shown in the following example:

   ```
   cicsupdate -r cicswint -P -c rd LocalLUName="CICSWINT" \
             LocalNetworkName="MYSNANET"                \
             LocalSysId="WINT"
   ```

   In this example, the Region Definitions (RD) attribute **LocalLUName** is set to `CICSWINT`, the **LocalNetworkName** attribute is set to `MYSNANET`, and the **LocalSysId** attribute is set to `WINT`. These values are added to the permanent database of region cicswint. (If you update RD attributes in a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the attributes are updated only in the permanent database, the region must be cold-started.)
3. Configure a Communications Definitions (CD) entry for each remote system with which the local system is to communicate by issuing the **cicsadd** command, as shown in the following example:

   ```
   cicsadd -r cicswint -P -c cd COS2     \
          ConnectionType=local_sna      \
          RemoteLUName="CICSOS2"        \
          RemoteNetworkName="MYSNANET"  \
          RemoteCodePageTR="ISO8859-1"  \
          LinkUserId="LINKCOS2"
   ```

   In this example, a Communications Definitions (CD) entry called `COS2` is added to the cicswint region's permanent database. (If you add a CD entry to a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the entry is added only to the permanent database, the region must be cold-started.)

   The attributes listed in this example configure the region as follows:

   - The **ConnectionType** attribute value `local_sna` specifies that local SNA is to be used.

- The **RemoteLUName** attribute value `CICSOS2` specifies the LU name of the remote system.
- The **RemoteNetworkName** attribute value `MYSNANET` defines the SNA network to which the remote system is connected.
- The **RemoteCodePageTR** attribute value (in this case, `ISO8859–1`) determines which character set flows across the network during transaction routing. The correct code page depends on the national language of your local region and the remote system type. See “Data conversion for transaction routing” on page 164 for information about how to choose this value.
- The **LinkUserId** attribute value `LINKCOS2` identifies a locally defined user ID that can be associated with inbound requests. Its value is related to the value of the **RemoteSysSecurity** attribute, which, in this case, is left at the default value of `local`.

The following attributes can also be specified:

- The **SNAConnectName** attribute specifies the name of the partner LU alias that is defined in the SNA product. It is required only if the partner LU alias is different from the partner LU name. (The *partner LU* is the SNA communication product’s term for the remote region’s LU name. The *partner LU alias* is an associated name for the partner LU; it is configured in the SNA communications product.)
- The **DefaultSNAModeName** attribute specifies the SNA modegroup that is to be used for intersystem requests when an SNA modename is not specified either in the PROFILE option of the EXEC CICS ALLOCATE command, or in the **SNAModeName** attribute of the Transaction Definitions (TD) entry. (A *modegroup*, or *modename*, is defined in the SNA communications product. It defines the number of sessions that are associated with a connection, and the characteristics of those sessions.) See one of the following documents for more information, depending on your SNA communications product:
  - *TXSeries for Multiplatforms Using IBM Communications Server for AIX with CICS*
  - *TXSeries for Multiplatforms Using IBM Communications Server for Windows Systems with CICS*
  - *TXSeries for Multiplatforms Using Microsoft SNA Server with CICS*
  - *TXSeries for Multiplatforms Using HP-UX SNAplus2 with CICS*
  - *TXSeries for Multiplatforms Using SNAP-IX for Solaris with CICS*

  If the **DefaultSNAModeName** attribute is set to "" (its default value), and the transaction has not provided a modename, the modename that is configured for the SNA product is used. Refer to “Default modenames” on page 110 for more information about the use of this attribute.
- The **OutboundUserIds** attribute determines the security information that is sent with outbound requests. “Setting up a CICS region to flow user IDs” on page 130 and “Setting up a CICS region to flow passwords” on page 131 describe how this option works.
- The **RemoteSysSecurity** attribute specifies the type of security that is required:
  - Using the default value of **local** causes CICS to run all incoming intersystem requests from the remote system under the user ID value that you specify in the **LinkUserId** field. The User Definitions (UD) entry for this user ID determines which resources these intersystem requests can access. This type of security is called *link security* and is described in “CICS link security” on page 126.
  - Choosing the **verify** or **trusted** values causes CICS to use the security information (such as the user ID and password) that is sent with the

intersystem request. CICS also uses the user ID that you specify in the **LinkUserId** field to restrict the resources that inbound intersystem requests can access. This type of security is called *user security* and is described in "CICS user security" on page 127.

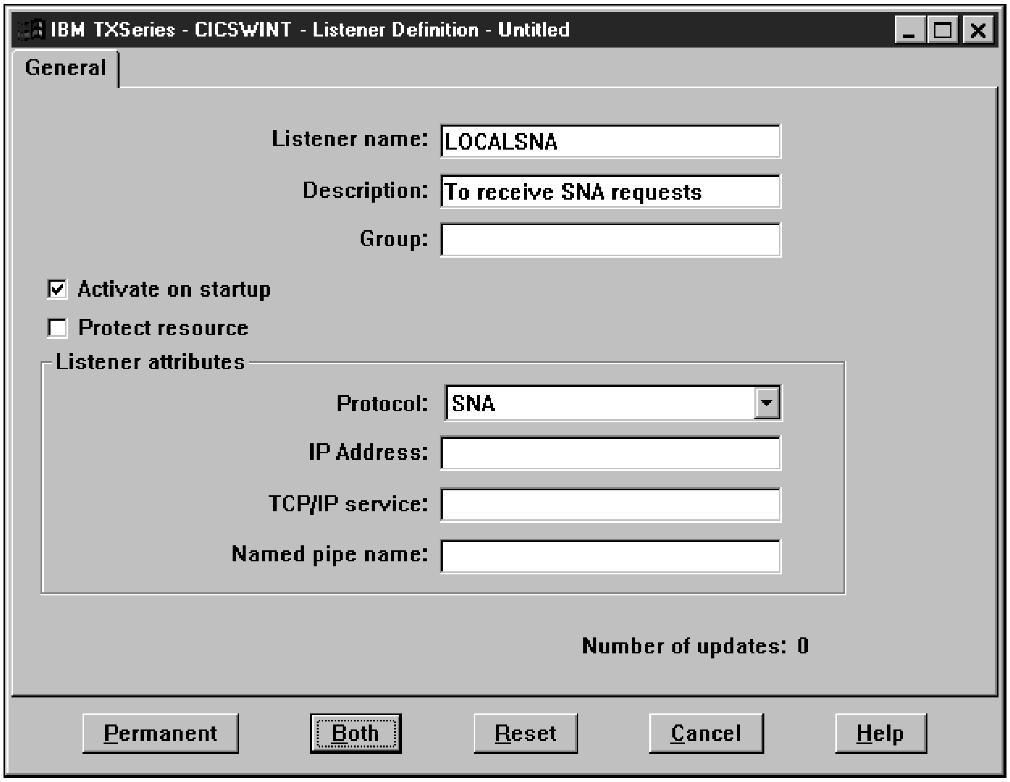
- See Chapter 6, "Configuring intersystem security," on page 123 for more information about how CICS security is configured, including descriptions of the **TSLKeyMask** and **RSLKeyMask** attributes.

Configuring CICS for local SNA support by using the IBM TXSeries Administration Tool (CICS on Windows platforms only)

To configure CICS for local SNA by using the IBM TXSeries Administration Tool, you must configure a Listener Definitions (LD) entry, update the **Local network name**, **Local SYSID**, and **Local LU name** Region Definitions (RD) attributes, and configure a Communications Definitions (CD) entry for each remote system with which your system is to communicate. Perform the following procedures to set these definitions (assume that default values are accepted for any attributes that are not discussed):

First, configure a Listener Definitions (LD) entry by doing the following:

1. In the IBM TXSeries Administration Tool Administration window, click the region name.
2. Click **Subsystem>Resources>Listener**. The Listeners window opens.
3. Click **Listeners>New**. The Listener Definition window opens.
4. In the **Listener name** field, type a name for the Listener Definitions (LD) entry. This name can be up to eight characters in length and must be different from the names of all other LD entries that are in your region. It does not have to be unique within the network and does not relate to any other names in the network.
5. Enter a description if desired.
6. Click the **SNA** option from the **Protocol** menu. An example configuration window is shown in Figure 33 on page 78.



Listener name: LOCALSNA

Description: To receive SNA requests

Group:

☑ Activate on startup

☐ Protect resource

Listener attributes

Protocol: SNA

IP Address:

TCP/IP service:

Named pipe name:

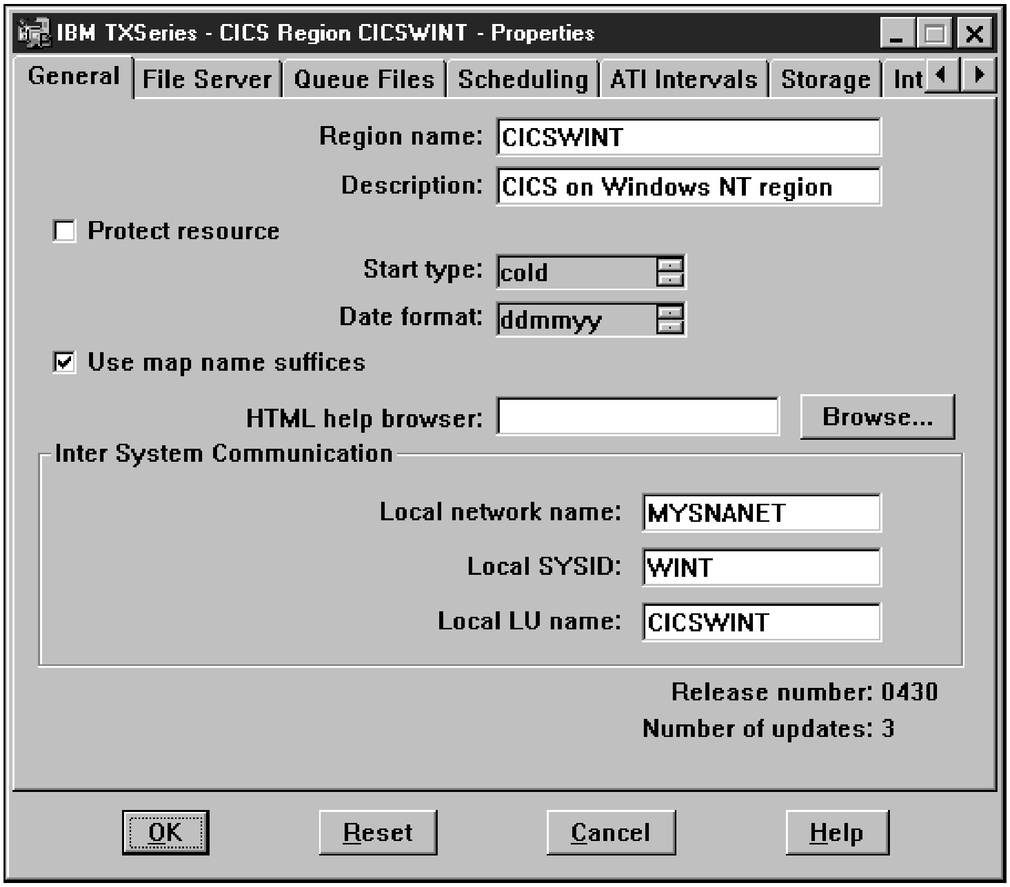
Number of updates: 0

Figure 33. Listener Definition window

7. Click the **Permanent** button to add the Listener Definitions (LD) entry to the permanent database, or click the **Both** button to add the entry to the permanent and runtime databases.

Next, update the **Local network name**, **Local SYSID**, and **Local LU name** Region Definitions (RD) attributes by doing the following:

1. In the IBM TXSeries Administration Tool Administration window, click the region name.
2. Click **Subsystem>Properties**. The Properties window opens.
3. Enter a description if desired.
4. Type the region’s network name in the **Local network name** field.
5. Type the region’s short name in the **Local SYSID** field.
6. Type the region’s local LU name in the **Local LU name** field. An example configuration window is shown in Figure 34 on page 79.



IBM TXSeries - CICS Region CICSWINT - Properties

Region name: CICSWINT

Description: CICS on Windows NT region

□ Protect resource

Start type: cold

Date format: ddmmyy

☑ Use map name suffices

HTML help browser:

Inter System Communication

Local network name: MYSNANET

Local SYSID: WINT

Local LU name: CICSWINT

Release number: 0430

Number of updates: 3

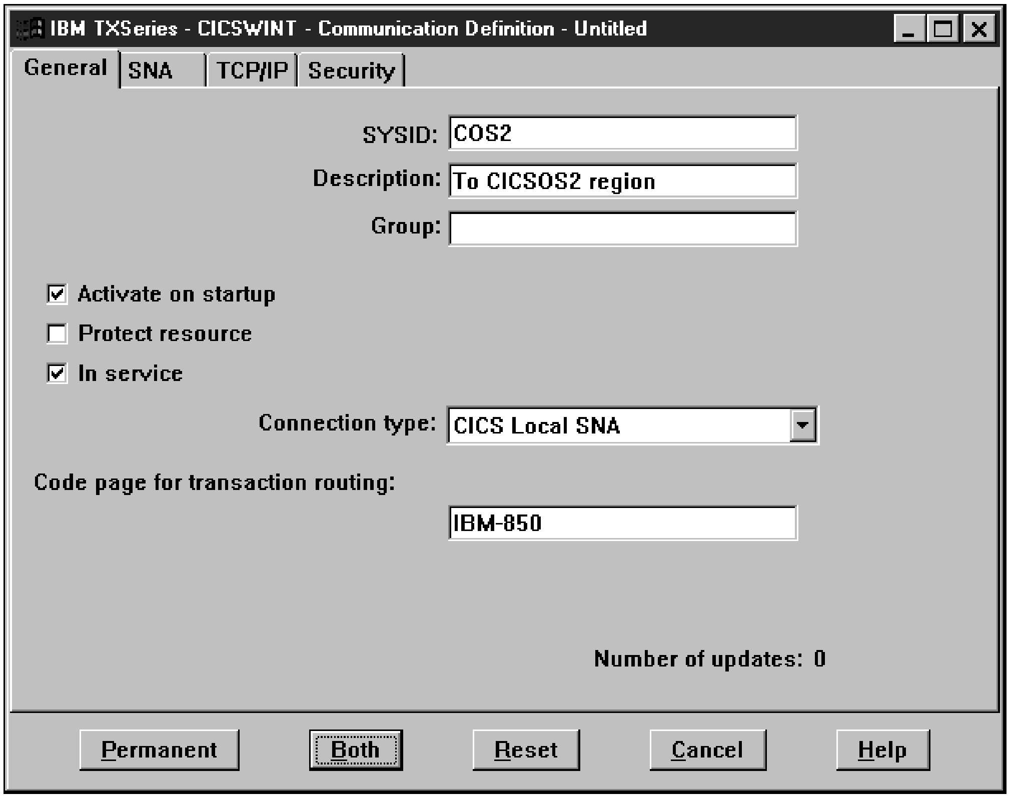
Figure 34. Region Definition Properties window

7. Click **OK** to add the updated Region Definitions (RD) attributes to the permanent database. (If you update RD attributes in a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the attributes are being updated only in the permanent database, the region would have to be cold-started.)

Finally, configure a Communications Definitions (CD) entry for each remote system with which your system is to communicate by doing the following:

1. Click the region name in the IBM TXSeries Administration Tool Administration window.
2. Click **Subsystem>Resources>Communication**. The Communications window opens.
3. Click **Communications>New**. The Communication Definition window opens. The attributes for a Communications Definitions (CD) entry are grouped on four tabs:
   - **General**: Contains attributes that are required by all CD entries.
   - **SNA**: Contains attributes that describe how the remote system communicates over an SNA network.
   - **TCP/IP**: Contains attributes that describe how the remote system is connected to a TCP/IP network. Attributes on this page are not applicable to a local SNA CD entry.
   - **Security**: Contains attributes that define how security is managed.

4. On the **General** tab:
   a. In the **SYSID** field, type a four-character name for the Communications Definitions (CD) entry. This name must be different from the names of all other CD entries that are in your region. However, it does not have to be unique within the network and does not relate to any other names in the network.
   b. Type a description if desired.
   c. Select the **CICS Local SNA** option from the **Connection type** menu.
   d. Accept the default value, or type the appropriate value in the **Code page for transaction routing** field. This value determines which character set flows across the network during transaction routing. The correct code page depends on the national language of your local region and the remote system type. See Chapter 7, "Data conversion," on page 147 for information about how to choose this value. An example configuration window is shown in Figure 35.


IBM TXSeries - CICSWINT - Communication Definition - Untitled
General
SNA
TCP/IP
Security
Permanent
Both
Reset
Cancel
Help

SYSID: COS2

Description: To CICSOS2 region

Group:

☑ Activate on startup

□ Protect resource

☑ In service

Connection type: CICS Local SNA

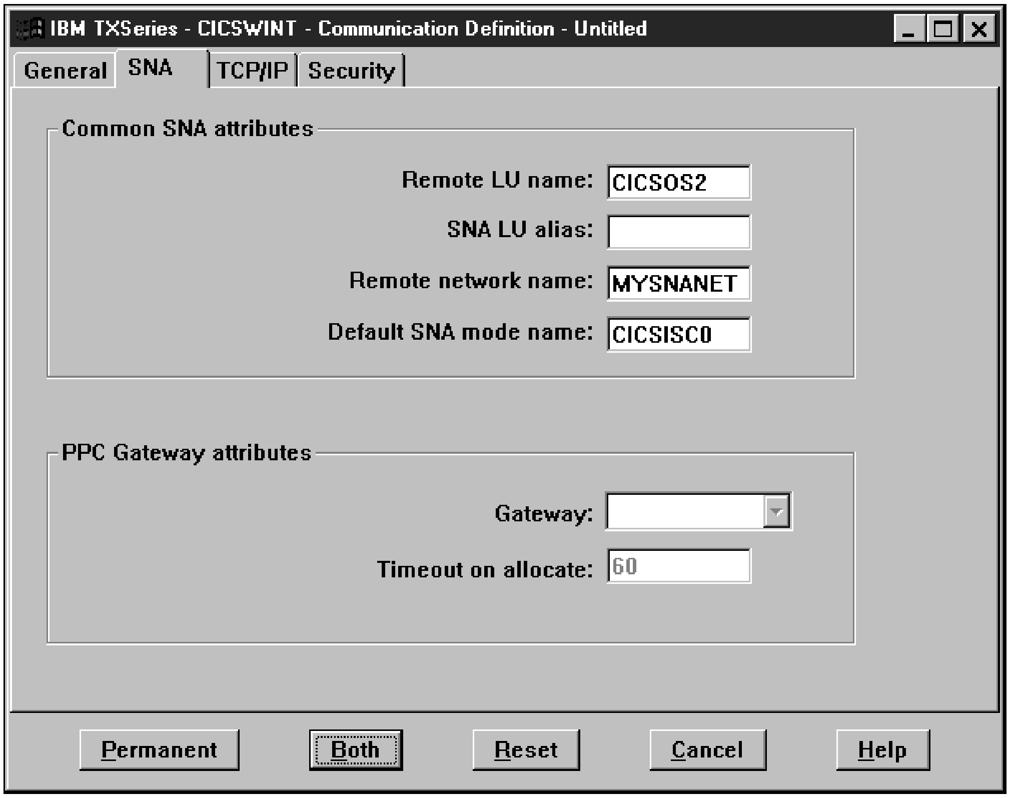
Code page for transaction routing: IBM-850

Number of updates: 0

Figure 35. Communication Definition window: General tab

5. Click the **SNA** tab. On this tab:
   a. Type the LU name of the remote system in the **Remote LU name** field.
   b. Optionally, in the **SNA LU alias** field, type the LU alias that is configured in your local SNA product for the remote system. (In this example, this field remains empty.)
   c. Type the SNA network to which the remote system is connected in the **Remote network name** field.
   d. In the **Default SNA mode name** field, type the name of the SNA modegroup that is to be used for intersystem requests when an SNA modename is not specified in either the PROFILE option of the EXEC CICS

ALLOCATE command or in the **SNAModeName** attribute of the Transaction Definitions (TD) entry. Refer to “Default modenames” on page 110 for more information about the use of this attribute. An example configuration window is shown in Figure 36.


IBM TXSeries - CICSWINT - Communication Definition - Untitled

Common SNA attributes

Remote LU name: CICSOS2

SNA LU alias:

Remote network name: MYSNANET

Default SNA mode name: CICSISC0

PPC Gateway attributes

Gateway:

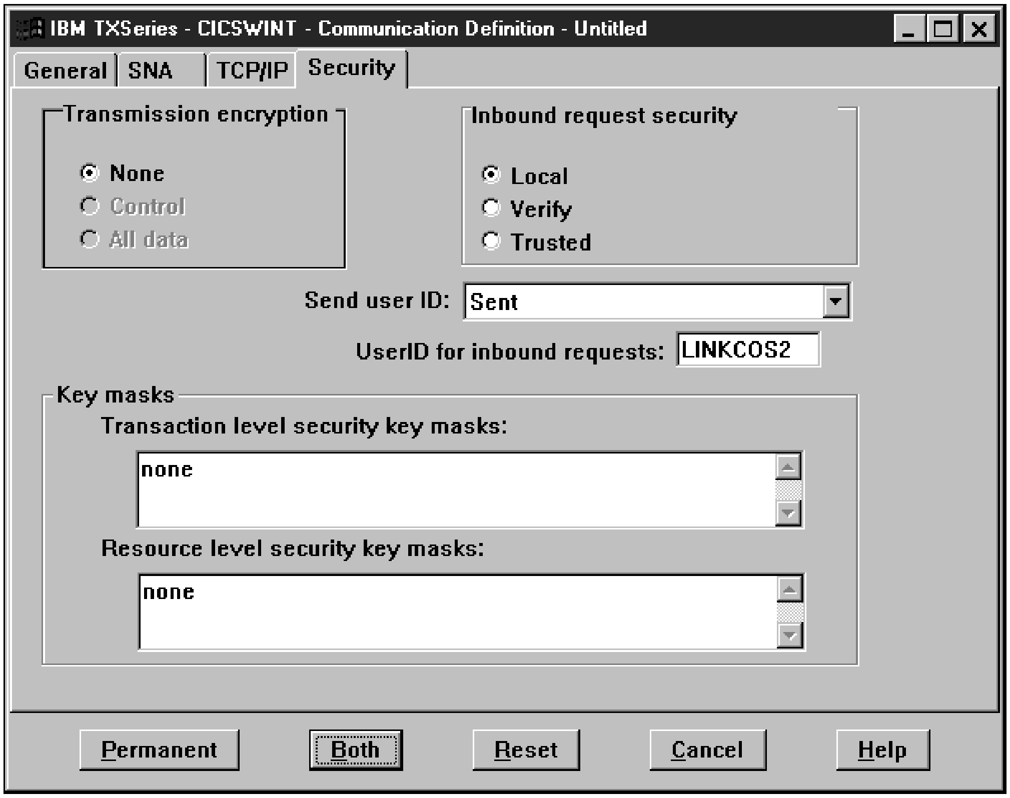
Timeout on allocate: 60

Figure 36. Communication Definition window: SNA tab

6. Click the **Security** tab. These attributes describe the security checks that are applied to all intersystem requests that use this CD entry. On this tab:
   a. In the **Inbound request security** area, click the type of security that you require:
      - Accepting the default **Local** option causes CICS to run all incoming intersystem requests from the remote system under the user ID value that you specify in the **UserID for inbound requests** field. The User Definitions (UD) entry for this user ID determines which resources these intersystem requests can access. This type of security is called *link security*, and is described in “CICS link security” on page 126.
      - Choosing the **Verify** or **Trusted** options causes CICS to use the security information (such as the user ID and password) that is sent with the intersystem request. CICS also uses the user ID that you specify in the **UserID for inbound requests** field to restrict the resources that inbound intersystem requests can access. This type of security is called *user security* and is described in “CICS user security” on page 127.

   b. Security information that is sent with outbound requests is determined by the value that is selected in the **Send user ID** field. “Setting up a CICS region to flow user IDs” on page 130 and “Setting up a CICS region to flow passwords” on page 131 describe how this option works.

   c. See Chapter 6, “Configuring intersystem security,” on page 123 for more information about how CICS security is configured, including descriptions of the **Transaction level security key masks** and **Resource level security key masks** fields. An example configuration window is shown in Figure 37.


IBM TXSeries - CICSWINT - Communication Definition - Untitled
General
SNA
TCP/IP
Security
Permanent
Both
Reset
Cancel
Help

Transmission encryption

● None
○ Control
○ All data

Inbound request security

● Local
○ Verify
○ Trusted

Send user ID: Sent

UserID for inbound requests: LINKCOS2

Key masks

Transaction level security key masks: none

Resource level security key masks: none

Figure 37. Communication Definition window: Security tab

7. Click the **Permanent** button to add the Communications Definitions (CD) entry to the permanent database, or click the **Both** button to add the entry to the permanent and runtime databases.
8. Repeat this process for any other Communications Definitions (CD) entries.

Configuring CICS for local SNA support by using SMIT (CICS for AIX only)

To configure CICS for local SNA by using SMIT, perform the following procedure (assume that default values are accepted for any attributes that are not discussed):

1. Ensure that you are logged into AIX with enough privileges to change the region database. (For example, log into AIX as the **root** user.)
2. Optionally, set the environment variable CICSREGION to the name of your CICS region. For example:

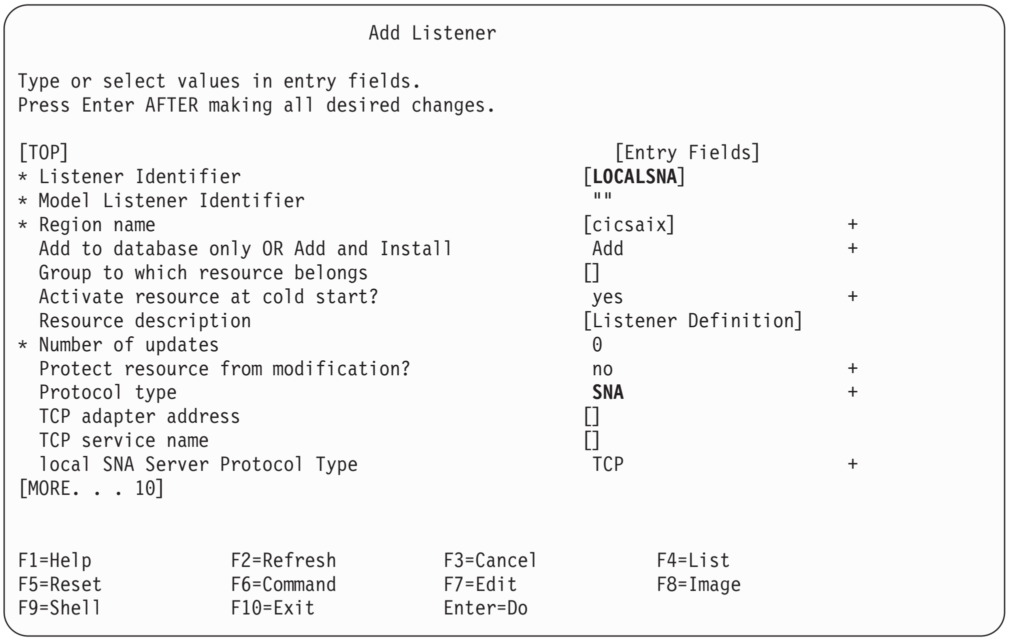
   ```
   export CICSREGION=cicsaix
   ```

   Note: In this procedure, the local region is called `cicsaix` to reflect a CICS for AIX system. Also, this example assumes that you are using the Korn shell; if you are using a different shell, change the **export** command accordingly.
3. Enter `smitty cicsregion` to start SMIT.
4. If you have not set the CICSREGION environment variable as described in step 2, select the option **Change Working CICS Region**.

5. Select your CICS region from the list that is displayed, and press the Enter key. The COMMAND STATUS screen verifies your selection.
6. Press the F3 key.
7. Configure a Listener Definitions (LD) entry by performing the following procedure:
   a. Select options:

      ```
      ► Define Resources for a CICS Region
          ► Manage Resource(s)
              ► Listeners
                  ► Add New
      ```

      The Add Listener panel is displayed.
   b. Enter a value for the **Model Listener Identifier** attribute. Use the name of an LD entry that you have defined previously, or press the Enter key to use the default value ("").
   c. Enter the name of the LD entry in the **Listener Identifier** field.
   d. Add a description in the **Resource description** field if desired.
   e. Select the **SNA** option from the **Protocol type** field. Figure 38 shows an example Add Listener SMIT panel. (Only the panel that shows updated attributes is shown.)



```
                                 Add Listener

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
* Listener Identifier                                [LOCALSNA]
* Model Listener Identifier                           ""
* Region name                                        [cicsaix]              +
  Add to database only OR Add and Install             Add                   +
  Group to which resource belongs                    []
  Activate resource at cold start?                    yes                   +
  Resource description                               [Listener Definition]
* Number of updates                                   0
  Protect resource from modification?                 no                    +
  Protocol type                                       SNA                   +
  TCP adapter address                                []
  TCP service name                                   []
  local SNA Server Protocol Type                      TCP                   +
[MORE. . . 10]


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 38. Add Listener SMIT panel

   f. Press the Enter key to create the LD entry. (If you add an LD entry to a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the entry is being added only to the permanent database, the region would have to be cold-started.) The COMMAND STATUS screen verifies successful creation of the definition.
   g. Press the F3 key three times to return to the `Manage Resource(s)` menu.
8. Update the **Region system identifier (short name)**, **Network name to which local region is attached**, and **Local LU name** Region Definitions (RD) attributes, by performing the following procedure:
   a. From the `Manage Resource(s)` menu, select the options:

```
► Region
    ► Show/Change
```

The Show/Change Region panel is displayed.

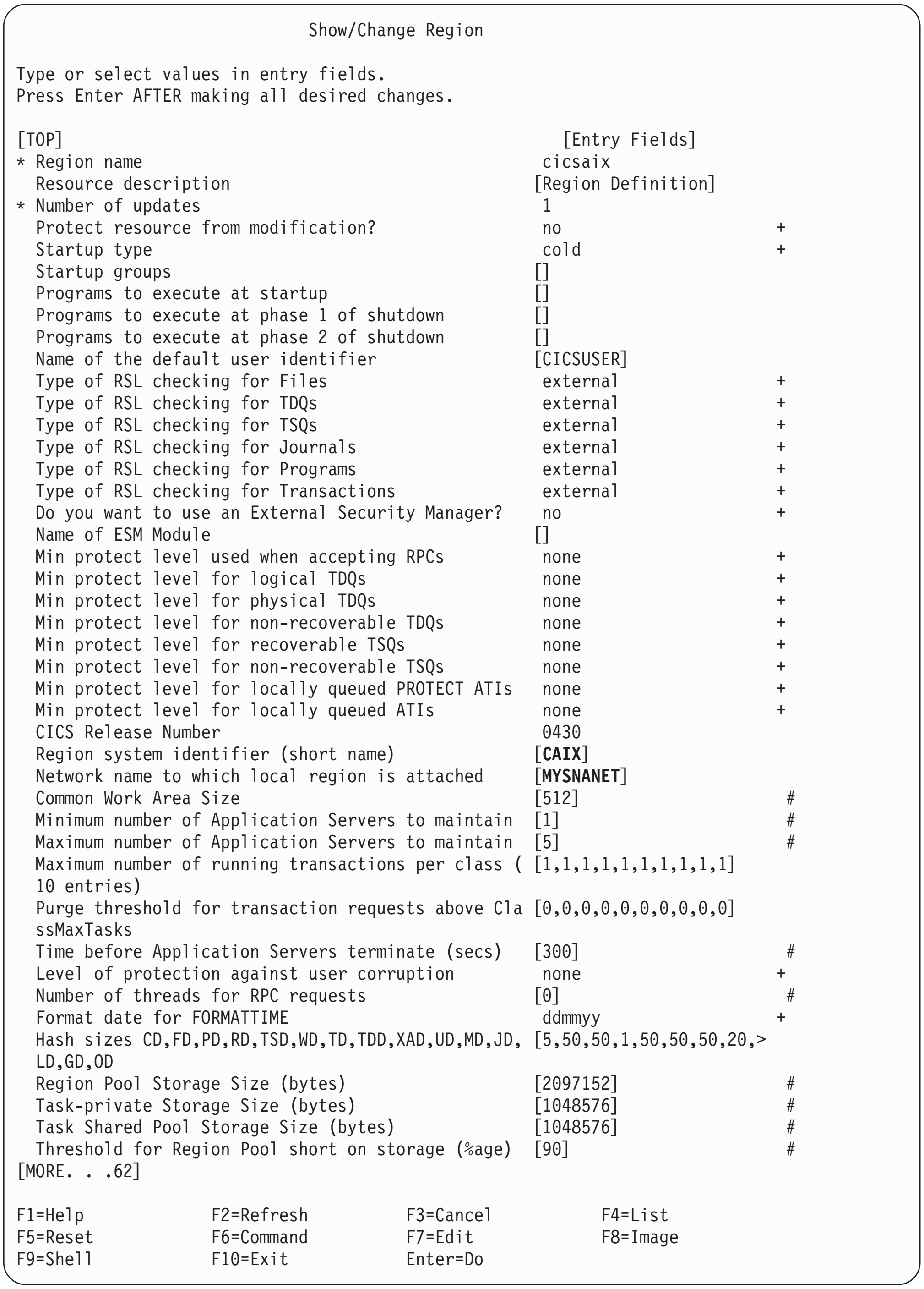
b. Enter a description in the **Resource description** field if desired.

c. Enter a four-character name for the CICS region, known as the SYSID, in the **Region system identifier (short name)** field.

d. Enter the network name in the **Network name to which local region is attached** field.

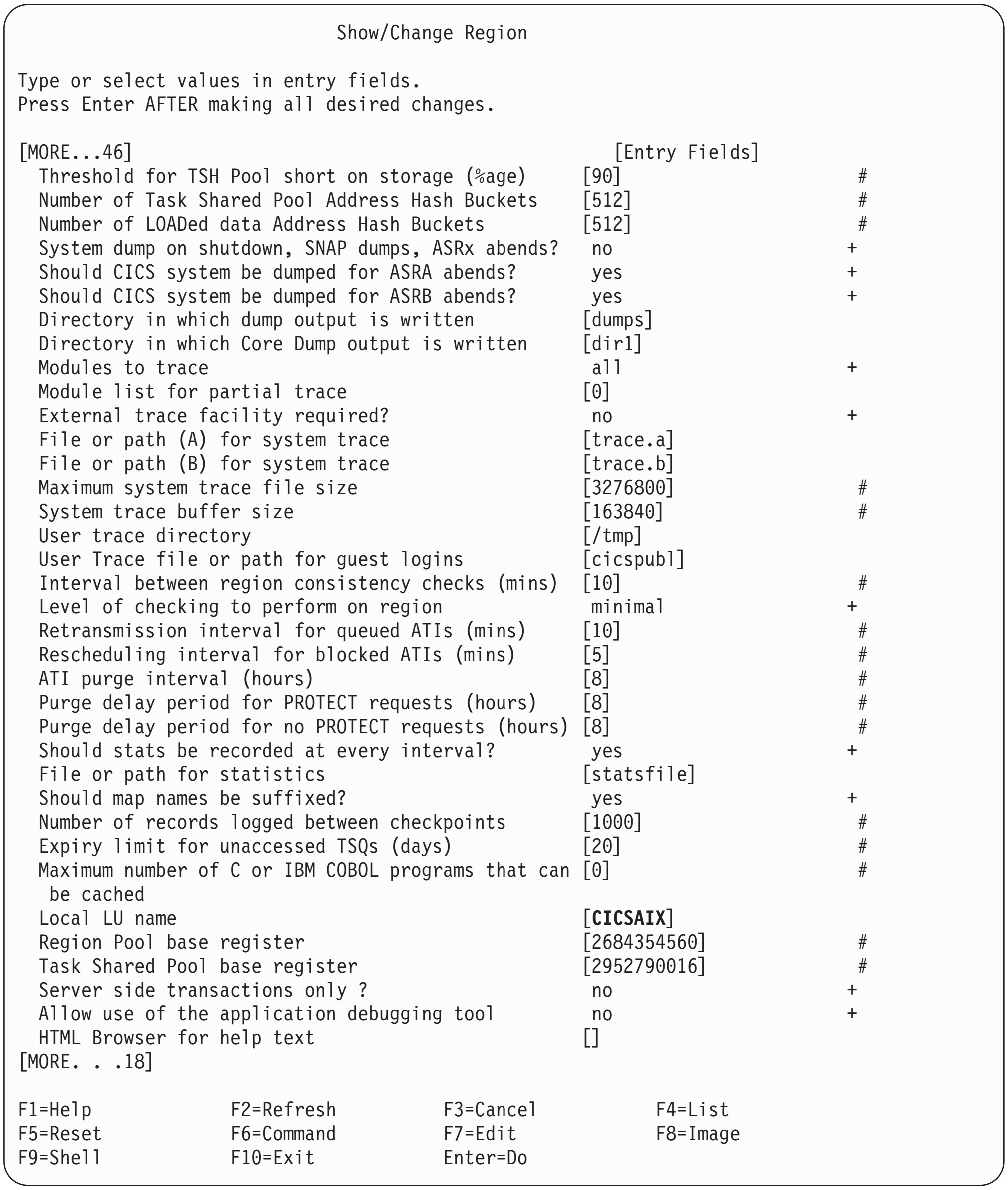
e. Enter the region's local LU name in the **Local LU name** field.

Figure 39 on page 85 and Figure 40 on page 86 show example Show/Change Region SMIT panels. (Only those panels showing updated attributes are shown.)



```
                              Show/Change Region

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
* Region name                                         cicsaix
  Resource description                               [Region Definition]
* Number of updates                                   1
  Protect resource from modification?                 no                      +
  Startup type                                        cold                    +
  Startup groups                                     []
  Programs to execute at startup                     []
  Programs to execute at phase 1 of shutdown         []
  Programs to execute at phase 2 of shutdown         []
  Name of the default user identifier                [CICSUSER]
  Type of RSL checking for Files                      external                +
  Type of RSL checking for TDQs                       external                +
  Type of RSL checking for TSQs                       external                +
  Type of RSL checking for Journals                   external                +
  Type of RSL checking for Programs                   external                +
  Type of RSL checking for Transactions               external                +
  Do you want to use an External Security Manager?    no                      +
  Name of ESM Module                                 []
  Min protect level used when accepting RPCs          none                    +
  Min protect level for logical TDQs                  none                    +
  Min protect level for physical TDQs                 none                    +
  Min protect level for non-recoverable TDQs          none                    +
  Min protect level for recoverable TSQs              none                    +
  Min protect level for non-recoverable TSQs          none                    +
  Min protect level for locally queued PROTECT ATIs   none                    +
  Min protect level for locally queued ATIs           none                    +
  CICS Release Number                                 0430
  Region system identifier (short name)              [CAIX]
  Network name to which local region is attached     [MYSNANET]
  Common Work Area Size                              [512]                     #
  Minimum number of Application Servers to maintain  [1]                       #
  Maximum number of Application Servers to maintain  [5]                       #
  Maximum number of running transactions per class ( [1,1,1,1,1,1,1,1,1,1]
  10 entries)
  Purge threshold for transaction requests above Cla [0,0,0,0,0,0,0,0,0,0]
  ssMaxTasks
  Time before Application Servers terminate (secs)   [300]                     #
  Level of protection against user corruption         none                    +
  Number of threads for RPC requests                 [0]                       #
  Format date for FORMATTIME                          ddmmyy                  +
  Hash sizes CD,FD,PD,RD,TSD,WD,TD,TDD,XAD,UD,MD,JD, [5,50,50,1,50,50,50,20,>
  LD,GD,OD
  Region Pool Storage Size (bytes)                   [2097152]                 #
  Task-private Storage Size (bytes)                  [1048576]                 #
  Task Shared Pool Storage Size (bytes)              [1048576]                 #
  Threshold for Region Pool short on storage (%age)  [90]                      #
[MORE. . .62]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 39. Show/Change Region SMIT panel (Part 1)



```
                              Show/Change Region

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[MORE...46]                                              [Entry Fields]
  Threshold for TSH Pool short on storage (%age)     [90]                    #
  Number of Task Shared Pool Address Hash Buckets    [512]                   #
  Number of LOADed data Address Hash Buckets         [512]                   #
  System dump on shutdown, SNAP dumps, ASRx abends?   no                    +
  Should CICS system be dumped for ASRA abends?       yes                   +
  Should CICS system be dumped for ASRB abends?       yes                   +
  Directory in which dump output is written          [dumps]
  Directory in which Core Dump output is written     [dir1]
  Modules to trace                                    all                   +
  Module list for partial trace                      [0]
  External trace facility required?                   no                    +
  File or path (A) for system trace                  [trace.a]
  File or path (B) for system trace                  [trace.b]
  Maximum system trace file size                     [3276800]               #
  System trace buffer size                           [163840]                #
  User trace directory                               [/tmp]
  User Trace file or path for guest logins           [cicspubl]
  Interval between region consistency checks (mins)  [10]                    #
  Level of checking to perform on region              minimal               +
  Retransmission interval for queued ATIs (mins)     [10]                    #
  Rescheduling interval for blocked ATIs (mins)      [5]                     #
  ATI purge interval (hours)                         [8]                     #
  Purge delay period for PROTECT requests (hours)    [8]                     #
  Purge delay period for no PROTECT requests (hours) [8]                     #
  Should stats be recorded at every interval?         yes                   +
  File or path for statistics                        [statsfile]
  Should map names be suffixed?                       yes                   +
  Number of records logged between checkpoints       [1000]                  #
  Expiry limit for unaccessed TSQs (days)            [20]                    #
  Maximum number of C or IBM COBOL programs that can [0]                     #
   be cached
  Local LU name                                      [CICSAIX]
  Region Pool base register                          [2684354560]            #
  Task Shared Pool base register                     [2952790016]            #
  Server side transactions only ?                     no                    +
  Allow use of the application debugging tool         no                    +
  HTML Browser for help text                         []
[MORE. . .18]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 40. Show/Change Region SMIT panel (Part 2)

f. Press Enter to update the Region Definitions (RD) attributes. (If you update RD attributes in a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the attributes are being updated only in the permanent database, the region would have to be cold-started.) The COMMAND STATUS screen confirms successful completion of the process.

g. Press the F3 key three times to return to the `Manage Resource(s)` menu.

9. Configure a Communications Definitions (CD) entry for each remote system with which your system is to communicate, by performing the following procedure:

   a. Select options:

      ► `Communications`
         ► `Add New`

      The Add Communication panel is displayed.

b. Enter a value for the **Model Communication Identifier** attribute. Use the name of a CD entry that you have defined previously, or press the Enter key to use the default value ("").

c. Enter the name of the CD entry in the **Communication Identifier** field.

d. Add a description in the **Resource description** field if desired.

e. Select **local_sna** in the **Connection type** field.

f. Enter the LU name of the remote system in the **Name of remote system** field.

g. Enter the name of the network to which the remote system is attached in the **SNA network name for the remote system** field.

h. In the **Default modename for a SNA connection** field, enter the name of the SNA modegroup that is to be used for intersystem requests when an SNA modename is not specified either in the PROFILE option of the EXEC CICS ALLOCATE command, or in the **SNAModeName** attribute of the Transaction Definitions (TD) entry. If the **Default modename for a SNA connection** attribute is set to "" (its default value), and the transaction has not provided a modename, the modename that is configured in the AIX SNA communications product's side information profile for the local LU name is used. Refer to "Default modenames" on page 110 for more information about the use of this attribute.

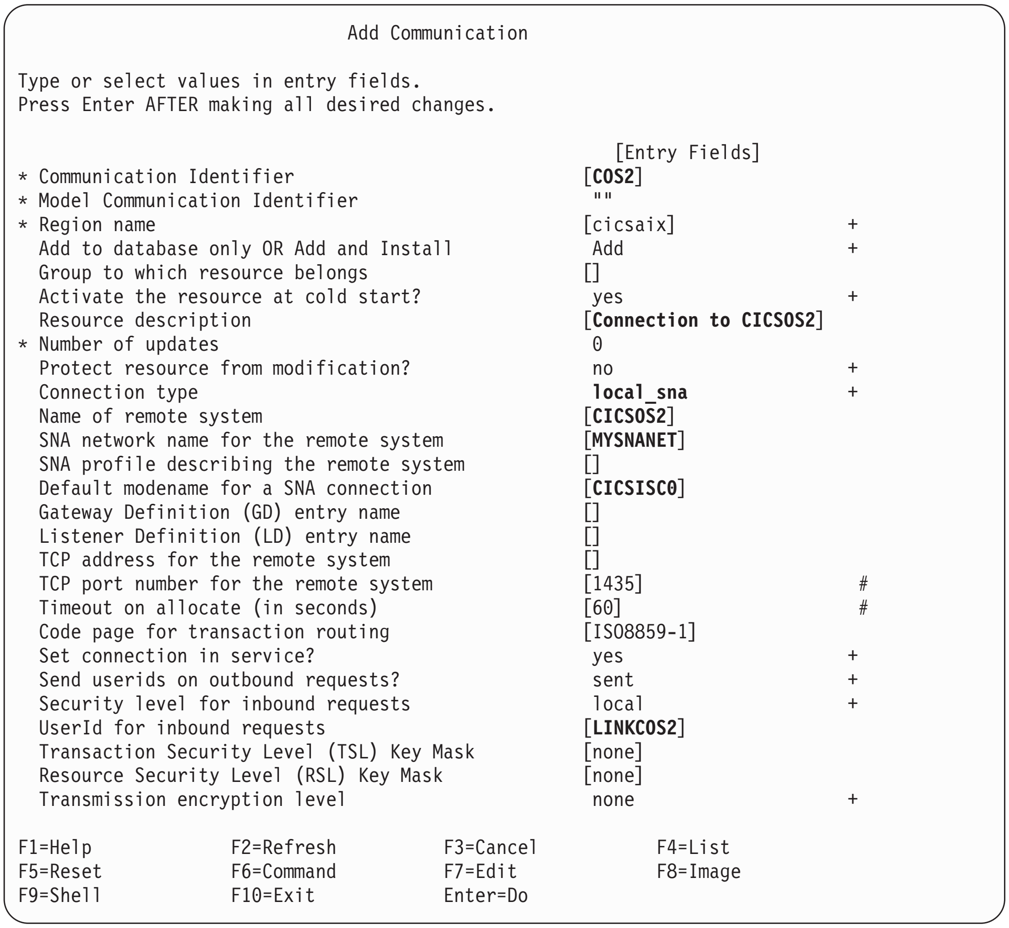
i. Accept the default value or enter the appropriate value in the **Code page for transaction routing** field. This value determines which character set flows across the network during transaction routing. The correct code page depends on the national language of your local region and the remote system type. See "Data conversion for transaction routing" on page 164 for information about how to choose this value.

j. Accept the default value or select the appropriate value for the **Send userids on outbound requests?** attribute to determine the type of security information (if any) that is sent with outbound requests. "Setting up a CICS region to flow user IDs" on page 130 and "Setting up a CICS region to flow passwords" on page 131 describe how this attribute works.

k. In the **Security level for inbound requests** field, select the type of security that you require:

   - Accepting the default **local** option causes CICS to run all incoming intersystem requests from the remote system under the user ID value that you specify in the **UserID for inbound requests** field. The User Definitions (UD) entry for this user ID determines which resources these intersystem requests can access. This type of security is called *link security* and is described in "CICS link security" on page 126.
   - Choosing the **verify** or **trusted** options causes CICS to use the security information (such as the user ID and password) that is sent with the intersystem request. CICS also uses the user ID that you specify in the **UserID for inbound requests** field to restrict the resources that inbound intersystem requests can access. This type of security is called *user security* and is described in "CICS user security" on page 127.

l. See Chapter 6, "Configuring intersystem security," on page 123 for more information about how CICS security is configured, including descriptions of the **Transaction Security Level (TSL) Key Mask** and **Resource Security Level (RSL) Key Mask** attributes. Figure 41 on page 88 shows an example Add Communication SMIT panel.



```
                              Add Communication

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
* Communication Identifier                            [COS2]
* Model Communication Identifier                       ""
* Region name                                         [cicsaix]               +
  Add to database only OR Add and Install              Add                    +
  Group to which resource belongs                     []
  Activate the resource at cold start?                 yes                    +
  Resource description                                [Connection to CICSOS2]
* Number of updates                                    0
  Protect resource from modification?                  no                     +
  Connection type                                      local_sna              +
  Name of remote system                               [CICSOS2]
  SNA network name for the remote system              [MYSNANET]
  SNA profile describing the remote system            []
  Default modename for a SNA connection               [CICSISC0]
  Gateway Definition (GD) entry name                  []
  Listener Definition (LD) entry name                 []
  TCP address for the remote system                   []
  TCP port number for the remote system               [1435]                   #
  Timeout on allocate (in seconds)                    [60]                     #
  Code page for transaction routing                   [ISO8859-1]
  Set connection in service?                           yes                    +
  Send userids on outbound requests?                   sent                   +
  Security level for inbound requests                  local                  +
  UserId for inbound requests                         [LINKCOS2]
  Transaction Security Level (TSL) Key Mask           [none]
  Resource Security Level (RSL) Key Mask              [none]
  Transmission encryption level                        none                   +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 41. Add Communication SMIT panel

m. Press Enter to create the CD entry. (If you add a CD entry to a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the entry is being added only to the permanent database, the region would have to be cold-started.) The COMMAND STATUS screen verifies the definition creation.

n. Press the F3 key three times to return to the `Manage Resource(s)` menu.

o. Repeat this process for any other Communications Definitions (CD) entries.

SNA Receive Timeout feature (For CICS on AIX only)

A timeout process now exists for use by SNA receive processes. A CICS application server that is waiting on a SNA receive from a remote CICS region can deallocate the conversation and force an abend in the CICS Application Server (cicsas) process. This abend triggers a controlled shutdown of the application server and the automatic startup of a new one. The newly started application server is then available to run any of the scheduled transactions that are in the queue. Abend message **AB61** describes the expiration of the SNA timeout period:

```
AB61
EXPLANATION: Local transaction aborted due to SNA timeout waiting on receive.
SYSTEM_ACTION: CICS abnormally terminates the application server.
USER_RESPONSE: Either reissue the transaction or increase the receive timeout
value and reissue the transaction.
```

The message that is associated with this abend is **ERZ59999E**. It states that the time-out period has expired. In this example, 'TRAN' is the local transaction name, 'n' is the specified timeout value, and 'SYS1' is the system definition to which the transaction was communicating:

```
ERZ59999E
"Transaction 'TRAN' has exceeded the allowed timeout='n' seconds on SNA
session to SysId='SYS1'"
EXPLANATION: The remote region with this SysID has not replied to a request
sent over SNA within the specified receive timeout value.
SYSTEM_ACTION: The local transaction will be abended with AB61 after the
SNA conversation has been deallocated with the remote region.
USER_RESPONSE: Retry the same transaction later or increase the timeout
value and retry.
Possible MSN values: 0097
```

When a region is configured for SNA Receive Timeout, conversations that are established to a remote host for Distributed Program Link (DPL), Distributed Transaction Processing (DTP), Transaction Routing, and CRTE hosted transactions are subject to it.

Configuration is through the environment variable **CICS_SNA_RECEIVE_TIMEOUT**. The contents of the string in this environment variable trigger and control the timing-out of SNA conversations to the specified systems in the string. To set this environment variable, add the following line to the environment file that is in the /var/cics_regions/ *region_name* directory:

```
CICS_SNA_RECEIVE_TIMEOUT="SYS1,N1,SYS2,N2,SYS3,N3,SYS5,N5,. . .SYSn,Nn"
```

where SYS1 to SYSn are the Communications Definitions in the CD stanza of the CICS region that are subject to the forced timeout. Each of these is a remote system identifier with a maximum size of four alphanumeric characters. N1 through Nn are the actual timeout values in seconds. Each is the timeout value for the preceding system identifier, and is an integer value greater than, or equal to, 0.

The environment variable string cannot exceed 1024 characters in length, and must strictly follow the above format with a comma "," separating each timeout value from its preceding system identifier. After it is configured, the region must be cold started to allow the new timeout values to be associated with the system definitions in the CD stanza.

Configuring CICS for PPC Gateway server SNA support

PPC Gateway server SNA support lets TXSeries for Multiplatforms regions communicate with SNA systems with synchronization level 2 support. (For descriptions of synchronization levels, see “Ensuring data integrity with synchronization support” on page 16.) The CICS region is connected to a TCP/IP network through the PPC Gateway server, which provides a link to the SNA network.

Note: For information about how to create a PPC Gateway server, refer to Chapter 8, “Creating and using a PPC Gateway server in a CICS environment,” on page 175.

PPC Gateway server SNA support requires that an appropriate SNA product be installed and configured on the same machine as is the PPC Gateway server. Such products are shown in Table 21 on page 90.

Table 21. Communications products for various platforms to enable PPC Gateway server SNA support

Platform	Communications product
Windows	IBM Communications Server or Microsoft SNA Server
AIX	IBM Communications Server
Solaris	SNAP-IX
HP-UX	SNAplus2

The PPC Gateway server can be on the same machine as is the CICS region, or on a different machine.

Note: CICS does not support the use of a PPC Gateway server on Solaris. However, a CICS for Solaris region can use a PPC Gateway server that is running on a non-Solaris platform.

A CICS region can use more than one PPC Gateway server. This type of configuration provides the following performance benefits:

- Network links from many remote machines are spread across more than one gateway machine, and, as a result, across multiple SNA products.
- If one PPC Gateway server fails, another is available as a backup.
- The processing load is spread across more than one PPC Gateway server.

A PPC Gateway server can also be shared by several CICS regions. This can be an economical alternative if your CICS regions do not make many SNA intercommunication requests. However, such a configuration can overload a PPC Gateway server and make problem determination difficult.

To enable your CICS region to use a PPC Gateway server, you must configure the following:

- A Gateway Definitions (GD) entry for each PPC Gateway server that the region uses. A GD entry is a one- through four-character name, whose attributes describe:
  – The name of the PPC Gateway server (specified in the **GatewayCDSName** attribute).
  – The SNA LU name that the PPC Gateway server is to use for the region (specified in the **GatewayLUName** attribute).

  Note: If a **GatewayLUName** value is not specified, CICS uses the **LocalLUName** attribute value as the region identifier for remote systems. However, the same **LocalLUName** attribute value cannot be shared between local SNA support and a PPC Gateway server or among multiple PPC Gateway servers. Be sure to create a unique Gateway Definitions (GD) entry for each system that is connecting through a PPC Gateway server.
- A Communications Definitions (CD) entry for each remote system with which the region is to communicate.

Note: Although no Region Definitions (RD) attributes must be configured to enable PPC Gateway server SNA communications, it is recommended that you specify a short name for the local region by using the **LocalSysId** attribute.

You also possibly need to configure the Transaction Definitions (TD) **TPNSNAProfile** attribute if both the following conditions apply:

- The local CICS region is a CICS for AIX region.
- Remote systems will request transactions that reside in the local region.

See the *TXSeries for Multiplatforms Administration Reference* for more information.

A Listener Definitions (LD) entry is not required because the PPC Gateway server sends requests to the region by using CICSIPC Remote Procedure Calls (RPCs). RPCs flow through the RPC Listener process, which is always present in a CICS region.

Figure 42 on page 92 shows the key configuration attributes. It shows a CICS region with a Gateway LU name of `CICSNTGW` connected through a PPC Gateway server to a remote system that is named `CICSESA`.

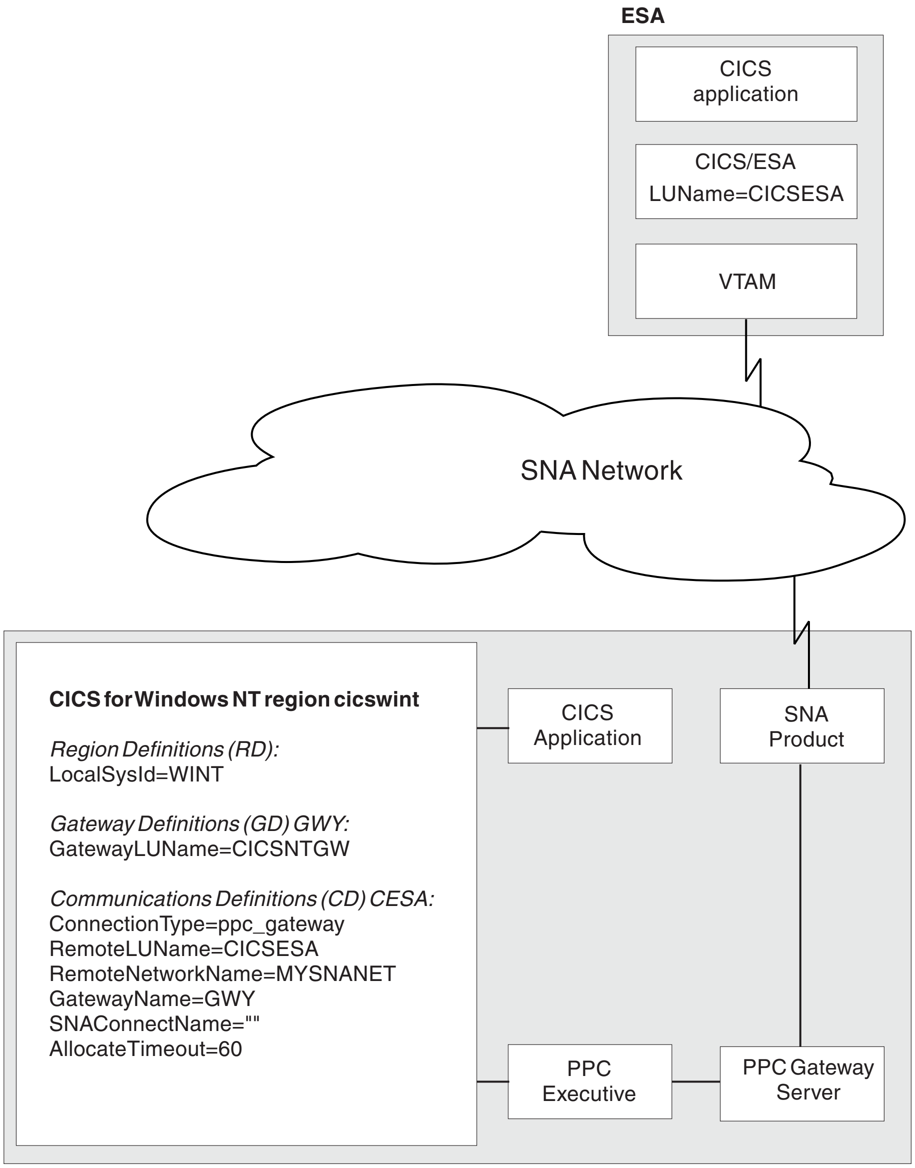


Figure 42. An example network using PPC Gateway server SNA support

The CICS region has a Region Definitions (RD) entry with the **LocalSysId** attribute identifying the short name for the local region. It has a Gateway Definitions (GD) entry with the **GatewayLUName** attribute set to `CICSNTGW`, which is the LU name that the PPC Gateway server uses to communicate with the local region. It also has a Communications Definitions (CD) entry called `CESA`, which defines the remote CICS/ESA system to the local region. The Communications Definitions (CD) **ConnectionType** attribute identifies the connection as a `ppc_gateway`, the **RemoteLUName** attribute defines the LU name of the remote region, and the **RemoteNetworkName** attribute defines the name of the network to which the remote region belongs. The **GatewayName** attribute is set to `GWY`, which is the name of the Gateway Definitions (GD) entry for the connection, the **SNAConnectName** attribute is left blank (`""`), which indicates that no alias exists for the partner LU name in the remote system, and the **AllocateTimeout** attribute is set to `60`, which indicates the wait time in seconds for an intersystem request to be started in the PPC Gateway server.

Figure 42 on page 92 shows a few of the resource definition attributes that can be configured for SNA communications. For more complete lists of attributes commonly used in communications, see Table 22, Table 23 on page 94, Table 24 on page 95, and Table 25 on page 95. They list and define the resource definition attributes that are used for PPC Gateway server SNA communications that are covered in this chapter. Command-line attribute names are given, along with their equivalent names in the IBM TXSeries Administration Tool and SMIT. The *TXSeries for Multiplatforms Administration Reference* and the *CICS Administration Guide* provide complete information about all CICS commands and definitions.

Note: The terminology in this chapter uses the attribute names of the CICS resource definitions that are referenced when CICS commands are used to configure a region. If you are using CICS on a Windows platform, you can use the IBM TXSeries Administration Tool rather instead of CICS commands to configure the resources. Likewise, if you are using CICS for AIX, you can use SMIT to configure the resources.

Table 22. Comparison of CICS Communications Definitions (CD) for PPC Gateway server SNA connections

Communications Definitions (CD) attributes	Equivalent IBM TXSeries Administration Tool resource definition attributes	Equivalent SMIT resource definition attributes	Use in PPC Gateway server connections
<Key>	SYSID	Communication Identifier	Specifies the name of the Communications Definitions (CD) entry.
AllocateTimeout	Timeout on allocate	Timeout on allocate (in seconds)	Specifies the wait time in seconds for an intersystem request to be started in the PPC Gateway server.
ConnectionType	Connection type	Connection type	Specifies the type of connection.
DefaultSNAModeName	Default SNA mode name	Default modename for a SNA connection	Specifies the SNA modegroup that is to be used for intersystem requests when an SNA modename is not specified in either the PROFILE option of the EXEC CICS ALLOCATE command or in the **SNAModeName** attribute of the Transaction Definitions (TD).
GatewayName	Gateway	Gateway Definition (GD) entry name	Specifies the name of a locally defined Gateway Definitions (GD) entry for the PPC Gateway server.
LinkUserId	UserID for inbound requests	UserId for inbound requests	Specifies a locally defined user ID to associate with inbound requests.
OutboundUserIds	Sent user ID	Send user IDs on outbound requests?	Specifies whether a user ID with or without a password is sent with outbound requests.

Table 22. Comparison of CICS Communications Definitions (CD) for PPC Gateway server SNA connections (continued)

Communications Definitions (CD) attributes	Equivalent IBM TXSeries Administration Tool resource definition attributes	Equivalent SMIT resource definition attributes	Use in PPC Gateway server connections
RemoteCodePageTR	Code page for transaction routing	Code page for transaction routing	Specifies the code page for transaction-routing data that is flowing between the local and remote regions.
RemoteLUName	Remote LU name	Name of remote system	Specifies the LU name of the remote system.
RemoteNetworkName	Remote network name	SNA network name for the remote system	Specifies the network name of the remote system.
RemoteSysSecurity	Inbound request security	Security level for inbound requests	Specifies how CICS is to process security information that is received with inbound requests.
RSLKeyMask	Resource level security key masks	Resource Security Level (RSL) Key Mask	Contains the list of resource link security keys that CICS uses to control access to resources from transactions.
SNAConnectName	SNA LU alias	SNA profile describing the remote system	Specifies the name of the remote system's partner LU alias.
TSLKeyMask	Transaction level security key masks	Transaction Security Level (TSL) Key Mask	Contains the list of transaction link security keys that CICS uses to control access to transactions.

Table 23. Comparison of CICS Gateway Definitions (GD) for PPC Gateway server SNA connections

Gateway Definitions (GD) attributes	Equivalent IBM TXSeries Administration Tool resource definition attributes	Equivalent SMIT resource definition attributes	Use in PPC Gateway server connections
<Key>	Gateway name	Gateway Identifier	Specifies the name of the Gateway Definitions (GD) entry. At least one entry must be defined that specifies the PPC Gateway server that the CICS region is to use for the connection.
GatewayLUName	Gateway LU name	SNA LU name of the gateway	Specifies the SNA LU name that the PPC Gateway server uses to communicate with the local region.

Table 24. Comparison of CICS Region Definitions (RD) for PPC Gateway server SNA connections

Region Definitions (RD) attribute	Equivalent IBM TXSeries Administration Tool resource definition attribute	Equivalent SMIT resource definition attribute	Use in PPC Gateway server connections
LocalSysId	Local SYSID	Region system identifier (short name)	Specifies the short name of the CICS region (as used in the SYSID option of several CICS commands).

Table 25. Comparison of CICS Transaction Definitions (TD) for PPC Gateway server SNA connections

Transaction Definitions (TD) attributes	Equivalent IBM TXSeries Administration Tool resource definition attributes	Equivalent SMIT resource definition attributes	Use in PPC Gateway server connections
SNAModeName	SNA mode name	SNA modename for this transaction	Specifies the modename on an allocate request to a remote system that is connected by SNA.
TPNSNAProfile	Not applicable	SNA TPN profile for APPC listener program	**(AIX SNA only)** Specifies the name of an AIX SNA TPN profile that is configured in an SNA communications product, for example, `CICSTPN`.

For information about how to configure these attributes, see the following sections. Each section presents an example PPC Gateway server SNA configuration:

- "Configuring CICS for PPC Gateway server SNA support from the command line" configures a local CICS for Windows region to communicate with a remote CICS/ESA region.
- "Configuring CICS for PPC Gateway server SNA support by using the IBM TXSeries Administration Tool (CICS on Windows platforms only)" on page 98 configures a local CICS for Windows region to communicate with a remote CICS/ESA region.
- "Configuring CICS for PPC Gateway server SNA support by using SMIT (CICS for AIX only)" on page 103 configures a local CICS for AIX region to communicate with a remote CICS/ESA region.

Many attribute values must match associated values that are used in related communications products, such as IBM Communications Server for Windows or IBM Communications Server for AIX. See the communications product's administration documentation for more information.

Configuring CICS for PPC Gateway server SNA support from the command line

This section uses the CICS commands **cicsadd** and **cicsupdate** to add or update intercommunication resources. For more information about the use of these commands, see the *TXSeries for Multiplatforms Administration Reference*.

To configure CICS for PPC Gateway server SNA, perform the following procedure (assume that default values are accepted for any attributes that are not discussed):

1. Configure a Gateway Definitions (GD) entry by issuing the **cicsadd** command, as shown in the following example:

   ```
   cicsadd -r cicswint -P -c gd GWY GatewayCDSName="cicsgwy"  \
           GatewayLUName="CICSNTGW"
   ```

   In this example, a Gateway Definitions (GD) entry called GWY is added to the permanent database of region cicswint. (If you add a GD to a region while it is running, PPC Gateway server support is not enabled until the region is reinitialized by either an autostart or a cold start. In this example, bcause the entry is being added only to the permanent database, the region would have to be cold-started.)

   The **GatewayCDSName** attribute is set to cicsgwy. If the value for the **GatewayCDSName** attribute does not begin with a slash character (/), CICS inserts the string **/.:/cics/ppc/gateway** in front of the supplied name. Therefore, the GD entry that GWY describes is a PPC Gateway server named: **/.:/cics/ppc/gateway/cicsgwy**.

   Because the **GatewayLUName** attribute is set to CICSNTGW, CICS notifies the PPC Gateway server to use the LU name CICSNTGW to communicate with the local region.

2. Update the **LocalSysId** Region Definitions (RD) attribute by issuing the **cicsupdate** command, as shown in the following example:

   ```
   cicsupdate -r cicswint -P -c rd LocalSysId="WINT"
   ```

   In this example, the Region Definitions (RD) attribute **LocalSysId** is set to WINT. This value is added to the permanent database of region cicswint. (If you update an RD attribute in a region while it is running, the change is not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the attribute is being updated only in the permanent database, the region would have to be cold-started.)

3. Configure a Communications Definitions (CD) entry for each remote system with which the local system is to communicate by issuing the **cicsadd** command, as shown in the following example:

   ```
   cicsadd -r cicswint -P -c cd CESA          \
           ConnectionType=ppc_gateway         \
           RemoteLUName="CICSESA"             \
           RemoteNetworkName="MYSNANET"       \
           GatewayName="GWY"                  \
           AllocateTimeout=60                 \
           RemoteCodePageTR="IBM-037"         \
           RemoteSysSecurity=trusted          \
           LinkUserId="LINKCESA"
   ```

   In this example, a Communications Definitions (CD) entry called CESA is added to the cicswint region's permanent database. (If you add a CD entry to a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the entry is being added only to the permanent database, the region would have to be cold-started.)

   The attributes that are listed in this example configure the region as follows:

   - The **ConnectionType** attribute value ppc_gateway specifies that PPC Gateway server SNA is to be used.
   - The **RemoteLUName** attribute value CICSESA specifies the LU name of the remote system.

- The **RemoteNetworkName** attribute value MYSNANET defines the SNA network to which the remote system is connected.
- The **GatewayName** attribute value of GWY specifies the PPC Gateway server that is to be used. (It is set to the name of the Gateway Definitions (GD) entry defined for the PPC Gateway server.)
- The **AllocateTimeout** attribute value of 60 defines, in seconds, how long CICS waits for the PPC Gateway server to accept an intersystem request.
- The **RemoteCodePageTR** attribute value (in this case, IBM-037) determines which character set flows across the network during transaction routing. The correct code page depends on the national language of your local region and the remote system type. See "Data conversion for transaction routing" on page 164 for information about how to choose this value.
- The **RemoteSysSecurity** attribute value of trusted specifies that CICS use the security information (such as user ID) that is sent with an intersystem request.
- The **LinkUserId** attribute value LINKCESA identifies a locally defined user ID that can be associated with inbound requests. Its value is related to the value of the **RemoteSysSecurity** attribute, which, in this case, is set to trusted.

The following attributes can also be specified:

- The **SNAConnectName** attribute specifies the name of the partner LU alias that is defined in the SNA product. It is required only if the partner LU alias is different from the partner LU name. (The *partner LU* is the SNA communications product's term for the remote region's LU name. The *partner LU alias* is an associated name for the partner LU; it is configured in the SNA communications product.)
- The **DefaultSNAModeName** attribute specifies the SNA modegroup that is to be used for intersystem requests when an SNA modename is not specified in either the PROFILE option of the EXEC CICS ALLOCATE command or in the **SNAModeName** attribute of the Transaction Definitions (TD) entry. (A *modegroup*, or *modename*, is defined in the SNA communications product. It defines the number of sessions that are associated with a connection and the characteristics of those sessions.) See one of the following documents for more information, depending on your SNA communications product:
  - *TXSeries for Multiplatforms Using IBM Communications Server for AIX with CICS*
  - *TXSeries for Multiplatforms Using IBM Communications Server for Windows Systems with CICS*
  - *TXSeries for Multiplatforms Using Microsoft SNA Server with CICS*
  - *TXSeries for Multiplatforms Using HP-UX SNAplus2 with CICS*
  - *TXSeries for Multiplatforms Using SNAP-IX for Solaris with CICS*

  If the **DefaultSNAModeName** attribute is set to "" (its default value), and the transaction has not provided a modename, the modename that is configured for the SNA product is used. Refer to "Default modenames" on page 110 for more information about the use of this attribute.
- The **OutboundUserIds** attribute determines the security information that is sent with outbound requests. "Setting up a CICS region to flow user IDs" on page 130 and "Setting up a CICS region to flow passwords" on page 131 describe how this option works.
- The **RemoteSysSecurity** attribute specifies the type of security that is required:
  - Using the default value of **local** causes CICS to run all incoming intersystem requests from the remote system under the user ID value that you specify in the **LinkUserId** field. The User Definitions (UD) entry for

this user ID determines which resources these intersystem requests can access. This type of security is called *link security* and is described in "CICS link security" on page 126.

– Choosing the **verify** or **trusted** values causes CICS to use the security information (such as the user ID and password) that is sent with the intersystem request. CICS also uses the user ID that you specify in the **LinkUserId** field to restrict the resources that inbound intersystem requests can access. This type of security is called *user security* and is described in "CICS user security" on page 127.

- See Chapter 6, "Configuring intersystem security," on page 123 for more information about how CICS security is configured, including descriptions of the **TSLKeyMask** and **RSLKeyMask** attributes.

Configuring CICS for PPC Gateway server SNA support by using the IBM TXSeries Administration Tool (CICS on Windows platforms only)

To configure CICS for PPC Gateway server SNA by using the IBM TXSeries Administration Tool, you must configure a Gateway Definitions (GD) entry and a Communications Definitions (CD) entry for each remote system with which your system is to communicate. It is also recommended that you update the **Local SYSID** Region Definitions (RD) attribute. Perform the following procedures to set these definitions (assume that default values are accepted for any attributes that are not discussed):

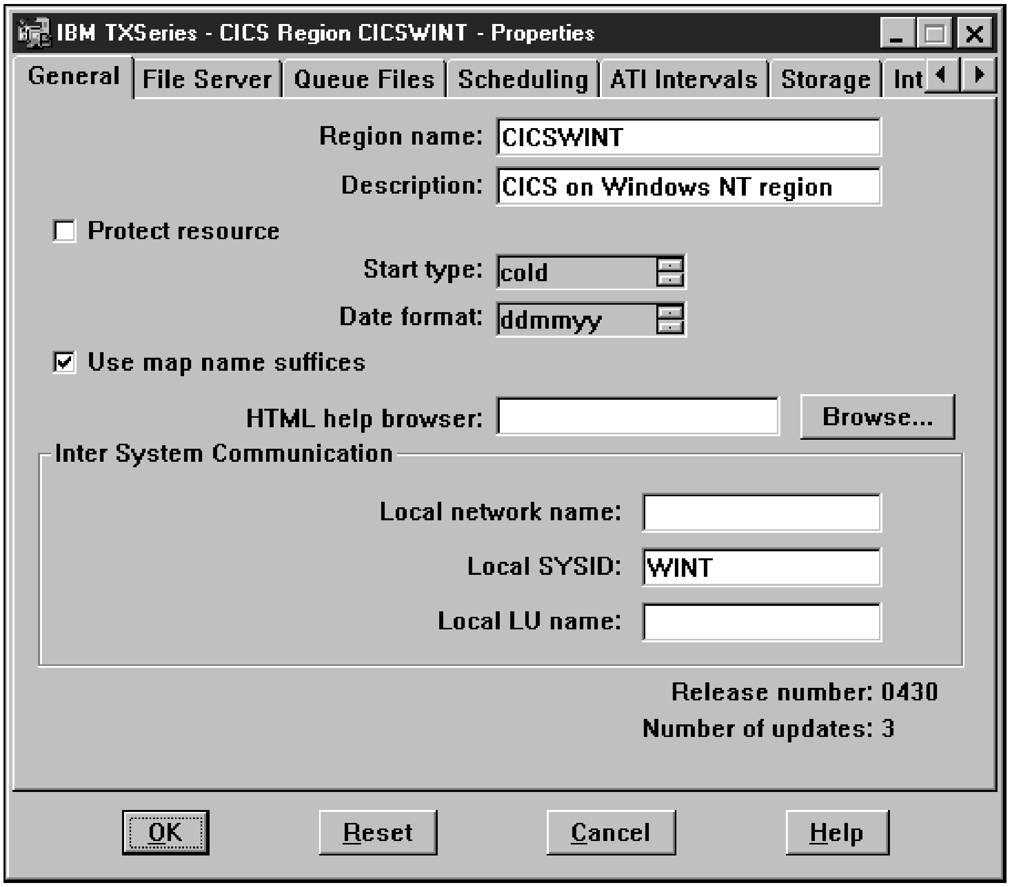
First, configure a Gateway Definitions (GD) entry by doing the following:

1. Click the region name in the IBM TXSeries Administration Tool Administration window.
2. Click **Subsystem>Resources>Gateway**. The Gateways window opens.
3. Click **Gateways>New**. The Gateway Definition window opens.
4. In the **Gateway name** field, type a name for the Gateway Definitions (GD) entry. This name can be up to four characters in length and must be different from the names of all other GD entries that are in your region. It does not have to be unique within the network and it does not relate to any other names in the network.
5. Type a description if desired.
6. Type the Gateway LU name in the **Gateway LU name** field.
7. Type the CDS name in the **CDS name** field. Specify either the short name for the PPC Gateway server (for example, `cicsgwy`) or the full path name of the PPC Gateway server (for example, **/.:/cics/ppc/gateway/cicsgwy**). If a name is provided without an initial slash (/), CICS inserts the string **/.:/cics/ppc/gateway** in front of the supplied name.
8. Click the **Permanent** button to add the Gateway Definitions (GD) entry to the permanent database, or click the **Both** button to add the entry to the permanent and runtime databases.
9. Repeat this process for any other Gateway Definitions (GD) entries.

Next, update the **Local SYSID** Region Definitions (RD) attribute by doing the following:

1. Click the region name in the IBM TXSeries Administration Tool Administration window.
2. Click **Subsystem>Properties**. The Properties window opens.

3. Type a description if desired.
4. Type the region's short name in the **Local SYSID** field. An example configuration window is shown in Figure 43.



Region name: CICSWINT

Description: CICS on Windows NT region

☐ Protect resource

Start type: cold

Date format: ddmmyy

☑ Use map name suffices

HTML help browser:

Inter System Communication

Local network name:

Local SYSID: WINT

Local LU name:

Release number: 0430

Number of updates: 3

Figure 43. Region Definition Properties window

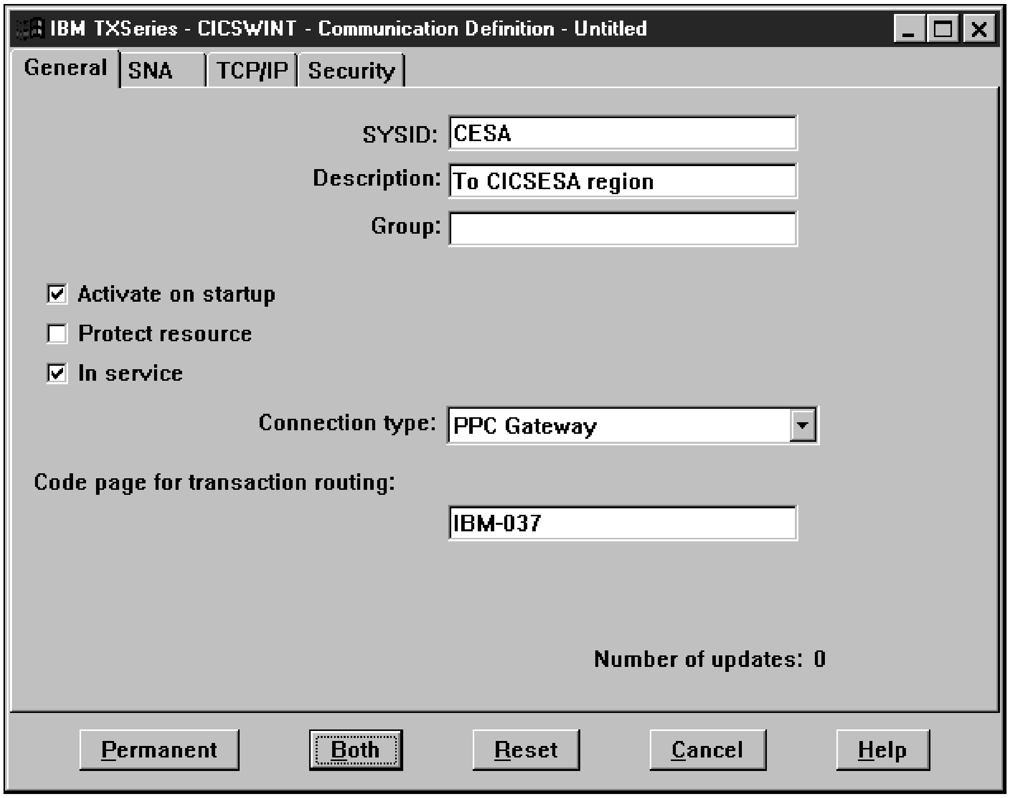
5. Click **OK** to add the updated Region Definitions (RD) attributes to the permanent database. (If you update RD attributes in a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the attributes are updated only in the permanent database, the region must be cold-started.)

Finally, configure a Communications Definitions (CD) entry for each remote system with which your system is to communicate by doing the following:

1. In the IBM TXSeries Administration Tool Administration window, click the region name.
2. Click **Subsystem>Resources>Communication**. The Communications window opens.
3. Click **Communications>New**. The Communication Definition window opens. The attributes for a Communications Definitions (CD) entry are grouped on four tabs:
   - **General**: Contains attributes that are required by all CD entries.
   - **SNA**: Contains attributes that describe how the remote system communicates over an SNA network.

- **TCP/IP**: Contains attributes that describe how the remote system is connected to a TCP/IP network. Attributes on this page are not applicable to a local SNA CD entry.
- **Security**: Contains attributes that define how security is managed.

4. On the **General** tab:
   a. In the **SYSID** field, type a four-character name for the Communications Definitions (CD) entry. This name must be different from the names of all other CD entries that are in your region. However, it does not have to be unique within the network and does not relate to any other names in the network.
   b. Type a description if desired.
   c. Select the **PPC Gateway** option from the **Connection type** menu.
   d. Accept the default value, or type the appropriate value in the **Code page for transaction routing** field. This value determines which character set flows across the network during transaction routing. The correct code page depends on the national language of your local region and the remote system type. See Chapter 7, "Data conversion," on page 147 for information about how to choose this value. An example configuration window is shown in Figure 44.



IBM TXSeries - CICSWINT - Communication Definition - Untitled

SYSID: CESA

Description: To CICSESA region

Group:

☑ Activate on startup

□ Protect resource

☑ In service

Connection type: PPC Gateway

Code page for transaction routing: IBM-037

Number of updates: 0

Figure 44. Communication Definition window: General tab

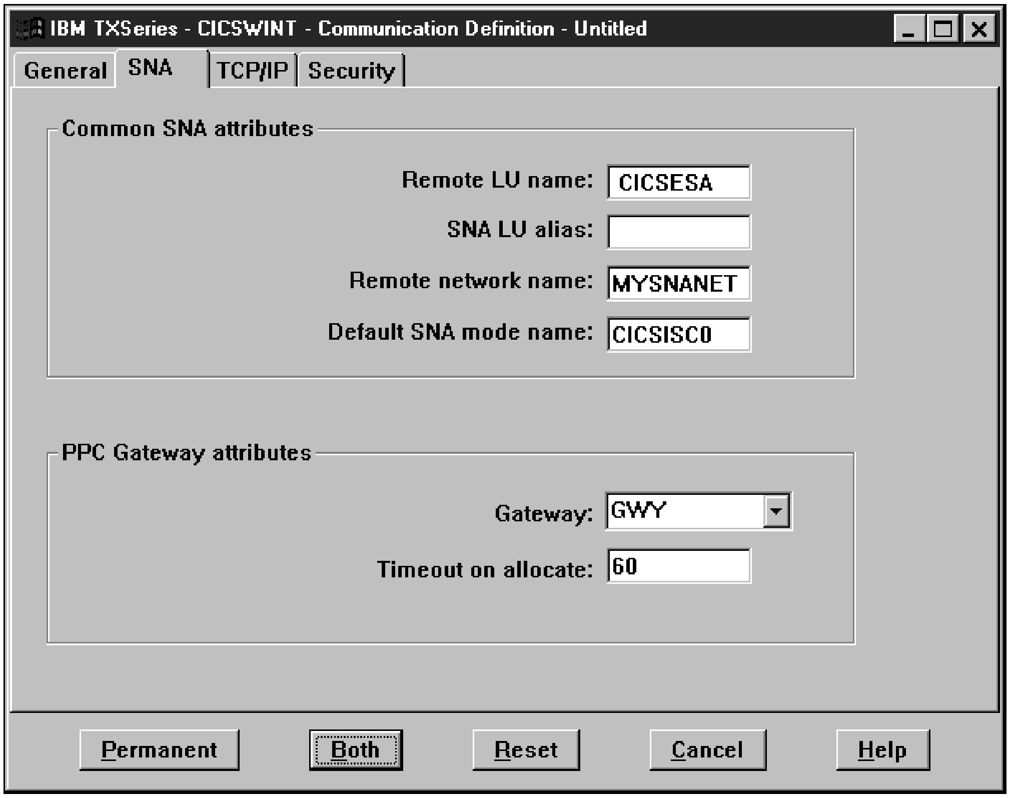
5. Click the **SNA** tab. On this tab:
   a. In the **Remote LU name** field, type the LU name of the remote system.
   b. Optionally, in the **SNA LU alias** field, type the LU alias that is configured for the remote system in the SNA product that is used by the PPC Gateway server. (In this example, this field is left empty.)

c. In the **Remote network name** field, type the SNA network to which the remote system is connected.

d. In the **Default SNA mode name** field, type the name of the SNA modegroup that is to be used for intersystem requests when an SNA modename is not specified either in the PROFILE option of the EXEC CICS ALLOCATE command, or in the **SNAModeName** attribute of the Transaction Definitions (TD) entry. Refer to “Default modenames” on page 110 for more information about the use of this attribute.

e. Select the name of the GD entry that was created in step 4 on page 98 for the **Gateway** field value.

f. Accept the default value of `60` seconds or type a value in the **Timeout on allocate** field to indicate the length of time CICS waits for the PPC Gateway server to accept an intersystem request. This specification prevents the CICS transaction that is issuing the intersystem request from becoming suspended if the PPC Gateway server becomes overloaded or fails while accepting an intersystem request. An example configuration window is shown in Figure 45.



IBM TXSeries - CICSWINT - Communication Definition - Untitled

Common SNA attributes

Remote LU name: CICSESA

SNA LU alias:

Remote network name: MYSNANET

Default SNA mode name: CICSISC0

PPC Gateway attributes

Gateway: GWY

Timeout on allocate: 60

Figure 45. Communication Definition window: SNA tab

6. Click the **Security** tab. These attributes describe the security checks that are applied to all intersystem requests that use this CD entry. On this tab:

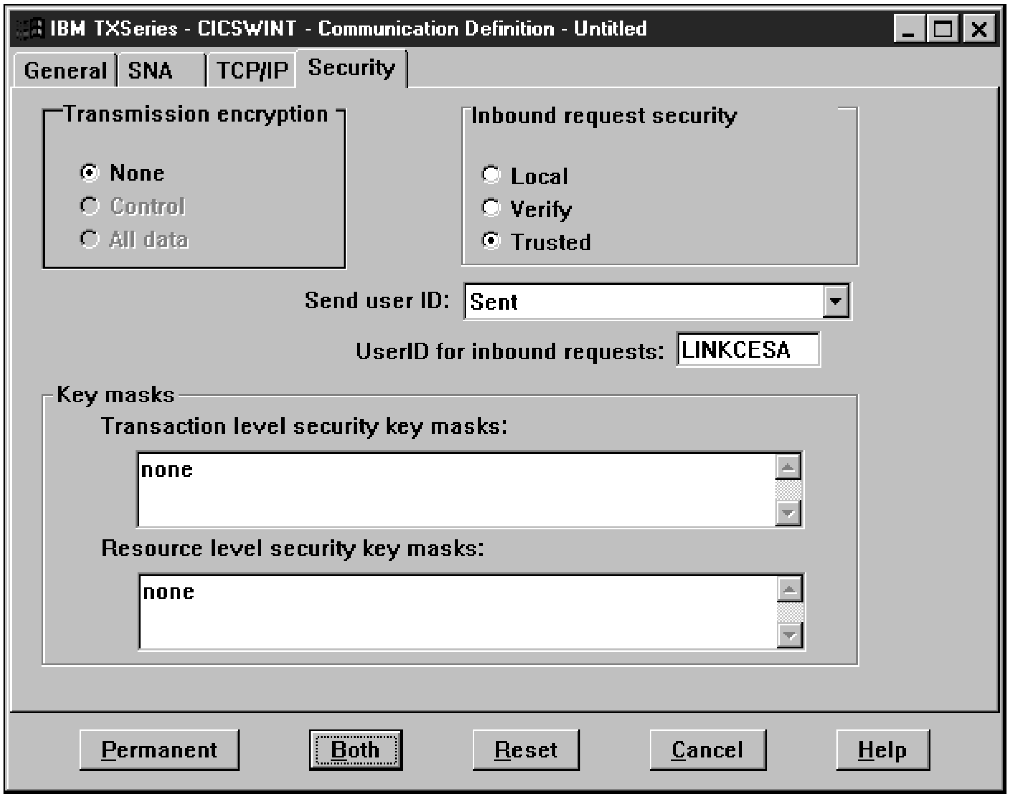
   a. In the **Inbound request security** area, click the type of security that you require:

      - Accepting the default **Local** option causes CICS to run all incoming intersystem requests from the remote system under the user ID value that you specify in the **UserID for inbound requests** field. The User Definitions (UD) entry for this user ID determines which resources these intersystem requests can access. This type of security is called *link security* and is described in “CICS link security” on page 126.

- Choosing the **Verify** or **Trusted** options causes CICS to use the security information (such as the user ID and password) that is sent with the intersystem request. CICS also uses the user ID that you specify in the **UserID for inbound requests** field to restrict the resources that inbound intersystem requests can access. This type of security is called *user security* and is described in “CICS user security” on page 127.

b. Security information that is sent with outbound requests is determined by the value that is selected in the **Send user ID** field. “Setting up a CICS region to flow user IDs” on page 130 and “Setting up a CICS region to flow passwords” on page 131 describe how this option works.

c. See Chapter 6, “Configuring intersystem security,” on page 123 for more information about how CICS security is configured, including descriptions of the **Transaction level security key masks** and **Resource level security key masks** fields. An example configuration window is shown in Figure 46.



IBM TXSeries - CICSWINT - Communication Definition - Untitled

Transmission encryption

● None
○ Control
○ All data

Inbound request security

○ Local
○ Verify
● Trusted

Send user ID: Sent

UserID for inbound requests: LINKCESA

Key masks

Transaction level security key masks: none

Resource level security key masks: none

Figure 46. Communication Definition window: Security tab

7. Click the **Permanent** button to add the Communications Definitions (CD) entry to the permanent database, or click the **Both** button to add the entry to the permanent and runtime databases.
8. Repeat this process for any other Communications Definitions (CD) entries.

Configuring CICS for PPC Gateway server SNA support by using SMIT (CICS for AIX only)

To configure CICS for PPC Gateway server by using SMIT, perform the following procedure. (Assume that default values are accepted for any attributes that are not discussed):

1. Ensure that you are logged into AIX with enough privileges to change the region database. (For example, log into AIX as the **root** user.)
2. Optionally, set the environment variable CICSREGION to the name of your CICS region. For example:

   ```
   export CICSREGION=cicsaix
   ```

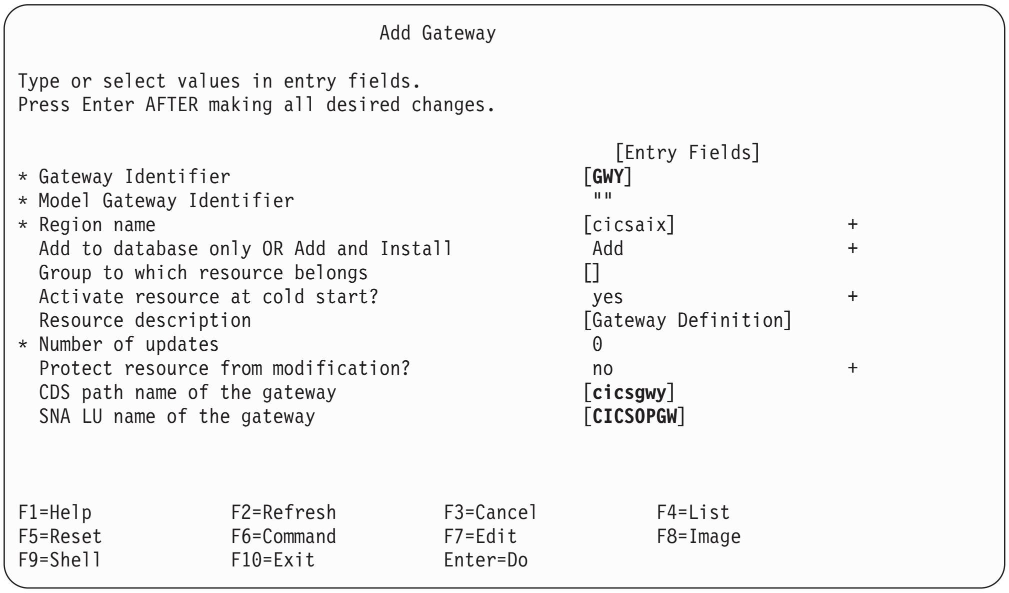
   Note: In this procedure, we call the local region `cicsaix` to reflect a CICS for AIX system. Also, this example assumes that you are using the Korn shell; if you are using a different shell, change the **export** command accordingly.
3. Enter `smitty cicsregion` to start SMIT.
4. If you have not set the CICSREGION environment variable as described in step 2, select the option **Change Working CICS Region**.
5. Select your CICS region from the list that is displayed and press the Enter key. The COMMAND STATUS screen verifies your selection.
6. Press the F3 key.
7. Configure a Gateway Definitions (GD) entry by performing the following procedure:
   a. Select options:

      ```
      ► Define Resources for a CICS Region
          ► Manage Resource(s)
              ► Gateways
                  ► Add New
      ```

      The Add Gateway panel is displayed.

   b. Enter a value for the **Model Gateway Identifier** attribute. Use the name of a GD entry that you have defined previously, or press the Enter key to use the default value ("").
   c. Enter the name of the GD entry in the **Gateway Identifier** field.
   d. Enter a description in the **Resource description** field if desired.
   e. In the **CDS path name of the gateway** field, enter the name of the PPC Gateway server that the region is to use.

      Note: This attribute accepts either the full path name of the PPC Gateway server or a partial name. If a partial name is supplied, CICS appends it to the path **/.:/cics/ppc/gateway/**.
   f. In the **SNA LU name of the gateway** field, enter the SNA LU name that the gateway uses to communicate with the local region. In the example in Figure 47 on page 104, the gateway uses the name `CICSOPGW`.



```
                                Add Gateway

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Gateway Identifier                                 [GWY]
* Model Gateway Identifier                            ""
* Region name                                        [cicsaix]              +
  Add to database only OR Add and Install             Add                   +
  Group to which resource belongs                    []
  Activate resource at cold start?                    yes                   +
  Resource description                               [Gateway Definition]
* Number of updates                                   0
  Protect resource from modification?                 no                    +
  CDS path name of the gateway                       [cicsgwy]
  SNA LU name of the gateway                         [CICSOPGW]



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

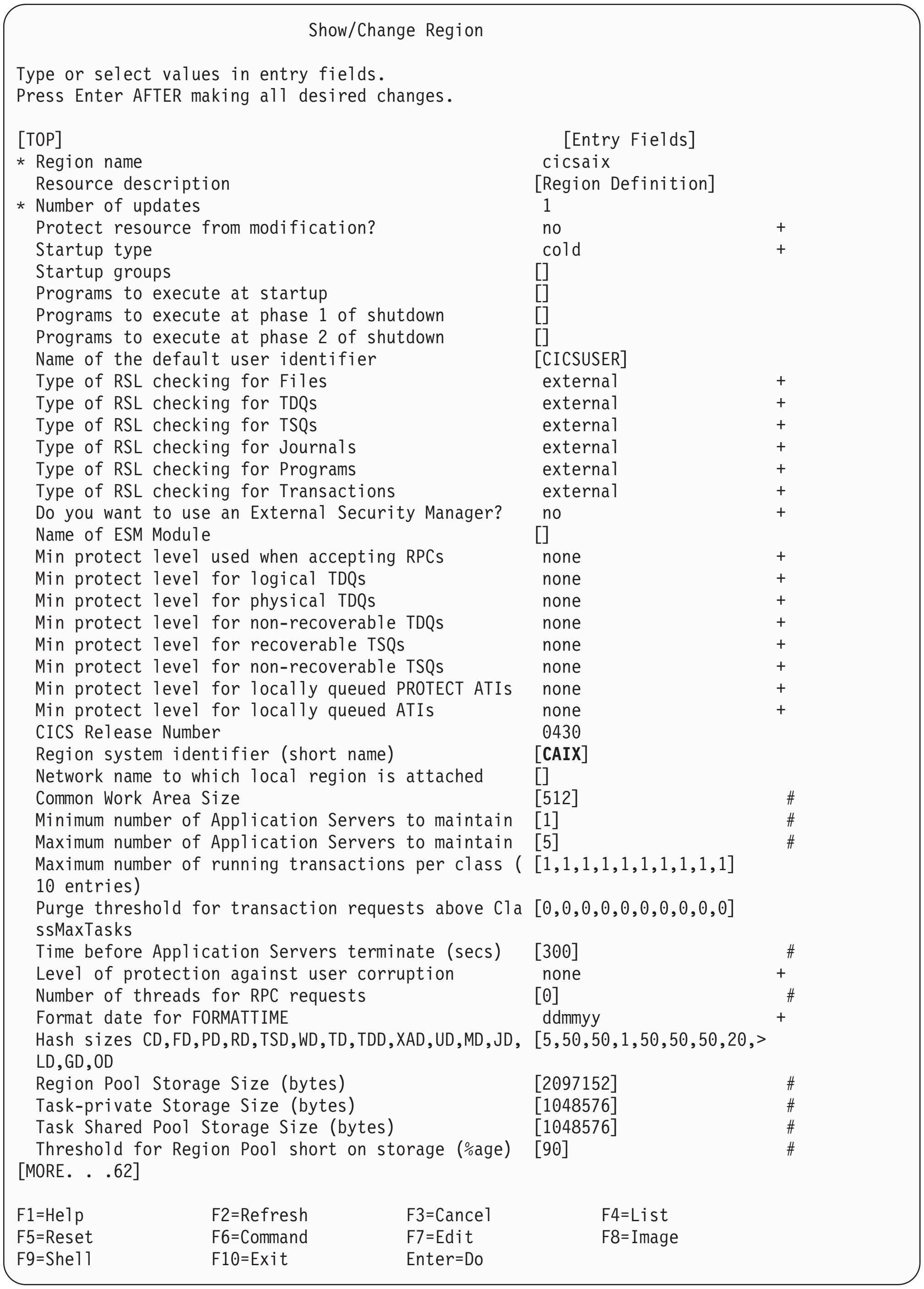
Figure 47. Add Gateway SMIT panel

8. Press the Enter key to create the GD entry. (If you add a GD entry to a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the entry is added only to the permanent database, the region must be cold-started.) The COMMAND STATUS screen verifies successful creation of the definition.
9. Press the F3 key three times to return to the `Manage Resource(s)` menu.
10. Update the **Region system identifier (short name)** Region Definitions (RD) attribute by performing the following procedure:
    a. From the `Manage Resource(s)` menu, select the options:

       ► `Region`
         ► `Show/Change`

       The Show/Change Region panel is displayed.
    b. Enter a description in the **Resource description** field if desired.
    c. Enter a four-character name for the CICS region, known as the SYSID, in the **Region system identifier (short name)** field.

       Figure 48 on page 105 shows an example Show/Change Region SMIT panel. (Only the panel that shows the updated attribute is shown.)



```
                              Show/Change Region

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
* Region name                                         cicsaix
  Resource description                               [Region Definition]
* Number of updates                                   1
  Protect resource from modification?                 no                     +
  Startup type                                        cold                   +
  Startup groups                                     []
  Programs to execute at startup                     []
  Programs to execute at phase 1 of shutdown         []
  Programs to execute at phase 2 of shutdown         []
  Name of the default user identifier                [CICSUSER]
  Type of RSL checking for Files                      external               +
  Type of RSL checking for TDQs                       external               +
  Type of RSL checking for TSQs                       external               +
  Type of RSL checking for Journals                   external               +
  Type of RSL checking for Programs                   external               +
  Type of RSL checking for Transactions               external               +
  Do you want to use an External Security Manager?    no                     +
  Name of ESM Module                                 []
  Min protect level used when accepting RPCs          none                   +
  Min protect level for logical TDQs                  none                   +
  Min protect level for physical TDQs                 none                   +
  Min protect level for non-recoverable TDQs          none                   +
  Min protect level for recoverable TSQs              none                   +
  Min protect level for non-recoverable TSQs          none                   +
  Min protect level for locally queued PROTECT ATIs   none                   +
  Min protect level for locally queued ATIs           none                   +
  CICS Release Number                                 0430
  Region system identifier (short name)              [CAIX]
  Network name to which local region is attached     []
  Common Work Area Size                              [512]                    #
  Minimum number of Application Servers to maintain  [1]                      #
  Maximum number of Application Servers to maintain  [5]                      #
  Maximum number of running transactions per class ( [1,1,1,1,1,1,1,1,1,1]
  10 entries)
  Purge threshold for transaction requests above Cla [0,0,0,0,0,0,0,0,0,0]
  ssMaxTasks
  Time before Application Servers terminate (secs)   [300]                    #
  Level of protection against user corruption         none                   +
  Number of threads for RPC requests                 [0]                      #
  Format date for FORMATTIME                          ddmmyy                 +
  Hash sizes CD,FD,PD,RD,TSD,WD,TD,TDD,XAD,UD,MD,JD, [5,50,50,1,50,50,50,20,>
  LD,GD,OD
  Region Pool Storage Size (bytes)                   [2097152]                #
  Task-private Storage Size (bytes)                  [1048576]                #
  Task Shared Pool Storage Size (bytes)              [1048576]                #
  Threshold for Region Pool short on storage (%age)  [90]                     #
[MORE. . .62]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 48. Show/Change Region SMIT panel

d. Press Enter to update the Region Definitions (RD) attribute. (If you update an RD attribute in a region while it is running, the change is not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the attribute is updated only in the permanent database, the region must be cold-started.) The COMMAND STATUS screen confirms successful completion of the process.

e. Press the F3 key three times to return to the `Manage Resource(s)` menu.

11. Configure a Communications Definitions (CD) entry for each remote system with which your system is to communicate, by performing the following procedure:

    a. Select options:

        ```
        ► Communications
            ► Add New
        ```

        The Add Communication panel is displayed.

    b. Enter a value for the **Model Communication Identifier** attribute. Use the name of a CD entry that you have defined previously, or press the Enter key to use the default value ("").

    c. Enter the name of the CD entry in the **Communication Identifier** field.

    d. Add a description in the **Resource description** field if desired.

    e. Select the **ppc_gateway** option in the **Connection type** field.

    f. Enter the LU name of the remote system in the **Name of remote system** field.

    g. Enter the name of the network to which the remote system is attached in the **SNA network name for the remote system** field.

    h. In the **Default modename for a SNA connection** field, enter the name of the SNA modegroup that is to be used for intersystem requests when an SNA modename is not specified in either the PROFILE option of the EXEC CICS ALLOCATE command or in the **SNAModeName** attribute of the Transaction Definitions (TD) entry. If the **Default modename for a SNA connection** attribute is set to "" (its default value), and the transaction has not provided a modename, the modename that is configured in the AIX SNA communications product's side information profile for the local LU name is used. Refer to "Default modenames" on page 110 for more information about the use of this attribute.

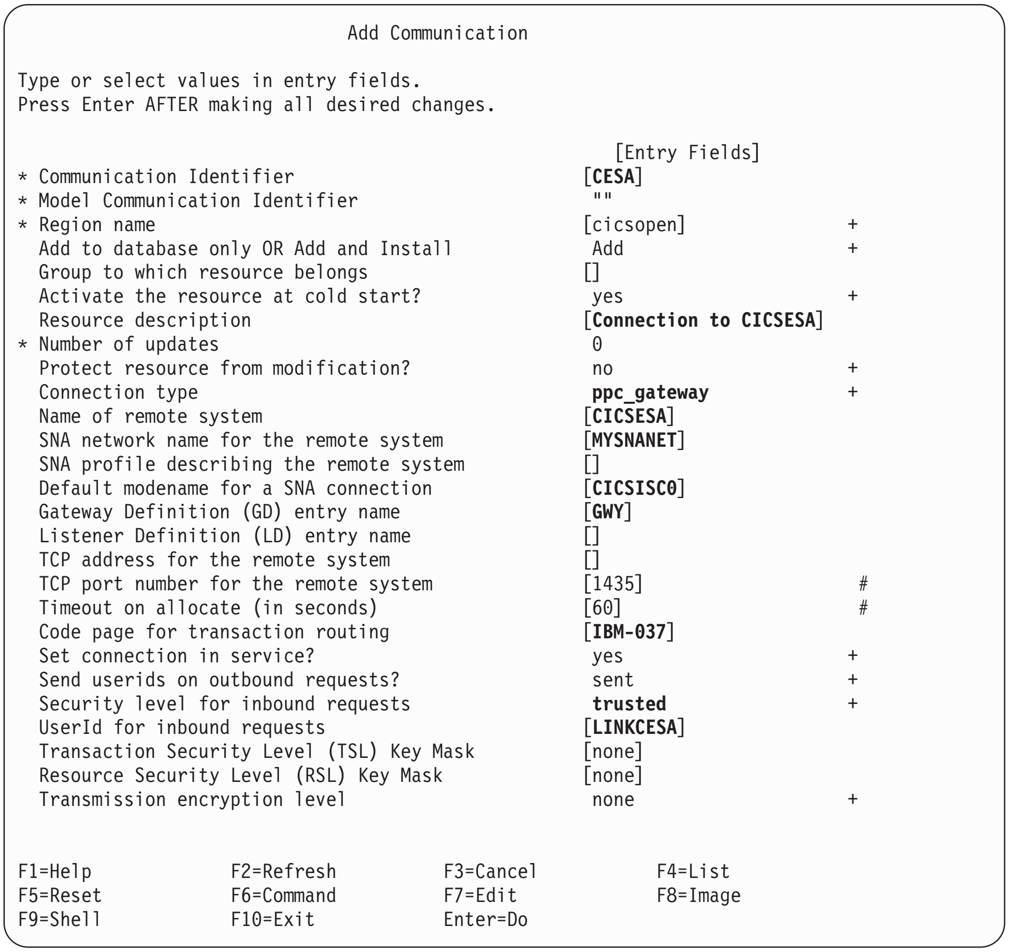
    i. Accept the default value or enter the appropriate value in the **Code page for transaction routing** field. This value determines which character set flows across the network during transaction routing. The correct code page depends on the national language of your local region and the remote system type. See "Data conversion for transaction routing" on page 164 for information on how to choose this value.

    j. Accept the default value or select the appropriate value in the **Send userids on outbound requests?** attribute to determine the type of security information (if any) that is sent with outbound requests. "Setting up a CICS region to flow user IDs" on page 130 and "Setting up a CICS region to flow passwords" on page 131 describe how this attribute works.

    k. In the **Security level for inbound requests** field, select the type of security that you require:

        - Accepting the default **local** option causes CICS to run all incoming intersystem requests from the remote system under the user ID value that you specify in the **UserID for inbound requests** field. The User Definitions (UD) entry for this user ID determines which resources these intersystem requests can access. This type of security is called *link security* and is described in "CICS link security" on page 126.
        - Choosing the **verify** or **trusted** options causes CICS to use the security information (such as the user ID and password) that is sent with the intersystem request. CICS also uses the user ID that you specify in the **UserID for inbound requests** field to restrict the resources that inbound intersystem requests can access. This type of security is called *user security* and is described in "CICS user security" on page 127.

l. See Chapter 6, “Configuring intersystem security,” on page 123 for more information about how CICS security is configured, including descriptions of the **Transaction Security Level (TSL) Key Mask** and **Resource Security Level (RSL) Key Mask** attributes. Figure 49 shows an example Add Communication SMIT panel.



```
                                Add Communication

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                     [Entry Fields]
* Communication Identifier                        [CESA]
* Model Communication Identifier                   ""
* Region name                                     [cicsopen]              +
  Add to database only OR Add and Install          Add                    +
  Group to which resource belongs                 []
  Activate the resource at cold start?             yes                    +
  Resource description                            [Connection to CICSESA]
* Number of updates                                0
  Protect resource from modification?              no                     +
  Connection type                                  ppc_gateway            +
  Name of remote system                           [CICSESA]
  SNA network name for the remote system          [MYSNANET]
  SNA profile describing the remote system        []
  Default modename for a SNA connection           [CICSISC0]
  Gateway Definition (GD) entry name              [GWY]
  Listener Definition (LD) entry name             []
  TCP address for the remote system               []
  TCP port number for the remote system           [1435]                   #
  Timeout on allocate (in seconds)                [60]                     #
  Code page for transaction routing               [IBM-037]
  Set connection in service?                       yes                    +
  Send userids on outbound requests?               sent                   +
  Security level for inbound requests              trusted                +
  UserId for inbound requests                     [LINKCESA]
  Transaction Security Level (TSL) Key Mask       [none]
  Resource Security Level (RSL) Key Mask          [none]
  Transmission encryption level                    none                   +


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 49. Add Communication SMIT panel

m. Press Enter to create the CD entry. (If you add a CD entry to a region while it is running, the changes are not enabled until the region is reinitialized by either an autostart or a cold start. In this example, because the entry is added only to the permanent database, the region must be cold-started.) The COMMAND STATUS screen verifies the definition creation.

n. Press the F3 key three times to return to the `Manage Resource(s)` menu.

o. Repeat this process for any other Communications Definitions (CD) entries.

Chapter 5. Configuring resources for intercommunication

This chapter describes how to configure resources on your CICS system so that they can be shared with other systems. The following are described:

- "Configuring transactions for intersystem communication"
- "Configuring intrapartition TDQs for intercommunication" on page 111
- "Configuring Program Definitions (PD) for DPL" on page 112
- "Defining remote resources for function shipping" on page 113
- "Defining resources for transaction routing" on page 116
- "Defining remote transactions for asynchronous processing" on page 120

Configuring transactions for intersystem communication

This section describes:

- "Transactions over an SNA connection (CICS on Open Systems only)"
- "Back-end DTP transactions over TCP/IP and an SNA connection"
- "Configuring for the indoubt condition"

Other sections that are relevant to configuring transactions for intercommunication are:

- "Defining remote transactions for transaction routing" on page 119
- "Defining remote transactions for asynchronous processing" on page 120

Transactions over an SNA connection (CICS on Open Systems only)

If you are using Communications Server for AIX, to make transactions available for function shipping, transaction routing, and distributed transaction processing (DTP) requests over an SNA connection, you need to set the Transaction Definitions (TD) **TPNSNAProfile** attribute to the name of an AIX SNA TPN Profile.

On HP-UX SNAplus2 or SNAP-IX for Solaris, you do not need to specify a **TPNSNAProfile** in the TD entry. Leave the attribute blank.

Back-end DTP transactions over TCP/IP and an SNA connection

CICS cannot distinguish between an inbound intersystem request for a DTP back-end transaction and an inbound intersystem request for a transaction-routed transaction. Therefore, it uses the **IsBackEndDTP** attribute of the Transaction Definitions (TD) to decide how to process the request. If you are using a back-end DTP transaction, set the transaction's TD attribute **IsBackEndDTP=yes**; otherwise, your back-end (remotely linked-to) program fails to start.

Note: In some conditions, setting **IsBackEndDTP=yes** for transactions that are not run as back-end DTP transactions can result in a failure to start the transaction. Therefore, do not set this attribute to **yes** unless it is required.

Configuring for the indoubt condition

Intercommunication with synchronization level 2 uses a *two-phase commit*. A two-phase commit is a protocol for the coordination of changes to recoverable resources when more than one resource manager is used by a single transaction. In the first phase, all resource managers are asked to prepare their work; in the second phase, they are all requested to either commit or back out (roll back).

While CICS is processing a two-phase commit for a transaction, a connection or system error might occur. Some errors cause the transaction to wait until either you correct the error, or it is corrected by CICS itself. These waiting transactions are said to be in an *Indoubt condition*.

Because you might not want the transaction to wait for the error to be corrected, do the following:

1. Use the Transaction Definitions (TD) **InDoubt** attribute to specify how the condition is to be handled. When a transaction is waiting in the first phase of the two-phase commit process, and a CEMT SET TASK FORCEPURGE is issued against the transaction, CICS uses this attribute to determine whether to commit or back out any changes that were made by the transaction before the wait occurred. The **InDoubt** attribute can be set to the following values:
   - **wait_commit** (commit changes)
   - **wait_backout** (back out changes)
2. Use CEMT INQUIRE TASK INDOUBT to determine whether the transaction is in the InDoubt condition.
3. Use CEMT SET TASK FORCEPURGE to resolve the indoubt condition in accordance with the TD **InDoubt** attribute setting.

Default modenames

All intersystem requests over SNA require a modename. You can specify a modename in several ways.

The modename can be specified explicitly for each intersystem request by using either:

- The Profile option of the EXEC CICS ALLOCATE command (DTP only)
- The **SNAModeName** attribute in the Transaction Definitions (TD) entry

Alternatively, if you do not use the PROFILE or the **SNAModeName** attribute, you can specify a default modename in the **DefaultSNAModeName** attribute on the Communications Definitions (CD) entry.

If you do not specify a modename in the PROFILE, or the **SNAModeName** attribute in the TD entry, or the **DefaultSNAModeName** attribute in the CD entry, the CICS_SNA_DEFAULT_MODE environment variable can be used to set a regionwide default mode name. It is read during region startup, and a message is logged to the console.*nnnnnn* file so that you can check whether CICS has picked up the value correctly. For further information about all the environment variables that CICS uses, refer to the *TXSeries for Multiplatforms Administration Reference*.

If none of the above is selected, the selection of modename depends on the configuration of the particular SNA product that you are using.

Whichever modename is used must be correctly configured in the SNA product.

When using a PPC Gateway

The modename that is used is the value that is set with the Gateway Server Definitions (GSD) **SNADefaultModeName** attribute. See “Creating a PPC Gateway server” on page 184 for more information.

SNA tuning

For assistance in tuning your SNA environment, refer to:

- "Defining remote transactions for transaction routing" on page 119
- "Defining remote transactions for asynchronous processing" on page 120
- "Designing distributed processes" on page 283
- *TXSeries for Multiplatforms Administration Reference*
- Performance documentation for your SNA product.

When using HP-UX SNAplus2

Use the environment variable CICS_SNA_THREAD_POOL_SIZE to change the number of threads that are available to handle incoming intersystem requests. This environment variable has a range of 1 through 100. It defaults to 6 if it is not specified. When high numbers of SNA back-end transactions are expected on CICS for HP-UX, CICS_SNA_THREAD_POOL_SIZE should be increased. This is configured in your region's environment file.

Note: This environment variable is relevant only when you are using local SNA support.

Configuring intrapartition TDQs for intercommunication

Intrapartition transient data queues (TDQs) can be configured with a trigger level value, such that when the number of records that are written to the queue reach that value, a transaction is automatically initiated. This is an example of automatic transaction initiation (ATI).

The purpose of that transaction is typically to process the records on the queue. The transaction can run locally on the region that owns the transient data queue, or it can use intercommunication functions.

The following Transient Data Definitions (TDD) attributes are used:

TriggeredTransId

This attribute identifies the transaction that is to be automatically initiated when the trigger level is reached. The purpose of transactions that are initiated in this way is to read records from the destination. This transaction must reside in the same region as is the TDQ that triggers it. However, it might do one of the following:

- Run in the background
- Write to a local or remote terminal
- Acquire a conversation to another system and take part in a DTP conversation with a back-end program

This is determined by the TDD **FacilityType** and **FacilityId** attributes.

TriggerLevel

This attribute defines the number of records that are to be accumulated before a task is automatically initiated to process them.

FacilityType

This attribute defines the type of principle facility that is allocated for a triggered task. Specify **file** for the triggered transaction to run as a background task, **terminal** for the transaction to run against a terminal, and **system** for the task to use a DTP conversation.

FacilityId
This attribute defines the triggered transaction's principal facility identifier.

The principal facility that is associated with a transaction started by ATI can be:
- A local terminal
- A terminal that is owned by a remote region
- A DTP conversation to a remote region

Terminals as principal facilities

A local terminal is owned by the region that owns the transient data queue and the transaction.

For an ATI triggered transaction to use a local terminal, specify the **FacilityType** as **terminal**, and specify the terminal identifier in the **FacilityId** attribute. This terminal identifier (TERMID) must be the name of the Terminal Definitions (WD) entry.

Remote terminals as principal facilities

If **FacilityType=terminal**, you can define a terminal that is remote from the region that owns the transient data queue and the associated transaction.

Use the **FacilityId** attribute to specify the name of the Terminal Definitions (WD) entry for the remote terminal. You must define the terminal itself as a remote terminal, and you must connect the terminal-owning region to the local region through an intersystem connection. This intersystem connection is defined in a Communications Definitions (CD) entry.

ATI with a remote terminal is a form of CICS transaction routing, and the normal transaction routing rules apply. Refer to Chapter 12, "Transaction routing," on page 263.

Remote systems as principal facilities

If **FacilityType=system**, set the **FacillityId** attribute to the name of a Communications Definitions (CD) entry. In this case, the triggered transaction is started with a conversation that is allocated to the requested system as its principal facility. The conversation is in "allocated" state and the transaction should issue an EXEC CICS CONNECT PROCESS to connect to a back-end transaction.

For more information about transient data queues, see *CICS Administration Guide*. For more information about TDDs, see *TXSeries for Multiplatforms Administration Reference*.

Refer to "Conversation initiation and the front-end transaction" on page 286

Configuring Program Definitions (PD) for DPL

All CICS applications can link to a program that is running in a remote system, either by using the SYSID option in the EXEC CICS LINK command, or by creating a Program Definitions (PD) entry that defines the program as remote. The *TXSeries for Multiplatforms Administration Reference* describes all the attributes of a PD entry. This section describes those attributes that are relevant for remote programs.

The key, or name, of the PD entry
: Specifies the local name of the remote program. This is the name that is used in a local EXEC CICS LINK command that invokes the program. In the example, the name is LOCLPGM.

RemoteSysId attribute
: Specifies the name of the Communications Definitions (CD) entry for the connection to the remote system. In the example below, the name of the CD entry is CONR.

RemoteName attribute
: Specifies the name of the program in the remote CICS system in which it is located. In the example, the name of the remote program is REMTPGM.

  Note: A PD is not required for programs that are dynamically linked.

Example of a PD for DPL

This command adds a PD entry, called LOCLPGM, in a region named cicsopen:

```
% cicsadd -r cicsopen                         \
              -c pd LOCLPGM                    \
              RemoteSysId="CONR"         \
              RemoteName="REMTPGM"
```

When the EXEC CICS LINK PROGRAM(LOCLPGM) command is executed locally, a link request is shipped on connection CONR to a remote CICS system in which program REMTPGM is executed, by using the remote mirror transaction.

Defining remote resources for function shipping

The *TXSeries for Multiplatforms Administration Reference* gives a full description of all CICS resource definitions. The following information contains guidance about parameters that are important in the definition of remote data resources that are to be accessed by function shipping. Those data resources are:

- Remote files
- Remote transient data queues
- Remote temporary storage queues

Defining remote files

A remote file is a file that resides on another region. CICS uses function shipping for file control requests that are made against a remote file.

CICS application programs can name a remote region explicitly on file control requests by means of the SYSID option. If this is done, you need not define the remote file in the local region.

More generally, however, applications are designed to access files without being aware of their location, and in this case you must define the remote file in the local File Definitions (FD) entry.

Note: The definition of the file in the system that is named in the SYSID parameter can itself be a remote definition.

FD entries for remote files

Defining a file entry as remote provides CICS with enough information to ship file control requests to a specified remote region. You do this by specifying, in the **RemoteSysId** attribute of the FD, the SYSID of the Communications Definitions

(CD) entry for the connection to the remote region. See the **cicsadd** command example that is shown in "Example of a file definition for function shipping" for further information about this.

You must define a link to this region. The identifier that is specified for the remote region must not be the identifier of the local region.

File names

You specify the name by which the file is known on the local region in the FD. Application programs in the local region use this name in file control requests. In the example, the name is LOCLFILE.

You specify the name by which the file is known on the remote CICS region in the **RemoteName** attribute in the FD. CICS uses this name when shipping file control requests to the remote region. In the example, the name is REMTFILE.

If the name of the file is the same on both the local and the remote region, you do not need to specify the **RemoteName**. However, consider carefully the desirability of using the local file name to provide a local alias for the remote file name. This technique is, of course, essential if files of the same name reside on both regions and you do not want your local system to access the remote file of the same name.

Record lengths

You can specify the record length, in bytes, of a remote file by using the **KeyLength** attribute in the FD entry. This value must be the same as that which is specified in the description of the file in the remote system in which it is locally defined. In the example, the record length is 6.

If your installation uses COBOL, specify the record length for any file that has fixed-length records.

In all other cases, the record length is either a mandatory option on file control commands or can be deduced automatically by the command-language translator.

You can specify the maximum record size, in bytes, of a remote file by using the **RecordSize** attribute in the FD entry. This value must be the same as that which is specified in the description of the file in the remote system in which it is locally defined. In the following example, the record length is 86.

Example of a file definition for function shipping

This command adds an FD entry, called LOCLFILE, in the region named cicsopen:

```
% cicsadd -r cicsopen -c fd LOCLFILE RemoteSysId="CESA"          \
                                RemoteName="REMTFILE"        \
                                KeyLen=6 RecordSize=86
```

This entry enables the function shipping request for file LOCLFILE over connection CESA to a remote CICS system, where it accesses a file REMTFILE, which has a key length of 6 and a record length of 86.

Defining remote transient data queues

A remote transient data queue is one that resides on another region. CICS uses function shipping for transient data requests that are made against a remote queue.

CICS application programs can use the SYSID option to name a remote region explicitly on transient data requests. If this is done, you do not need to define the remote transient data queue on the local region.

More generally, however, applications are designed to access transient data queues without being aware of their location, and in this case you must define the remote queue in the local Transient Data Definitions (TDD).

TDD entries for remote transient data queues

A remote entry in the TDD provides CICS with enough information to ship transient data requests to a specified remote region. The name of the TDD entry is the name of the queue that is used by local transaction, (see the **cicsadd** command example shown in “Example of a TDD for function shipping”). In the example, the local name of the transient data queue is LTDQ.

Remote name
Use the TDD **RemoteName** attribute to specify the name of the transient data queue in the remote CICS system in which it is defined. In the example, this name is RTDQ.

Remote system
Use the TDD **RemoteSysId** attribute to specify the name of the Communications Definitions (CD) entry that defines the connection to the remote system in which the transient data queue resides. In the example, this name is CESA.

Example of a TDD for function shipping

This command adds a TDD entry, called LTDQ, to a CICS group, called GROUP, in the region named cicsopen:

```
% cicsadd -r cicsopen -c tdd LTDQ RemoteSysId="CESA"         \
                                 RemoteName="RTDQ"
```

This entry enables the function shipping of an access request for transient data queue LTDQ over connection CESA to a remote CICS system, where it accesses transient data queue RTDQ.

Defining remote temporary storage queues

CICS application programs use temporary storage queues to store data for later retrieval. A remote temporary storage queue is one that resides on another region. CICS uses function shipping for temporary storage requests that are made against a remote queue.

CICS application programs can use the SYSID option to name a remote region explicitly on temporary storage requests. If this is done, you need not define the remote temporary storage queue on the local region.

More generally, however, applications are designed to access temporary storage queues without being aware of their location, and in this case you must define the remote queue in the local Temporary Storage Definitions (TSD).

TSD entries for remote temporary storage queues

A remote entry in the TSD provides CICS with enough information to ship temporary storage requests to a specified remote region. The name of the TSD entry is the name of the queue that used by the local application (see the **cicsadd** command example shown in “Example of a TSD for function shipping” on page 116). In the example, the local name of the temporary storage queue is LOCLTSQ.

Remote system

Use the TSD **RemoteSysId** attribute to specify the name of the Communications Definitions (CD) entry for the connection to the remote system in which the real temporary storage queues reside. In the example, this connection name is CESA.

Remote queue name

Use the TSD **RemoteName** attribute to specify the name of the temporary storage prefix that is used by the remote system in which the queues are located. In the example, this prefix is REMTTSQ.

Example of a TSD for function shipping

This command adds a TSD entry, called LOCLTSQ, to a CICS group, called GROUP, in the region named cicsopen:

```
cicsadd -r cicsopen -c tsd LOCLTSQ RemoteSysId="CESA"          \
                                   RemoteName="REMTTSQ"
```

This entry enables the function shipping request for temporary storage queue LOCLTSQ over connection CESA to a remote CICS system, where it accesses temporary storage queue REMTTSQ.

Defining resources for transaction routing

Transaction routing enables a terminal that is owned by one CICS system (the terminal-owning region) to be connected to a transaction that is owned by another CICS system (the application-owning region). The following definitions are required:

- In the terminal-owning region:
  - If the terminal is not autoinstalled, it must be defined as a local resource on the terminal-owning region, by using a Terminal Definitions (WD) entry.
  - The transaction must be defined as a remote resource on the terminal-owning region if it is to be initiated from a local terminal or by automatic transaction initiation (ATI).
  - For dynamic transaction routing, the terminal-owning region requires the following information to be defined in the Transaction Definitions (TD):
    - The local name of the transaction
    - An indication that this is a dynamic transaction (**Dynamic=yes**)

    In dynamic transaction routing, values that are defined in the TD can be changed by the dynamic transaction routing user exit. Therefore, information such as the name of the connection to the region that owns the transaction and the name of the transaction in the region that owns it are not mandatory. For more information, see “Dynamic transaction routing” on page 267.
- In the application-owning region:
  - If the terminal definition is not shipped from the terminal-owning region, it must be defined on the application-owning region.
  - The transaction must be defined as a local resource on the application-owning region.

These rules also apply to intermediate systems when indirect routing is used. (See “Indirect links for transaction routing” on page 266.)

Note: The information that follows briefly describes only those resource definition attributes that are required for transaction routing. Refer to the *TXSeries for Multiplatforms Administration Reference* for complete details.

Shipping terminal definitions

To avoid the need for a remote definition in the application-owning region, a terminal can be defined as shippable in its local definition in the terminal-owning region. To do this, set the **IsShippable** attribute to **yes** for the Terminal Definitions (WD) for that terminal.

Note: This parameter defaults to **yes** and needs to be coded only when the intent is to prohibit terminal shipping.

For terminals that are autoinstalled, you *must* set the **IsShippable** attribute to **yes**. In effect, this gives automatic installation of remote terminals.

When a remote transaction is invoked from a shippable terminal (refer to the **IsShippable** attribute in the Terminal Definitions in the *TXSeries for Multiplatforms Administration Reference*), the request that is transmitted to the application-owning region is flagged to show that a shippable terminal definition is available. If the application-owning region already has a definition of the terminal (which might have been shipped previously), it ignores the flag. Otherwise, it asks for the definition to be shipped. A shipped terminal definition is retained until one of the following events occurs:

- The autoinstalled terminal definition on the terminal-owning region is deleted, or the user logs off a terminal that is not autoinstalled.
- The WD entry for the logged-on terminal on the terminal-owning region is changed or deleted.
- The system that shipped the terminal definition (the terminal-owning region) is restarted.
- The system that received the shipped terminal definition (the application-owning region) is restarted.

Note: The WD **IsShippable** attribute defaults to **yes**. Change this attribute to **no** when you do not want a terminal definition to be shipped. Do not change the attribute to **no** for terminals that are autoinstalled.

Fully qualified terminal identifiers

A unique identifier is used for every terminal that is involved in transaction routing. The identifier is formed from the APPLID of the terminal-owning region and the terminal identifier that is specified in the terminal definition on that region.

For example, if the APPLID of the terminal-owning region is PRODSYS, and the terminal identifier is L77A, the fully-qualified terminal identifier is PRODSYS.L77A.

Note: When referring to a remote region, the APPLID for that region is known in CICS as the Communications Definitions (CD) **RemoteLUName**.

The following rules apply to all forms of remote terminal definitions:

- You must associate the terminal definition with a region whose NETNAME is the **RemoteLUName** (or APPLID) of the terminal-owning region.
- You must always specify the real terminal identifier, either directly or by means of an alias.

Defining terminals to the application-owning region

If you are not going to ship the terminal definition from the terminal-owning region to the application-owning region, you must define the terminal on the application-owning region. The following attributes are required:

Terminal identifier
The name by which the terminal is known (see "Fully qualified terminal identifiers" on page 117). Application programs use this name in terminal control requests. In the following example, in which the application-owning region is CICS, the terminal identifier is REMT. This is the key of the WD entry for this terminal as defined on the application-owning region.

Remote name of terminal
The name by which the terminal is known on the terminal-owning region is defined by using the WD **RemoteName** attribute. In the following example, the name is LOCL. In a TXSeries for Multiplatforms terminal-owning region, this is the key of the WD entry for this terminal in the TOR.

The name by which a terminal is known in the application-owning region is usually the same as the name that is in the terminal-owning region. You can, however, choose to call the remote terminal by a different name (an alias) in the application-owning region.

If the terminal name is to be the same on both the terminal-owning region and the application-owning region, you do not need to specify the **RemoteName**.

Remote Region Name
Specify the name of the Communications Definitions (CD) entry for the connection to the terminal-owning region in the WD **RemoteSysId** attribute. This is used to route transaction output from the application-owning region back to the terminal in the terminal-owning region. In the example shown below, this name is CESA.

User Area Size
You set the **TCTUALen** in the WD entry to specify the length of the terminal user area. The value should be the same as that which is specified in the terminal's definition in its owning region. In the example, this value is 100.

NetName
The NETNAME of the terminal. This must match the NETNAME as defined for the local terminal in the remote system.

Example of a WD entry for a remote terminal

This command adds a WD entry, called REMT, in the region named cicsopen:

```
cicsadd -r cicsopen -c wd REMT RemoteSysId="CESA" RemoteName="LOCL"    \
          TCTUALen=100 NetName="TERM0001"
```

This entry is installed in an application-owning region that receives transaction routing requests from a remote terminal LOCL. The entry enables routing of data from the application-owning region across connection CESA back to the terminal-owning region.

Using the alias for a terminal in the application-owning region

When an ISC session is being used to provide the transaction routing services between the terminal-owning-region region (TOR) and the application-owning-region (AOR), the name by which a terminal is known in the AOR is designated as the alias for the fully qualified terminal name used in the TOR.

Usually, the alias that is on the AOR is the same as the TERMID that is on the TOR. However, if two or more TORs use similar sets of terminal identifiers (TERMIDs) for transaction routing, a naming conflict can occur. When the name on a shipped terminal definition conflicts with the name of a surrogate terminal that is already installed in the AOR, CICS assigns a randomly-generated alias to the duplicate TERMID to resolve the naming conflict. In this case, the value of the original TERMID is stored in the **RemoteName** field.

For example, the TOR named *Region1* has a terminal with the TERMID *T001*. When this terminal is shipped, it has the fully qualified name of *Region1.T001*, the alias of *T001*, and the **RemoteName** is set equal to NULL on the AOR. If the AOR then receives a terminal request from *Region2* that is also using the TERMID *T001*, the AOR uses the fully qualified name of *Region2.T001*. It assigns a randomly generated alias, for example, *XXXX*, in place of the duplicate TERMID, and the value of the **RemoteName** field is set to the name of the original TERMID, *T001*.

You must take care to ensure that the correct TERMID is used when a user on the AOR issues the CECI START TRAN (*tranName*) TERMID (*T001*) command. The results of the transaction are returned to the terminal with the fully qualified name of *Region1.T001*.

If the actual intention is to return the results of the transaction to the terminal that has the fully qualified name of *Region2.T001*, the alias *XXXX* needs to be identified as the TERMID. The user can get the correct TERMID by using the CICS-supplied transaction CEMT INQUIRE TERMINAL or CEMT INQUIRE NETNAME.

Similarly, if an application on the AOR is using the command EXEC CICS START TRAN (*tranName*) TERMID (*termId*) and the possibility exists for duplicate TERMIDs, the application must get the required alias that is to be used as the TERMID, by issuing an EXEC CICS INQUIRE NETNAME command or an ASSIGN FACILITY command.

Defining remote transactions for transaction routing

A remote transaction for CICS transaction routing is a transaction, which is owned by another region (the application-owning region), that can be run from the local region by using a terminal that is owned by the local region (the terminal-owning region).

You define a remote transaction in the same way as that in which you define a local transaction, except that some of the operands are not required.

For details of all the attributes that define transactions, see the *TXSeries for Multiplatforms Administration Reference*.

TD attributes for remote transactions

To support transaction routing, you must define the remote transaction to the local region by using a Transaction Definitions (TD) entry. The name of the TD entry is the local name for the transaction. In the example in "Example of a TD entry for a remote transaction" on page 120, the local name of the transaction is LTRN.

Connection to the remote system

Set the **RemoteSysId** attribute in the TD to specify the name of the

Communications Definitions (CD) entry of the connection to the remote region in which the transaction resides. In the example, the name of the connection is CESA.

Remote transaction name
Specify the name by which the transaction is known in the remote region, by using the **RemoteName** attribute in the TD. In the example, this name is RTRN. If the transaction name is to be the same as both the local and remote region, you do not need to specify a remote transaction name.

CICS translates transaction names from local names to remote names when a request to run a transaction is transmitted from one region to another.

Transaction work area (TWA)
You can set the **TWASize** attribute in the TD to zero because the relay transaction does not require a TWA.

Transaction security
You can define the transaction level security key for this transaction with the **TSLKey** TD attribute. Specify transaction security for routed transactions that are user-initiated. You do not need to specify resource security checking on the local transaction because the relay transaction does not access resources. However, the actual TDs in the remote system need resource security checking to be set up, if required.

Example of a TD entry for a remote transaction

This command adds a TD entry, called LTRN in the region named cicsopen:

```
% cicsadd -r cicsopen -c td LTRN RemoteSysId="CESA"             \
                                RemoteName="RTRN"
```

This entry causes a request for transaction LTRN to be routed over connection CESA to a remote CICS system, in which transaction RTRN is executed.

Defining remote transactions for asynchronous processing

The only remote resource definitions that are needed for asynchronous processing are those for transactions that are named in the TRANSID option of EXEC CICS START commands. An application can use the EXEC CICS RETRIEVE command to obtain the name of a remote temporary storage queue that the transaction subsequently names in a function shipping request.

A remote transaction for CICS asynchronous processing is a transaction that is owned by another region and that runs from the local region only by means of EXEC CICS START commands.

CICS application programs can use the SYSID option to name a remote region explicitly on EXEC CICS START commands. If this is done, you need not define the remote transaction on the local region.

More generally, however, applications are designed to start transactions without being aware of their location, and in this case you must define the remote transaction in the local Transaction Definitions (TD).

Note: If the transaction is owned by another region and can be run by CICS transaction routing and by EXEC CICS START commands, you must define the transaction for transaction routing.

Remote transactions that are started only by EXEC CICS START commands require only basic information in the local Transaction Definitions (TD). This information consists of:

- **RemoteSysId**
- **RemoteName**
- **LocalQ**
- **RSLCheck**

You can specify local queuing for remote transactions that are initiated by EXEC CICS START requests.

Chapter 6. Configuring intersystem security

This chapter describes:

- How to ensure that systems that attempt to attach to the local region are authorized to do so
- How to provide permission to specific remote systems and users to have access to local resources

Overview of intersystem security

The security requirements for a region that communicates with other systems are an extension of the security requirements for a single, stand-alone region. CICS security uses the concepts of user sign-on to give a user authority to access sensitive transactions and resources. These facilities are extended for intercommunication functions to include the remote users and remote systems.

It is assumed that you are familiar with setting up security for a single region. You need to understand:

- How to define users to CICS by generating User Definitions (UD) entries
- What it means to authenticate a user on your local system
- How transaction security can be used to restrict a user's ability to run CICS transactions
- How resource security can be used to restrict a user's access to resources managed by the region

For more information about these topics, see the *CICS Administration Guide*.

CICS assumes that it is the responsibility of every system to verify the authenticity of all requests that it receives. These inbound requests can be received from individual users or remote systems. The remote system can be another CICS system or a non-CICS system.

Implementing intersystem security

Some initial effort is required to determine how to set up intercommunication in accordance with the policies of your local system, and the privileges that the users of your region require. However, after the security checking is set up, CICS security checking operates automatically without the need of remote users or application programs to take specific security actions.

The security checks that TXSeries for Multiplatforms supports are based on Systems Network Architecture (SNA) LU 6.2 security services. Therefore, they are available for controlling access from SNA-connected systems. Where appropriate, these services have also been extended to systems that are connected by TCP/IP.

When you configure your CICS system, you define all the remote systems from which you want to accept intercommunication requests in Communications Definitions (CD) entries. Define the security levels that you are prepared to give to those remote systems with the **RemoteSysSecurity**, **LinkUserid**, **RSLKeyMask**, and **TSLKeyMask** attributes in the CD entry. However, before accepting requests, your system must first verify that the remote system is actually the system that it claims to be. This verification process is known as authentication. The mechanisms that are available to do this depend upon the type of network connection.

Summary of CICS intersystem security

The security checks that CICS performs do the following:
- Restrict the access of remote systems to the CICS region
- Restrict the access of requests to CICS resources
- Restrict the access of particular users

The following sections describe how to set up these security checks:
- "Identifying the remote system"
- "Authenticating systems across CICS family TCP/IP connections"
- "Authenticating systems across CICS PPC TCP/IP connections"
- "Authenticating systems across SNA connections" on page 125
- "Authenticating systems across PPC Gateway server connections" on page 126
- "CICS link security" on page 126
- "CICS user security" on page 127

"Authenticating systems across SNA connections" on page 125, "CICS link security" on page 126, and "CICS user security" on page 127 tell you how to set points at which you can apply security checks in the processing of an incoming request. Review the description of security for local regions in *CICS Administration Guide* before you attempt to set up security for intercommunications. Communications Definitions (CD) are described in the *TXSeries for Multiplatforms Administration Reference*.

Identifying the remote system

The first security problem that your CICS system has to resolve is to determine correctly the identity of the remote system that is attempting to initiate an intercommunication request. You must ensure that an alien remote system cannot impersonate another.

The ways that are available to CICS to identify a remote system depend upon the network protocol that is used in the connection.

Authenticating systems across CICS family TCP/IP connections

When an intersystem request is received on a CICS family TCP/IP connection, your region *cannot* verify the identity of the remote system. No mechanism is available that enables you to detect when an unauthorized system has deliberately impersonated another.

CICS can extract the Internet Protocol (IP) address and port number of the remote system, but this is easy for an alien system to imitate.

If you have defined a Listener Definitions (LD) entry to allow CICS family TCP/IP connections, and unless your TCP/IP network is private and secure, define the security attributes in the CD entry on the assumption that the identity of the remote system has not been verified.

Authenticating systems across CICS PPC TCP/IP connections

Because CICS PPC TCP/IP can be configured to connect the regions on same machine only, no authentication service is required.

Authenticating systems across SNA connections

When a remote system uses an SNA connection to communicate with your CICS system, it must first establish a *session* with your system. That session is created by an exchange of flows called a *BIND*. You can associate a password with the BIND. This process is known as *bind-time security*, or *LU-LU verification*. It enables each system to verify the identity of the other.

These passwords are not sent between the two systems. Each system demonstrates its knowledge of the password by being able to correctly encrypt random numbers that are supplied by the partner, using the password as a key. The bind is successful only when both systems can establish that they have the same password.

Figure 50 shows the SNA flows that are exchanged to support bind-time security. If either system discovers that the encrypted value received is not the value that is expected, it flows an SNA UNBIND request to the remote system, and a session is not established.

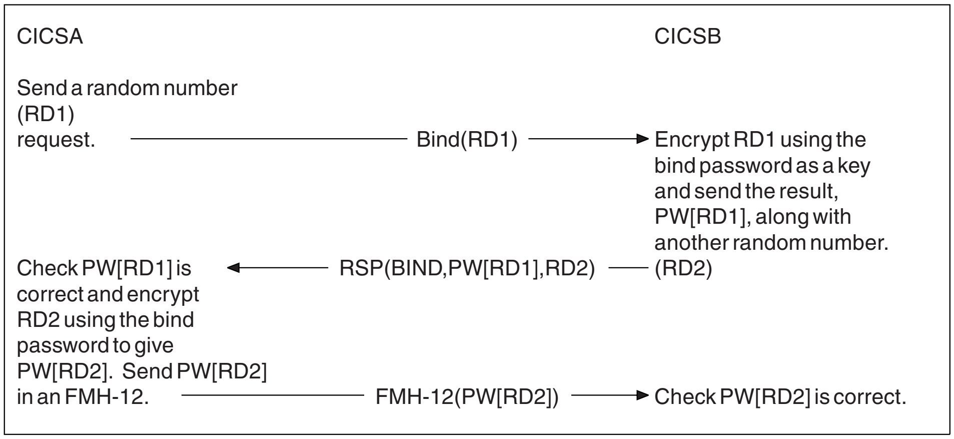

CICSA
CICSB
Send a random number (RD1) request.
Bind(RD1)
Encrypt RD1 using the bind password as a key and send the result, PW[RD1], along with another random number. (RD2)
RSP(BIND,PW[RD1],RD2)
Check PW[RD1] is correct and encrypt RD2 using the bind password to give PW[RD2]. Send PW[RD2] in an FMH-12.
FMH-12(PW[RD2])
Check PW[RD2] is correct.

Figure 50. The bind password exchange

Bind passwords are set up in the SNA product that is managing your SNA connectivity. Refer to your SNA product documentation for a description of how to set the bind password for a connection.

Notes:

1. Bind-time security is optional in the SNA LU 6.2 architecture. Because it is optional, the remote systems to which you are connecting might not support BIND passwords.
2. To maintain maximum confidence in the identity of each connected system, it is recommended that different bind passwords be used between each pair of systems that you are configuring. However, when the number of systems grows, this might become unmanageable. Therefore, unique bind passwords are not a requirement of the SNA LU 6.2 architecture and so are not enforced.

It is important that you are familiar with the descriptions of bind security that are given in the documentation for the SNA product that you are using. Refer to the SNA books that are listed in “Bibliography” on page 355.

Authenticating systems across PPC Gateway server connections

If you are using the PPC Gateway server to support SNA connections to remote systems, CICS issues a CICSIPC Remote Procedure Call (RPC) to contact the PPC Gateway server (which can be configured on the same machine). Similarly, the PPC Gateway server uses an RPC to schedule an inbound intersystem request from an SNA remote LU. Since the PPC Gateway and the CICS region are in same machine, there is no need for authenticated RPCs.

CICS link security

You can define security levels to the transactions and resources in your CICS system that apply to all intercommunication requests that are received from a particular system. This form of security is known as *link security*.

To use link security, you must have a security manager. CICS has its own security manager, but if your operating system supports an external security manager that TXSeries for Multiplatforms supports, you can use that instead of, or in conjunction with, the CICS internal security manager. An external security manager is a user-supplied program that allows you to define your system's own security mechanism for preventing unauthorized user access to resources from application programs and the unauthorized initiation of CICS transactions.

The sections that follow help you implement CICS internal security, which uses Transaction Security Level (TSL) and Resource Security Level (RSL) keys to restrict access. For information about using external security managers, refer to the *CICS Administration Guide*.

The security keys that are defined for link security apply to all requests that are received from a particular remote system. This means that the list of security keys must include *all* the keys that are needed by *every* user from the remote system. If the needs of the users from a remote system vary, this list of security keys might give more access to some users than is needed. If this is not acceptable, consider using the security that is described in "CICS user security" on page 127, which allows you to set up security keys based not only on the system that sent the request, but also on the user who is associated with that request also.

If link security is enough, it can be set up as follows:

1. Set the Communications Definitions (CD) **RemoteSysSecurity** attribute to **local** for the connection to the system for which you want link security implemented.
2. Specify a link user ID for the connection with the CD **LinkUserId** attribute. This user ID is associated with any request that is received from the remote system that is at the other end of the connection.
3. Create a User Definitions (UD) entry for the connection's link user ID. Use the UD **TSLKeyList** and the **RSLKeyList** attributes to specify the transactions and resources that can be accessed by inbound requests.

An alternative implementation of link security follows:

1. Set the CD **RemoteSysSecurity** attribute to **local** for the connection to the system for which you want link security implemented.
2. *Do not* specify a link user ID for the connection with the CD **LinkUserId** attribute.
3. Use the CD **TSLKeyMask** and **RSLKeyMask** attributes to define the TSL and RSL keys to use for the connection.

These two methods differ as follows:

- When a link user ID is specified for the connection, the user is logged on as the connection's link user ID, and the keys that are defined for the link user ID are used. This is the preferred method.
- When a link user ID is not specified for the connection, the user is logged on as the region's default user ID, and the connection's **TSLKeyMask** and **RSLKeyMask** keys are used. Any keys that are defined for the region's default user ID are ignored.

In either case, you must have TSL and RSL keys assigned to the resources and transactions for which you want link security applied. For further information, see the discussion of the **RSLKey** and **TSLKey** settings in "Using CRTE and CESN to sign on from a remote system" on page 142.

CICS user security

CICS user security provides a more granular security checking than link security does because it allows you to base the TSL and RSL keys that apply to inbound requests not only on the remote system but also on the user who originated the request. It is done by setting up the remote system to flow the user ID of the user with the intersystem request. When the request is received from the remote system, your region provides access to any transaction or resource that matches the security keys that are defined for both the flowed user ID and the connections link user ID. These keys are defined with the User Definitions (UD) **TSLKeyList** and **RSLKeyList** attributes.

Note: A link user ID is specified with the Communications Definitions (CD) **LinkUserId** attribute. If a link user ID is not specified, (that is, if the setting is **LinkUserId=""**), the keys that are defined with the CD **TSLKeyMask** and **RSLKeyMask** attributes are used in conjunction with the flowed user ID instead.

Combining the keys for the link with the keys for the user ensures that remote users who have the same user ID as do local users (or remote users from other systems) can be set up with different access privileges.

Two ways are possible to set up user security:

Method one:

1. Ask the administrator of the remote system to generate the following:
   a. A list of the user IDs of the remote users from the remote system who will be accessing your region and the security keys that each of them will need. Then, generate a list of security keys that consists of all the security keys that these users require. (You can define them in two ways. Descriptions follow.) These keys are known as the *link keys* for the system.

      Refer to Table 26 on page 128. In this table, user TOM from SYS2 needs access to resources that have an RSL key of 2 assigned to them, and user DICK from SYS2 needs access to resources that have an RSL key of 3 assigned to them. This means SYS2 has link keys of 2 and 3, as shown in the "Composite link keys" column:

Table 26. Security keys assigned to example systems

Remote systems	**SAM RSLKeys**	**FRED RSLKeys**	**TOM RSLKeys**	**DICK RSLKeys**	**HARRY RSLKeys**	**Composite link keys**
Keys for SYS2			2	3		2|3
Keys for SYS3	6	6	3	4	5|6|7|8	3|4|5|6|7|8
Keys for SYS4			8	9	10	8|9|10
Keys for SYS5				4	4|6	4|6

b. A list of those user IDs that appear in both the remote system and your local system, irrespective of whether they will be used to access your region from the remote system. For each of these user IDs, check the level of security if they access your region from the remote system. To do this, compare the keys that are common to the local User Definitions (UD) entry for the user ID with the link keys, as calculated in step 1a on page 127. Then, look at the transactions and resources to which these specifications give them access. If the results are unacceptable, either the keys that are assigned to your local transactions and resources must be changed, or the user ID in one of the systems must be renamed. Refer to Table 27.

Table 27. Security keys assigned to example users

Systems	**SAM RSLKeys**	**FRED RSLKeys**	**TOM RSLKeys**	**DICK RSLKeys**	**HARRY RSLKeys**
Keys for SYS1 (local users)	4|6	4|6			
Keys for SYS2 (remote)			2	3	
Keys for SYS3 (remote)	6	6	3	4	5|6|7|8
Keys for SYS4 (remote)			8	9	10
Keys for SYS5 (remote)				4	4|6

In this example, user IDs SAM and FRED appear on the local system (SYS1) and one of the remote systems (SYS3). Both users who are on the local system are assigned **RSLKey** values of 4 and 6. SAM and FRED from SYS3 require access to resources that are identified by an **RSLKey** of 6. However, because SAM and FRED are already defined on the local system as having access to resources that are identified by an **RSLKey** of 4 and 6, and because the link for SYS3 requires **RSLKey**s of 3|4|5|6|7|8, SAM and FRED from SYS3 will be given access to resources that are identified by an **RSLKey** of 4 in addition to 6. Remember that the keys are not flowed, only the user IDs. If you do not want SAM and FRED to have access to resources with **RSLKey**s 4 and 6 on your local system (ISC1), do not add these keys to the **RSLKeyList**s for SAM and FRED in the UD entries on the local system. Consider also the access that is automatically given to SAM or FRED if they are added to SYS5 after security is defined for SYS1 and if the administrators of SYS1 are not made aware of the change.

Note: You need to go through this exercise each time that a new user ID is added to any of the remote systems to which you have granted **trusted** access.

2. Set up the remote system to flow user IDs. This is described in:
   a. "Setting up a CICS region to flow user IDs" on page 130.
   b. "Receiving user IDs from SNA-connected systems" on page 133.

3. Set the CD **RemoteSysSecurity** attribute to **trusted** or **verify** for the connection to the remote system. Refer to “Setting the RemoteSysSecurity attribute to trusted or verify” on page 130 for an explanation of the differences between **trusted** and **verify**.
4. Define User Definitions (UD) for each of the user IDs that were identified in step 1a on page 127.
5. Define a UD entry for the name of the remote system. This is important because some CICS tasks (which are initiated by the remote system) can run with the remote system name as the user ID. It is recommended that this user ID is used *only* for this purpose (that is, it is not used as a normal user ID). The default attribute values are enough for this UD entry.
6. Define a UD for the link user ID. This user ID represents the remote system and requires the keys as calculated in the “Composite Keys” column for that system.
7. Set the CD **LinkUserId** attribute to the user ID that is defined in step 5.

Method two:

1. Ask the administrator of the remote system to generate the lists of user IDs, as described in steps 1a and 1b in the previous list.
2. Set up the remote system to flow user IDs. Refer to step 2 in the previous list for references to information that tells you how to do this.
3. Set the CD **RemoteSysSecurity** attribute to **trusted** or **verify** for the connection to the system for which you want user security implemented.
4. Define User Definitions (UD) for each of the user IDs that are identified in step 1a of the previous list.
5. Define a UD entry for the name of the remote system. This is important because some CICS tasks (which are initiated by the remote system) can run with the remote system name as the user ID. It is recommended that this user ID is used *only* for this purpose (that is, that it is not used as a normal user ID). The default attribute values will be enough for this UD entry.
6. *Do not* specify a link user ID for the connection with the CD **LinkUserId** attribute.
7. Use the CD **TSLKeyMask** and **RSLKeyMask** attributes to define the TSL and RSL keys to use for the connection.

The differences between these two methods:

- When a link user ID is specified for the connection, the user is logged on as the flowed user ID, and the keys that are defined for both the link user ID and the flowed user ID are used. This is the preferred method.
- When a link user ID is not specified for the connection, the user is logged on as the flowed user ID, and the **TSLKeyMask** and **RSLKeyMask** keys that are defined for both the connection and the flowed user ID are used.

In either case, you must have TSL and RSL keys assigned to the resources and transactions for which you want user security applied, as described in the discussions of the **RSLKey** and **TSLKey** attributes in “Using CRTE and CESN to sign on from a remote system” on page 142.

This description provides a simple approach to using the resource definitions to set up user security. “Using CRTE and CESN to sign on from a remote system” on page 142 provides further examples and descriptions of how CICS security resources definitions can be used.

Setting the RemoteSysSecurity attribute to trusted or verify

The **RemoteSysSecurity** attribute can be set to either **trusted** or **verify**.

Use the value **trusted** as follows:

- When the remote system does not send passwords with user IDs. This applies to CICS Transaction Server for z/OS systems. TXSeries for Multiplatforms systems might send passwords, but their default behavior is not to send them.
- When the remote SNA system does send passwords, but the SNA product that you are using verifies the password and does not pass it to CICS. Refer to "Receiving user IDs from SNA-connected systems" on page 133 for further details about this.

Use the value **verify** as follows:

- To ensure that user IDs are always accompanied by a password
- When you are using an SNA product that does pass the password to CICS

Use **verify** when using CICS family TCP/IP connections to CICS OS/2.

Setting up a CICS region to flow user IDs

A region flows users IDs to a remote system if the **OutboundUserIds** attribute of the local CD entry for the remote system is set to **sent**, **sent_maybe_with_password**, or **sent_only_with_password**. If the remote system to which you are sending cannot receive inbound user IDs, set the **OutboundUserIds** attribute to **not_sent**.

Table 28 shows which user ID is flowed when a local task issues an intersystem request to a remote system.

Table 28. Which user ID is flowed when local task issues a request

Characteristics of the local task	**User identifier sent by CICS to the remote region**
Task with associated terminal; user signed on	Terminal user identifier.
Task with associated terminal; no user signed on	The region's **DefaultUserId**.
Task with no associated terminal started by interval control EXEC CICS START	User identifier for the task that issued the EXEC CICS START command.
Task with no associated terminal, triggered by a transient data queue	User Identifier whose TDQ write triggered the task.
CICS system task	Local **Sysid** of the region.
Program started by Distributed Program Link (DPL)	The user ID is the one associated with the task. This depends on the CD **RemoteSysSecurity** and **LinkUserId** settings for the connection that started the task. For example: **RemoteSysSecurity=local**: either link user ID or default user ID. **RemoteSysSecurity=trusted** or**verify**: either flowed user ID or default user ID.

Table 28. Which user ID is flowed when local task issues a request (continued)

Characteristics of the local task	**User identifier sent by CICS to the remote region**
Back-end Program started by Distributed Transaction Processing (DTP).	The user ID is the one that is associated with the task. This depends on the CD **RemoteSysSecurity** and **LinkUserId** settings for the connection that started the task. For example: **RemoteSysSecurity=local**: either link user ID or default user ID. **RemoteSysSecurity=trusted** or **verify**: either flowed user ID or default user ID.

By default, CICS never flows passwords with these user IDs. This is because all local user IDs that are running transactions in a region have had their passwords checked. Therefore, set up remote systems in the SNA definitions to accept user IDs that are flowed by a region as **already_verified** See “Receiving user IDs from SNA-connected systems” on page 133 for more information.

If you want your CICS region to send passwords, see “Setting up a CICS region to flow passwords.”

Setting up a CICS region to flow passwords

It can sometimes be necessary for the local CICS region to send a password and user ID to a remote system. This can occur if the CICS region is acting as a client gateway to a CICS for MVS/ESA host, and you want to control all security with RACF on the host. It can also be needed when you need to implement user security, but your SNA product (such as the Microsoft Microsoft SNA Server for Windows) does not support sending already_verified user IDs.

To configure CICS to send passwords:

1. Create a DFHCCINX user exit that will cause CICS to save passwords received from clients.
2. Configure the CD entry for the connection to the remote system to enable the local region to send the password.

Notes:

1. Whenever CICS saves the password in storage, it encrypts the password. However, if SNA is used to flow passwords, they are sent over the SNA network in plain text as required by the SNA architecture.
2. Only the Universal Client software can be used when the user ID and password are to flow to another system.

The DFHCCINX parameters that determine whether to save the password are:

- `CICS_CCINX_PSWD_CHECK_AND_DROP` (the default)
- `CICS_CCINX_PSWD_CHECK_AND_KEEP`
- `CICS_CCINX_PSWD_IGNORE_AND_DROP`
- `CICS_CCINX_PSWD_IGNORE_AND_KEEP`

If you want to use any of these settings, you must also set `RemoteSysSecurity` to `CICS_CCINX_SECURITYTYPE_VERIFY`.

The CD parameters that determine whether to send the password to the remote systems are:

```
OutboundUserIds=sent_only_with_pswd
OutboundUserIds=sent_maybe_with_pswd
```

These are described in "Writing your own version of DFHCCINX" on page 62. The following sections describe some scenarios.

To send passwords after local verification

This is needed if the local SNA product does not support sending already_verified user IDs; for example, when using Microsoft Microsoft SNA Server. The password is verified in the local region, then flowed to the remote region.

```
DFHCCINX:
  RemoteSysSecurity=(CICS_CCINX_SECURITYTYPE_VERIFY|
                     CICS_CCINX_PSWD_CHECK_AND_KEEP)
CD:
     OutboundUserIds=sent_only_with_pswd
```

To send passwords without local verification

Sending passwords without local verification might be needed when you are using the CICS region as a client gateway. The local CICS region does not verify the password before sending it to the remote system.

```
DFHCCINX:
 RemoteSysSecurity=(CICS_CCINX_SECURITYTYPE_VERIFY|
                    CICS_CCINX_PSWD_IGNORE_AND_KEEP)

CD:
    OutboundUserIds=sent_only_with_pswd
```

If the local CICS region does not receive a password from the client, no user ID is sent to the remote system.

Alternatively, the CD can be configured as follows:

```
CD:
   OutboundUserIds=sent_maybe_with_pswd
```

Then if the local CICS region does not receive a password from the client, the user ID is sent to the remote system as already_verified).

Do not send passwords

If you do not want the local CICS region to send passwords to a remote system:

```
DFHCCINX:
   RemoteSysSecurity=(CICS_CCINX_SECURITYTYPE_VERIFY|
                      CICS_CCINX_PSWD_CHECK_AND_DROP)

CD:
     OutboundUserIds=sent, or
     OutboundUserIds=not_sent
```

This is the normal default operation, where the local CICS region verifies the passwords but does not save them, and therefore they cannot be sent to a remote system. This is the most secure setup.

No password checks

If you are operating in a secure environment, you might want to disable all security. Passwords are not verified in the local CICS region and are not be sent to the remote system.

```
DFHCCINX:
   RemoteSysSecurity=(CICS_CCINX_SECURITYTYPE_VERIFY|
                      CICS_CCINX_PSWD_IGNORE_AND_DROP)

CD:
   OutboundUserIds=not_sent
```

Receiving user IDs from SNA-connected systems

The SNA LU 6.2 architecture has optional support for flowing user IDs and passwords between SNA-connected systems. This is called *conversation-level security* (or sometimes *attach security*, or *conversation security*). Conversation-level security is provided in the 212, 213, 217, 218, 219, 220, 221, and 222 SNA LU 6.2 option sets. Because this security is optional, each system needs a definition of the level of security that it can, or wants to, accept from a particular remote system. You can set up a different conversation *security acceptance* level on each end of a connection between two systems.

The possible levels of security acceptance are:

- Conversation-level security not supported or required.

  This option means that user IDs and passwords must not be sent to this system. If they are sent, it is considered an SNA protocol violation.
- Conversation-level security supported.

  This option means that user IDs can be sent to this system, but in order for them to be accepted, they must be accompanied by a valid password. Select this level if you want to receive user IDs from a system that does not have its own security manager and so cannot verify its own users (as with CICS OS/2).
- Already verified supported.

  This option means that user IDs can be sent to this system and these user IDs do not need to be accompanied by a password because the remote system is trusted to have verified the user ID against a password before the intersystem request was sent. CICS Transaction Server for z/OS, CICS/ESA, CICS/MVS, CICS/VSE, and TXSeries for Multiplatforms always send verified user IDs, and a remote SNA system that wants to receive user IDs from these systems must accept already verified user IDs.
- Persistent verification supported.

  This option allows the verification that is associated with a user ID and password pair to persist over a number of intersystem requests. Both the user ID and password are sent on the first request. However, if they are valid, only the user ID is required on subsequent requests. The user ID can be sent without a password for a user-defined period of time, or until the initiating system sends a sign-off request.
- Both already verified and persistent verification supported.

  This option allows both already-verified requests and requests that use persistent verification.

The security acceptance levels that are available to you depend on the support that your local SNA product provides.

When using Communications Server for AIX

For example, Communications Server for AIX supports three values for the *Security acceptance* field in LU 6.2 partner LU definition. These are *none* (for no conversation-level security), *conversation* (for conversation-level security) and *already_verified* (for already verified support).

CICS Transaction Server for z/OS 4.1, which provides its own SNA services, supports all five levels of security acceptance. These are specified in the ATTACHSEC option of the CICS Transaction Server for z/OS CONNECTION definition. Table 29 shows how they compare.

Table 29. Comparing security options between platforms

Security acceptance level	**No conversation-level security**	**Conversation-level security**	**Already verified support**	**Persistent verification**	**Already verified and persistent verification**
Communications Server for AIX	None	Conversation	already_verified	See Note 1	See Note 1
Communications Server for Windows	See Note 1	Deselect **Conversation Security Support**	Select **Conversation Security Support**	See Note 1	See Note 1
Microsoft SNA Server	Deselect **Conversation Security**	Select **Conversation Security**	Select **Accepts Already Verified user Names**	See Note 2	See Note 2
HP-UX SNAplus2	None	Conversation	conv-security-ver	See Note 1	See Note 1
SNAP-IX for Solaris	None	Conversation	conv-security-ver	See Note 1	See Note 1

Notes:

1. Not supported with this SNA product.
2. Microsoft Microsoft SNA Server uses the values in the transaction definition created by **cicssnatpns**.

These security levels define the security options that a system supports, rather than the level of security that is expected on each intersystem request. For example, an intersystem request that has no user ID associated with it can be received from a remote system on which you have requested “already verified”.

The SNA LU 6.2 architecture specifies that the level of security that is required for an intersystem request is dependent on the transaction name that is requested. This is because it is recognized that some transactions are more sensitive than others are.

CICS uses transaction security level (TSL) and resource security level (RSL) keys to provide security checking at the transaction and resource level.

When using Communications Server for AIX

Communications Server for AIX also allows you to define a **SecurityRequired** level in the TPN profile that you define for your CICS transactions.

Because CICS provides security support, the **SecurityRequired** attribute is not required and can be set to **none** without security exposure.

The security acceptance level that you define locally is sent to the remote SNA system as one of the parameters of the session bind request and response. Systems such as CICS Transaction Server for z/OS, CICS/ESA, CICS/MVS and CICS/VSE, which have fully integrated SNA support and so have access to the bind, can examine the security acceptance level received and decide from that whether to send user IDs. Systems such as CICS OS/2 and TXSeries for Multiplatforms, which use a separate SNA product, require that you set them up to send user IDs. (See "Setting up a CICS region to flow user IDs" on page 130.)

It is important that the security acceptance level that is defined in the remote system matches the setup that is defined in the region. Table 30 shows examples of the security parameters that are defined to flow user IDs between a region and CICS Transaction Server for z/OS.

Table 30. Defining security parameters

Types of definitions	**CICS Transaction Server for z/OS**
Security setup on the remote system	CONNECTION ATTACHSEC=IDENTIFY . Because the local region and CICS Transaction Server for z/OS do not need to send passwords with user IDs, each remote system must be set up to accept already-verified user IDs.
Communications Server for AIX LU 6.2 partner LU profile for the remote system	. Security accepted= already_verified . Because the local region and CICS Transaction Server for z/OS do not need to send passwords with user IDs, each remote system must be set up to accept already-verified user IDs.
Local HP-UX SNAplus2 LU 6.2 Remote APPC LU definition	. con-security-ver conversation security .
Local SNAP-IX Remote APPC LU definition	. con-security-ver conversation security .
IBM Communications Server for Windows	. Select conversation security .

Table 30. Defining security parameters (continued)

Types of definitions	CICS Transaction Server for z/OS
Microsoft Microsoft SNA Server	. Select Accepts Already Verified User Names .

CD entry on the local region for the remote system	.
	RemoteSysSecurity=trusted
	OutboundUserIds=sent
	LinkUserId=<>
	.

The *CICS Administration Guide* has information about security for local regions.

Link security and user security compared

Link security and user security are defined with the following resource definition attributes. The actual TSL and RSL keys that are assigned to remote users are based on a combination of how these attributes are defined. The following list describes the attributes individually. "How the resource definition security attributes are used" on page 137 describes how these attributes are used in conjunction with each other.

The RSLKey attribute

The resources that are represented by the following resource definitions can be assigned a resource security level (RSL) key by use of the **RSLKey** attribute. This key is used to determine who has access to the resource.

- File Definitions (FD): Allows a user to access the file
- Journal Definitions (JD): Allows a user to write to the journal
- Program Definitions (PD): Allows a user to run the program
- Transaction Definitions (TD): Allows a user to issue EXEC CICS START for the transaction
- Transient Data Definitions (TDD): Allows a user to access the queue
- Temporary Storage Definitions (TSD): Allows a user to access the queue

The TD TSLKey attribute

Transactions can be assigned a transaction security level (TSL) key by use of the **TSLKey** attribute. This key is used to determine who can execute the transaction.

The User Definitions (UD) TSLKeyList and RSLKeyList attributes

These attributes contain the list of TSL and RSL keys that are defined for a user. These keys allow the user access to the resources and transactions that have the same TSL and RSL keys defined for them.

The Communications Definitions (CD) RemoteSysSecurity attribute

This attribute specifies whether to use link security or user security.

The CD LinkUserId attribute

This attribute specifies a link user ID for the connection. The UD **TSLKeyList** and **RSLKeyList** attributes that are defined for the connection's link user ID are used to determine which resources requests from this connection can access.

The CD TSLKeyMask and RSLKeyMask attributes
These attributes contain the list of TSL and RSL keys that control access to transactions and resources for the connection. These attributes are used when a link user ID is not defined for the connection.

The Region Definitions (RD) DefaultUserId attribute
This attribute specifies a default user for the region. This default user is used when a user ID is required but one is not available. UD **TSLKeyList** and **RSLKeyList** attributes that are defined for the region's default user are used to determine which resources this user ID can access.

The Communications Definitions (CD) OutboundUserIds attribute
This attribute specifies whether a user ID is to be sent on the outbound request.

How the resource definition security attributes are used

The following maps show how the level of security is determined. They show:

- Which user ID is used when the remote user logs in
- Which security keys are used

Map 1. Start

1. **Is RemoteSysSecurity=local for this connection**

   YES Go to "Map 2. RemoteSysSecurity=local (Link Security)."

   NO Skip to the next question.

2. **If RemoteSysSecurity=trusted for this connection**
   - **Is a user ID flowed from the remote system with this request?**

     YES Go to "Map 4. RemoteSysSecurity=trusted and a user ID is Flowed (User Security)" on page 138.

     NO Go to "Map 3. RemoteSysSecurity=trusted and a user ID is not Flowed (User Security)" on page 138.

3. **If RemoteSysSecurity=verify for this connection**
   - **Is a user ID flowed from the remote system with this request?**

     YES Go to "Map 6. RemoteSysSecurity=verify and a user ID is Flowed (User Security)" on page 138.

     NO Go to "Map 5. RemoteSysSecurity=verify and a user ID is not Flowed (User Security)" on page 138.

Map 2. RemoteSysSecurity=local (Link Security)

1. **Is a link user defined for this connection?**

   YES Skip to the next question.

   NO CICS logs the user in as the region's default user, and uses the keys that are defined in the CD attribute **TSLKeyMask** and **RSLKeyMask**. CICS ignores TSL and RSL keys that might be defined for the default user.

2. **Is a UD entry defined for the connection's link user?**

   YES CICS logs the user in as the connection's link user and uses the TSL and RSL keys that are defined in the UD entry for the connection's link user.

   NO CICS logs the user in as the connection's link user and grants public access.

Map 3. RemoteSysSecurity=trusted and a user ID is not Flowed (User Security)

1. Because no user ID is supplied, the default user ID is used.
2. **Is a UD entry defined for the region's default user?**

   YES Skip to the next question.

   NO CICS logs the user in as the region's default user and grants public access.
3. **Is a link user defined for this connection?**

   YES Skip to the next question.

   NO CICS logs the user in as the region's default user and uses the TSL and RSL keys that are defined both for the default user and for the connection's **TSLKeyMask** and **RSLKeyMask**.
4. **Is a UD entry defined for the connection's link user?**

   YES CICS logs the user in as the region's default user and uses the TSL and RSL keys that are defined both for the default user and for the connection's link user.

   NO CICS logs the user in as the region's default user and grants public access. CICS ignores TSL and RSL keys that might be defined for the default user.

Map 4. RemoteSysSecurity=trusted and a user ID is Flowed (User Security)

1. **Is a UD entry defined for the selected user ID?**

   YES Skip to the next question.

   NO CICS logs in as the selected user ID and grants public access.
2. **Is a link user defined for this connection?**

   YES Skip to the next question.

   NO CICS logs in as the flowed user ID and uses the TSL and RSL keys that are defined both for the flowed user ID and for the connection's **TSLKeyMask** and **RSLKeyMask**.
3. **Is a UD entry defined for the connection's link user?**

   YES CICS logs in as the flowed user ID and uses the TSL and RSL keys that are defined both for the flowed user ID and for the connection's link user.

   NO CICS logs in as the flowed user ID and grants public access. Ignore TSL and RSL keys that might be defined for the flowed user ID.

Map 5. RemoteSysSecurity=verify and a user ID is not Flowed (User Security)

1. CICS logs the user in as the region's default user and uses the TSL and RSL keys that are defined both for the default user and for the connection's **TSLKeyMask** and **RSLKeyMask**.

Map 6. RemoteSysSecurity=verify and a user ID is Flowed (User Security)

1. **Is a password supplied?**

   YES If the password is correct, the flowed user ID is selected. If the password is not correct, use the default user ID.

NO The default user ID is selected.

2. **Is a UD entry defined for the selected user ID?**

   YES Skip to the next question.

   NO CICS logs in as the selected user ID and grants public access.

3. **Is a link user defined for this connection?**

   YES Skip to the next question.

   NO CICS logs in as the selected user ID and uses the TSL and RSL keys that are defined both for the user ID and for the connection's **TSLKeyMask** and **RSLKeyMask**.

4. **Is a UD entry defined for the connection's link user?**

   YES CICS logs in as the selected user ID and use the TSL and RSL keys that are defined both for the flowed user ID and for the connection's link user.

   NO CICS logs in as the selected user ID and grants public access. Ignore TSL and RSL keys that might be defined for the flowed user ID.

Examples of link and user security

Table 31 on page 140 contains resource definitions for the example system, SYS1. Assume that defaults are used where attributes are not shown in these example definitions. For example, **TSLKeyMask=none** and **RSLKeyMask=none** for SYS2, SYS3, and SYS5.

For more information, see "Authenticating systems across SNA connections" on page 125 and "CICS link security" on page 126.

The remote systems that are involved are represented by the following CD that reside on SYS1:

SYS2 This system has the following users on it:

SAM Who needs access to TRN6, PROG6, and FILE6 on SYS1.

FRED Who needs access to TRN6, PROG6, and FILE6 on SYS1.

TOM Who needs access to TRN3, PROG3, and FILE4. FILE5, FILE7 and FILE8 on SYS1.

LINK3
Who does not require access to any of the resources on SYS1.

SYS3 This system has the following users on it:

DICK Who needs access to TRN4, PROG4, and FILE4 on SYS1.

HARRY
Who needs access to TRN4 and TRAN6, PROG4 and PROG6, and FILE4 and FILE6 on SYS1.

LINK2
Who does not require access to any of the resources on SYS1.

SYS4 This system is using link security. All users from this system are given the same level of security. These users need access to TRN4 and TRAN6, PROG4 and PROG6, and FILE4 and FILE6 on SYS1.

SYS5 This system is using link security. All users from this system are given the same level of security. These users need access to TRN6, PROG6, and FILE6 on SYS1.

The following table shows definitions on REGIONA, which is the local CICS system:

Table 31. Example resource definitions for link security

Resource definition category	**Attributes and values**
CD	• SYS2: RemoteSysSecurity=trusted LinkUserId="LINK2" (user IDs are flowed from this system) • SYS3: RemoteSysSecurity=trusted LinkUserId="LINK3" (user IDs are not flowed from this system) • SYS4: RemoteSysSecurity=local LinkUserId="" TSLKeyMask=4|6 RSLKeyMask=4|6 • SYS5: RemoteSysSecurity=local LinkUserId="LINK5"
RD	REGIONA DefaultUserId=CICSUSER
UD	• CICSUSER: TSLKeyList=4|6 RSLKeyList=4|6 • LINK2: TSLKeyList=all RSLKeyList=all • LINK3: TSLKeyList=all RSLKeyList=all • LINK5: TSLKeyList=6 RSLKeyList=6 • SAM: TSLKeyList=6 RSLKeyList=6 • FRED: TSLKeyList=6 RSLKeyList=6 • TOM: TSLKeyList=3 RSLKeyList=3|4|5|7|8 • HARRY: TSLKeyList=4|5|7 RSLKeyList=4|5|7
PD	• PROG4: RSLKey=4 • PROG6: RSLKey=6
TD	• TRN3: TSLKey=3 RSLKey=3 • TRN4: TSLKey=4 RSLKey=4 • TRN6: TSLKey=6 RSLKey=6
FD	• FILE3: RSLKey=3 • FILE4: RSLKey=4 • FILE5: RSLKey=5 • FILE6: RSLKey=6 • FILE7: RSLKey=7 • FILE8: RSLKey=8

In these examples:

- Users from SYS2 are given access to those resources that are specified by the **TSLKeyList** and **RSLKeyList** attributes as defined in UD entries on SYS1 for the individual users. The reason for this is that **RemoteSysSecurity=trusted**, user IDs are flowed from SYS2, and the **TSLKeyList** and **RSLKeyList** attributes for LINK2 (the link user ID for SYS2) specifies **all**. LINK3 is a user on SYS2. Although it is not intended that this user should have access to any resource on SYS1, LINK3 is given access to all resources.
- LINK2 from SYS3 is given access only to those resources that are assigned to the link user ID and that are also defined for the default user ID. This is true of all users from SYS3 because user IDs are not flowed. Note that this is an unusual way of setting up security. It is better to use link security. The reason for this is that HARRY is defined locally and, if user IDs started to flow from the remote system, HARRY would be given access to keys 5 and 7 in addition to 4, and HARRY would lose access to 6.

- Users from SYS4 are given access to those resources that are specified in the CD **TSLKeyMask** and **RSLKeyMask** attributes. This is because **RemoteSysSecurity=local** and a link user is **not** specified.
- Users from SYS5 are given access to those resources that are specified in the UD for LINK5. This is because **RemoteSysSecurity=local** and a link user is specified.

Security and function shipping

This section gives additional information about security for function shipping.

Security requirements for the mirror transaction

When CICS receives a function-shipped request, the started transaction is the mirror transaction. The CICS-supplied definitions of the mirror transactions (CPMI, CVMI, and CSM*) all specify Resource Security Level (RSL) checking, but the Transaction Security Level (TSL) key is set to 1, which gives public access to these mirror transactions. The mirror transactions can therefore be run by any remote region and user that have permission to access your region, but the transactions can access only those resources for which the link and remote user have authority.

You can modify the appropriate mirror Transaction Definitions (TD) to achieve the level of security you that require, or create new mirror transactions with different levels of security, based on the supplied TD. Each mirror transaction must specify **DFHMIRS** as the **ProgName** attribute.

Note: Also read “Migration considerations for function shipping” on page 345, “CICS link security” on page 126, and “CICS user security” on page 127.

Outbound security checking for function-shipping requests

If you include a remote resource in your resource definitions, you can arrange for CICS to perform security checking locally, just as if the resource is local. This check occurs before the function-shipping request is sent to the remote system and occurs independently of any checks that the remote system makes.

In addition, if you specify the **RSLCheck** attribute as **internal** or **external** in the Transaction Definitions (TD) entry for a transaction, CICS raises the NOTAUTH condition locally if the transaction attempts to issue an EXEC CICS command with the SYSID option specified. This is because the SYSID option bypasses the local security checking for resources.

Note: The CICS-supplied transaction CECI is set up with **RSLCheck=internal**. This means that it cannot be used to try out function-shipping requests using the SYSID option until the TD definition for CECI is changed to **RSLCheck=none**.

The NOTAUTH condition

If a transaction attempts to access a resource but does not satisfy the resource security checks, CICS raises the NOTAUTH condition.

If a resource is being accessed as part of a function-shipping request and the CICS mirror transaction does not have access to it, CICS returns the NOTAUTH condition to the requesting transaction in the remote system, where the transaction can handle the condition in the usual way.

If the requesting transaction is in a release of CICS that does not support the NOTAUTH condition, CICS raises an alternative condition instead.

EXEC CICS start requests from a remote system

When a transaction is started by an EXEC CICS START request that is issued from a remote system, the request is handled as a function-shipping request. Therefore, the security checks that are applied before the transaction is invoked are run against the RSL keys that are defined for that transaction. These security checks, in addition to the user ID that is established for the request, are done in accordance with the normal inbound security rules, as described in “Link security and user security compared” on page 136.

When the transaction is scheduled, it is handled as a local request; that is, no consideration is given to the security of the link. The TSL keys are checked only against the TSL keys that are defined for the user ID, and future accesses to resources are granted to the transaction as if it were a local transaction. Refer to the *CICS Administration Guide* for details of how TSL and RSL keys are used to control access to local transactions and resources.

It is important to consider the security of remote EXEC CICS START requests when planning intersystem security for your region. In particular, check whether the RSL keys are at least as restrictive as are the TSL keys for your transactions. The only time when this consideration does not apply is when the user ID that is used for an incoming request can always be guaranteed to have no greater security than the security that is associated with the link for that request. For example, this condition applies when a region always uses link security with a link user ID specified.

Using CRTE and CESN to sign on from a remote system

Users from remote systems can sign on to your local system by using the following procedure:

1. Use CRTE.

   Using CRTE, the remote users are logged in as the region’s default user ID. The region’s default user ID is specified with the Region Definitions (RD) **DefaultUserId** attribute.

   The user is given access to transactions and resources as specified with the **TSLKeyList** and **RSLKeyList** attributes that are in the User Definitions (UD) entry for the region’s default user ID and that are also defined for the connection’s link user ID. A link user ID is specified with the Communications Definitions (CD) **LinkUserId** attribute.

   Alternatively, if a link user ID is not specified, the **TSLKeyMask** and **RSLKeyMask** from the CD are used in place of the link user ID’s keys.
2. Use CESN.

   If the user requires access to transactions or resources that are not assigned to the region’s default user ID, CESN can be used to change the user ID.

   When CESN is used, the user assumes access to transactions and resources as specified with the **TSLKeyList** and **RSLKeyList** attributes that are in the User Definitions (UD) entry for the new user ID and that are also defined for the connection’s link user ID.
3. Use CESF or CSSF.

   When the user logs off, by using either CESF or CSSF, the logged-on user ID then becomes the region’s default user ID, and the user is given access to transactions and resources as specified with the **TSLKeyList** and **RSLKeyList** attributes that are in the User Definitions (UD) entry for the region's default user ID and that are also defined for the connection's link user ID (same as item 1).

Intersystem security checklist

Use this checklist to plan and implement intersystem security.

When link security or user security is used

It is good practice to ensure that the transaction RSL keys, which are specified with the Transaction Definitions (TD) **RSLKeys** attribute, are equally or more restrictive than are the TSL keys, which are specified with the TD **TSLKeys** attribute, for any given transaction. Refer to "EXEC CICS start requests from a remote system" on page 142 for an explanation of the importance of restricting the RSL keys for intersystem security.

When link security is used

If you specify **RemoteSysSecurity=local** in the remote system's Communications Definitions (CD) entry, which means that all users from that system have the same level of security, ensure that the link security has adequate restrictions.

- Consider this scenario: SAM on the local system, SYS1, has public access only. It is not intended that SAM be given access to any resource or transaction that is higher than public on SYS1. SAM uses the transaction-routing transaction, CRTE, to log in to SYS2 and becomes the default user ID on SYS2. SAM, as the default user ID on SYS2, can now function ship back to SYS1, which has implemented link security with SYS2 (**RemoteSysSecurity=local**) and can therefore gain access to all transactions and resources that SYS1 allows the users of SYS2 to have.
- To avoid this problem: Do not use link security, where **RemoteSysSecurity=local**, for a connection between your system and a remote system that is not secure enough to allow all users of that system access to your resources.

When user security is used

If you specify **RemoteSysSecurity=trusted** or **verify** in the remote system's CD entry, you can allow a flexibly restrictive combination of keys because they can be restricted, not only by the keys locally defined for the connection's link user ID, but for the user ID flowed from the remote system.

Using the connection's link user ID, rather than the connection's **TSLKeyMask** or **RSLKeyMask** attributes, to specify the link keys for the connection is the advised method. This allows you to associate a distinctive user ID with the link and provides you with the ability to use that user ID for each connection that requires the same link keys. However, consider the following:

Ensure that user IDs in the network do not conflict.

- Consider this scenario: SAM is a secure user on the local system, SYS1, and has access to sensitive transactions. GEORGE from SYS2 requires the same security levels on SYS1 as does SAM. The link user ID is set up to allow GEORGE access to these sensitive transactions, as is GEORGE's user ID on SYS1.

  SAM on SYS2 is not secure and should not be given access to the same transactions as those to which GEORGE has access, but, because SAM is defined on SYS1 as having access to these transactions, SAM from SYS2 assumes the local SAM's security levels when SAM from SYS2 logs in to SYS1, and can, therefore, gain access to the sensitive transactions.

- To work round this problem: Discuss your security requirements with the system administrator of the remote system.

  Note: You and the system administrator on the remote system need to agree about the user IDs that can and cannot be used on either system.

  Be careful when selecting a name for the link user ID.

- Consider this scenario: User security is implemented between SYS1 and SYS2, so it is expected that all remote requests from SYS2 to SYS1 must flow a user ID that is defined on SYS1 or be given public access only. The link user ID for the link between SYS1 and SYS2, RE9X32Y, has enough security levels to allow GEORGE access to parts ordering, and to allow SAM access to parts query. The user ID for SAM on SYS1 is restrictive enough so that SAM cannot order parts.

  The link user ID on SYS1 for the connection between SYS1 and SYS3 is PATTY. This user ID also has enough security levels to allow access to parts ordering. PATTY is also a local CICS user on SYS2, but PATTY on SYS2 is not considered to be a secure user of SYS1. However, because user ID PATTY is defined on SYS1 as having access to parts ordering, PATTY from SYS2 will also be given access to parts ordering on SYS1.

- To work round this problem: Because the link user ID needs to be given enough security to allow access to all transactions and resources that the collective users of the remote system require, it is important that you do not use a trivial or common name for the link user ID. It is also important that you are in agreement with the system administrator of the remote system as to what user IDs can and cannot be used on either system.

Common configuration problems with intersystem security

This section lists symptoms of security problems that are the result of configuration errors. The descriptions suggest possible solutions and point you to further information as required.

Note: Before looking for your symptom in this list, ensure that you are not experiencing any of the symptoms that are described in "Common intercommunication errors" on page 216.

Remote user is logged in to wrong user ID

If a remote user is logged in to a user ID other than the intended user ID, and perhaps given access to the wrong transactions or resources, check the following:

- If the user is using the transaction-routing transaction, CRTE, the flowed user ID is ignored and the user is logged in as the region's default user ID. This is normal operation. The user can use CESN to log on to a locally defined user ID. However, the user security level is still subject to the link keys that are defined for the connection. For more information, see "Using CRTE and CESN to sign on from a remote system" on page 142.
- Is the flowed user ID the same as a locally defined default user ID or link user ID? It is advised that you do not use names for a region's default user ID or a connection's link user ID that are in conflict with other user IDs in the network.
- Was a user ID flowed? If a user ID was not flowed and one was required, the user is logged in as the default user. If you expected a user ID to be flowed and one was not, go to "Flowed user IDs are not received from a remote system" on page 145.

- Was a password flowed? If you are using **RemoteSysSecurity**=**verify**, user IDs that are received by CICS without a password are discarded, and the user is logged on as the default user. If user IDs are received without passwords, set **RemoteSysSecurity**=**trusted**.
- If you expected the link user ID to be used on this request, but it was not, check whether **RemoteSysSecurity=local** and whether a UD entry exists for the link user ID.

User IDs are not flowed to a remote system

If your local region is supposed to be flowing user IDs to a remote system, but is not, check whether:

- The security acceptance field in the remote system is set correctly
- The **OutboundUserIds=sent** is set in the Communications Definitions (CD) entry for the connection to the remote system that is expecting flowed user IDs from your system

Flowed user IDs are not received from a remote system

If a remote system is flowing user IDs to your system, but your system is not receiving them:

- Check whether **RemoteSysSecurity=trusted** or **verify** is set in the CD entry for the connection to the remote system that is supposed to be flowing user IDs to your system. Note that because Microsoft Microsoft SNA Server product does not forward passwords to CICS, always use **RemoteSysSecurity=trusted** with that product.
- If you are using Communications Server for AIX, check whether the security acceptance field that is in your Communications Server for AIX connections definitions is set to "already_verified" or to "conversational".
- If you are using HP-UX SNAplus2, specify conversation-level security.
- If you are using SNAP-IX for Solaris, specify conversation-level security.

CD RSLKeyMask and TSLKeyMask attributes are ignored

This might be the normal operation because the only time that the connection's **RSLKeyMask** and **TSLKeyMask** keys are used is when a link user ID is not specified for the connection. It is advisable to use the link user ID (**LinkUserId**) for link keys.

Unexpected message ERZ045006W - no obvious effect on security

CICS uses the region name as the user ID to run some CICS tasks. If the user ID that is specified in this message corresponds to the name of a remote region, define a UD entry with the region name as the UD key.

Access is unexpectedly restrictive (message ERZ045006W)

This symptom occurs when no User Definitions (UD) entry exists that corresponds to the user ID that is being used for this inbound request. To fix this problem:

1. Determine, from the message, or from the information in "How the resource definition security attributes are used" on page 137, the user ID that is being used.
2. When you know the actual user ID that was used, determine whether this user ID was the flowed user ID (if one was sent), the connection's link user ID, or the region's default user ID.

3. Check whether the correct user ID was used. If not, ensure that you have configured your resources correctly. Use the maps that are given in “How the resource definition security attributes are used” on page 137 to help you determine how the resource definition attributes must be set to ensure that the correct user ID is used.

   If the correct user ID was used, ensure that a UD entry exists for that user ID.

Access unexpectedly rejected or granted on inbound request

Either the user logged in with the wrong user ID, or the keys that are defined for the user ID are not correct. To fix this problem, do the following:

1. Use messages that are associated with the error to determine the actual user ID that was used.
2. When you know the actual user ID that was used, determine whether this user ID was the flowed user ID (if one was sent), the connection’s link user ID, or the region’s default user ID.
3. Based on how your security is set up (or how you want your security to be set up) determine whether the user ID that was actually used was the correct user ID. If it was not the correct user ID, ensure that you have configured your resources correctly. Use the maps that are given in “How the resource definition security attributes are used” on page 137 to help you determine how the resource definition attributes should be set to ensure that the correct user ID is used. Refer also to “Intersystem security checklist” on page 143 for helpful information about administration of user IDs across a network.

   If the correct user ID was used, establish the access that was granted for this user during this particular inbound request. The access granted is a combination of link keys and user keys, or it can be link keys alone, depending on how you have security configured for the connection. For example, the user is given access to transactions and resources based on various combinations of:
   - The keys that are defined for the connection’s link user ID
   - The **TSLKeyMask** or **RSLKeyMask** keys that are defined for the connection
   - The keys that are defined for the region’s default user ID
   - The keys that are defined for the flowed user ID
   - The TSL and RSL key that is assigned to each transaction and resource

If you establish that the problem occurs while attempting to access a transaction or resource, you will need to derive TSL and RSL key lists by using the maps and diagrams that are given in “Link security and user security compared” on page 136. Determine whether the keys that the user is allocated are those keys that are needed to access the required transactions or resources.

When you have established the reason why the access is being wrongly allowed or revoked, you might want to change the privileges of the user ID on this inbound request, or change the access keys on the resources or transactions that are being accessed. Your decision on how to cure this problem must take into account the effect that any change will have on the network.

The discussion has assumed that you are familiar with:
- “Overview of intersystem security” on page 123
- “Authenticating systems across SNA connections” on page 125
- “CICS link security” on page 126
- “CICS user security” on page 127
- “Link security and user security compared” on page 136
- “Intersystem security checklist” on page 143

Chapter 7. Data conversion

Different hardware platforms and operating systems use different standards to represent data. Some use ASCII (American National Standard Code for Information Interchange); some use EBCDIC (Extended Binary-coded Decimal InterChange). Each can use different binary patterns to represent data. Some use a single hexadecimal byte to represent their characters; others might use more. Each has its own way of storing numeric data.

When TXSeries for Multiplatforms communicates with other systems and data flows from one system to another, that data might need to be converted from one format to another. This chapter describes the configuration that is necessary to support this data conversion.

Introduction to data conversion

The encoding of a character set is referred to as a *code page*. A code page defines the meaning of all code points. For example, some code pages define meaning to all 256 code points for an eight-bit code, other code pages define meaning for the 128 code points for a seven-bit code. Data conversion is needed when you have to convert data from one code page to another.

TXSeries for Multiplatforms uses the **iconv** call to convert the encoding of characters from one code page to another. The **iconv** call needs a *conversion table* for each combination of a “from code page” and a “to code page”. For example, if you are converting from IBM-850 to ISO8859-1, you will need an IBM-850 to ISO8859-1 conversion table on your operating system.

CICS on Windows Systems

The table in Appendix D, “Data conversion tables (CICS on Windows Systems and CICS for Solaris only),” on page 347 shows how data can be converted from one code page to another.

CICS on Open Systems

The names of the conversion tables represent the from-to conversions. Although different operating systems can have their own naming conventions, a conversion table source file is generally named as follows:

```
IBM-850_ISO8859-1  on IBM
amere}iso81        on HP 9000
```

This shows that the first part of the name represents the from-code page, and the second part represents the to-code page.

Some operating systems allow you to generate your own conversion tables. To find out more about this, refer to the information about **genxlt** and **iconv** that is provided with your operating system.

SBCS, DBCS, and MBCS data conversion considerations

SBCS, DBCS, and MBCS represent different code page layouts, as described below:

SBCS: single byte character set
This describes an encoding in which each hexadecimal value has a simple relationship with a character. Up to 256 characters can be defined. Not all hexadecimal values necessarily have a meaning. The EBCDIC IBM-037 code page is one example of such a code page; the ASCII IBM-850 is another.

DBCS: double byte character set
This describes an encoding in which some hexadecimal values are recognized as being the first byte of a two-byte sequence that collectively identifies a particular character. This kind of encoding allows far more characters to be defined; in theory up to 65536. The IBM-932 code page (Japanese) is one example of a DBCS code page in which:

```
X'00' to X'7F' are single-byte codes
X'81' to X'9F' are double-byte introducer
X'A1' to X'DF' are single-byte codes
X'E0' to X'FC' are double-byte introducer
```

In this example, code points such as X'80' are undefined and have no meaning. A code point such as X'81' is recognized as being the first byte of a two-byte code. The second byte can be any of the 256 possible values. Other DBCS code pages have different organizations; each one is structured according to need.

MBCS: multibyte character set
This encoding describes a character set in which hexadecimal sequences of arbitrary length are associated with particular characters. eucJP is an example of a multibyte encoding. The principle is the same as that which is described for DBCS, except that particular hex values are recognized as being simple SBCS values or introducers for longer MBCS strings. Sequences of different lengths can be identified within a single code page.

To summarize, MBCS can be viewed as a more generalized form of DBCS, in which sequences of arbitrary length can be defined. SBCS is the simplest of all, in which every sequence is only one byte long.

Note: MBCS is not supported by a standard, although *wide characters* are. A wide character set is one in which the encoding supports multiple bytes (2, 3, or 4), but does so consistently; mixed length sequences do not exist in the set. For instance, IBM’s wide characters are two bytes (DBCS) and Hewlett-Packard’s are four bytes.

In TXSeries for Multiplatforms, no difference exists in coding the CICS-supplied program that performs the standard conversions for SBCS, DBCS, or MBCS data.

With DBCS and MBCS data, you can convert data within the same language (Japanese, Korean, Traditional Chinese, and Simplified Chinese) between the following types of character sets:
- DBCS (ASCII) and DBCS (EBCDIC)
- DBCS (ASCII) and MBCS (EUC)
- DBCS (EBCDIC) and MBCS (EUC)

Cross-language conversion is not possible. Conversions between SBCS and DBCS or MBCS are also not possible.

In the conversions that are listed above, the length of the converted data might differ from the original length because of the insertion and deletion of shift-out (SO) characters and shift-in (SI) characters and the DBCS or MBCS code scheme difference. An SO character is a code extension character that substitutes, for the graphic characters of the standard character set, an alternative set of graphic characters upon which an agreement exists or that has been designated by use of code-extension procedures. An SI character is a code extension character that is used to terminate a sequence that has been introduced by the SO character to make effective the graphic characters of the standard character set.

CICS does not have the logic to handle any special treatment of data-length changes. For example, 20-byte user data in a 20-byte input buffer might become more than 20 bytes long after conversion, in which case the data might be truncated. The application must compensate for expansion during data conversion. For more information about how to use shift-out/shift-in (SO/SI) characters, refer to the *IBM 3270 Information Display Programmer's Reference*.

In transaction routing, 3270 data streams are always flowed across a network in EBCDIC. Routed transaction BMS panels behave in the same way as do ordinary EBCDIC 3270 screens, regardless of the CICS platform or the code page. Therefore, no MBCS-unique considerations are needed for transaction routing.

Numeric data conversion considerations

Data conversion is not only concerned with the representation of character data. The method of storing numbers can vary among different hardware platforms. This means that conversion routines might be required in order to convert numerical (binary) data that is received from a remote system into the local machine format.

As an example, Figure 51 on page 150 shows how some of the standard C language data types are represented on the RS/6000, HP 9000 Series 800 computers, and Intel® computers.

The major difference between the two representations that are shown is the *byte ordering*. The decimal value 550 is stored as 0x0226 on the RS/6000, HP 9000 Series 800 computers, and SPARC machines. These machines use the *big-endian* format for numbers, where the most significant byte of the number is stored in the lower machine address and the least significant byte is stored in the higher machine address. IBM mainframes are also big-endian machines.

On the Intel computers, the bytes are reversed, so decimal 550 is stored as 0x2602. This format is called *little-endian*.

```
Representation of numbers on an IBM RISC System/6000, HP9000 Series 800
Computer, SNI Series, and SPARC machines
C Data type              Value  Size(bytes)   Represented as:
unsigned short int         550       2          02 26
short int                  550       2          02 26
short int                 -550       2          FD DA
unsigned long int       555550       4          00 08 7A 1E
long int                555550       4          00 08 7A 1E
long int               -555550       4          FF F7 85 E2
unsigned int               550       4          00 00 02 26
int                        550       4          00 00 02 26
int                       -550       4          FF FF FD DA
Representation of numbers on  Digital Alpha machines
C Data type              Value  Size(bytes)   Represented as:
unsigned short int         550       2          26 02
short int                  550       2          26 02
short int                 -550       2          DA FD
unsigned long int       555550       8          1E 7A 08 00 00 00 00 00
long int                555550       8          1E 7A 08 00 00 00 00 00
long int               -555550       8          E2 85 F7 FF FF FF FF FF
unsigned int               550       4          26 02 00 00
int                        550       4          26 02 00 00
int                       -550       4          DA FD FF FF
Representation of numbers on Intel machines

C Data type              Value  Size(bytes)    Represented as:

unsigned short int          550          2         26 02
short int                   550          2         26 02
short int                  -550          2         DA FD
unsigned long int        555550          4         1E 7A 08 00
long int                 555550          4         1E 7A 08 00
long int                -555550          4         E2 85 F7 FF
unsigned int                550          4         26 02 00 00
int                         550          4         26 02 00 00
int                        -550          4         DA FD FF FF
```

Figure 51. Representation of numbers

If numerical data is sent between two machines that use different byte ordering, either the sender or receiver must swap the bytes around so that the data is correctly interpreted by the receiving system. In most cases, CICS can be configured to do this byte swapping automatically. However, you might need to include byte-swapping functions in the user exit for function shipping (refer to “Coding a nonstandard data conversion program” on page 161), and transaction routing (refer to “Writing your own version of DFHTRUC” on page 166).

Figure 52 on page 151 and Figure 53 on page 151 show two example C functions that can be used for halfword and fullword byte swapping. The first swaps the bytes for a two-byte number such as a `short int`, and the second swaps the bytes for a four-byte number such as an `int`. Any byte-swapping routine must be aware of the number of bytes that are used to store a number. Therefore, refer to your programming language documentation to check the sizes of the data types that your programs use.

```
/*
*-
* Halfword (2 byte) Byte Swap
*-
*/
 void HalfWord_ByteSwap(void  *Buffer)
 {
    register unsigned char    SavedByte;           /* temp variable */
    unsigned char            *BufferPtr = Buffer;  /* temp pointer */
     /*
      * Save the first byte into the temporary buffer,
      * move the second byte to the first byte position
      * and finally put the saved first byte into the
      * second byte position.
      */
     SavedByte = *BufferPtr;
     *BufferPtr = *(BufferPtr+1);
     *(BufferPtr+1) = SavedByte;
      return;
}
```

Figure 52. Sample C function for halfword swap

```
/*
 *-----------------------------------------------------------------
 * Fullword (4 byte) Byte Swap
 *-----------------------------------------------------------------
 */

void FullWord_ByteSwap(void    *Buffer)
{
    register unsigned char    SavedByte;          /* temp variable */
    unsigned char            *BufferPtr = Buffer; /* temp pointer */

    /*
     * Save the first byte into the temporary buffer,
     * move the fourth byte to the first byte position
     * and finally put the saved first byte into the
     * fourth byte positi  on.
     */

    SavedByte = *BufferPtr;
    *BufferPtr = *(BufferPtr+3);
    *(BufferPtr+3) = SavedByte;

    /*
     * Save the second byte into the temporary buffer,
     * move the third byte to second byte position
     * and finally put the saved second byte into the
     * third byte position.
     */

    SavedByte = *(BufferPtr+1);
    *(BufferPtr+1) = *(BufferPtr+2);
    *(BufferPtr+2) = SavedByte;

    return;
}
```

Figure 53. Sample C function for fullword swap

The machine hardware and operating system can also affect the size of a data type. Variations exist in the size of the `long` data type. It is four bytes on the RS/6000, HP 9000 Series 800 computers, Intel computers, and SPARC machines. If you are sending numerical data between different types of machine, choose data types that are the same size on both machines to simplify the data conversion. The CICS standard data types in *prodDir*/include/cicstype.h provide C language data types that are the same size for all CICS workstation products.

Finally, if the data that your programs send is defined in a record structure, your data conversion routines must handle the padding that can be automatically inserted into the structure to maintain boundary alignment. For example, consider the record structure that is shown in Figure 54, which is defined in the C language. It contains a one-byte character (`OneByteCharacter`), a four-byte integer (`FullWordInteger`), another one-byte character (`AnotherCharacter`) and finally a two-byte integer (`HalfWordInteger`).

```
struct
{
  char       OneByteCharacter;
  int        FullWordInteger;
  char       AnotherCharacter;
  short int  HalfWordInteger;
} ExampleRecord;
```

Figure 54. Example record structure

Although only eight bytes of storage have been defined, the structure actually takes up 12 bytes. Three bytes are inserted after `OneByteCharacter` so `FullWordInteger` starts on a fullword boundary and an additional byte is inserted after `AnotherCharacter`, so `HalfWordInteger` starts on a halfword boundary.

Summary of data conversion for CICS intercommunication functions

Data conversion is required when the code page that is used in a CICS transaction is different from the code page that is used for the resource. The resource can be a file or a temporary storage or transient data queue; or a program, a transaction, or a terminal.

Data conversion has to be considered for:
- Function shipping, distributed program link, and asynchronous processing
- Transaction routing
- Distributed transaction processing

The data conversion requirements for the three intercommunication functions are different. Also, these requests might appear as outbound or inbound to the TXSeries for Multiplatforms region. That is, an application on a TXSeries for Multiplatforms region might initiate a function-shipping request to a remote system (an outbound request), or a remote system might initiate a function-shipping request to a resource that is owned by the TXSeries for Multiplatforms region (an inbound request). Key points are described in the following sections:

Function shipping, distributed program link, and asynchronous processing

1. On the resource-owning system, define a data conversion template for the resource as defined in “Standard data conversion for function shipping, DPL and asynchronous processing” on page 159. Each resource, such as a file or a temporary storage queue, must have a template defined for it if its data is to be converted. These templates indicate how each field in the resource is to be converted.

2. Generate the conversion template by using **cicscvt**.
3. Place the template on the appropriate directory, for example:

   varDir/cics_regions/*region*/database/FD/FILE.cnv
4. Set the **TemplateDefined** attribute in the resource definition entry for the resource to **yes**. This attribute specifies that a template exists for the resource. See the information about Schema File Definitions that is given in the *TXSeries for Multiplatforms Administration Reference* for an example description of the **TemplateDefined** attribute.

Transaction routing:

1. Set the **RemoteCodePageTR** attribute in the Communications Definitions (CD) entry for the remote system to the code page that is to be used to flow transaction routing data.
2. You can customize the transaction routing user exit, DFHTRUC. This user exit converts the COMMAREA and the TCTUA that flow with the transaction routing data. Refer to “Writing your own version of DFHTRUC” on page 166 for further details.

Distributed transaction processing:

- Because DTP programs use application specific data areas, CICS cannot supply data conversion macros or user exits for DTP. The technique that you use is determined by the design of your application.

A full description of the data conversion requirements for the three intercommunication functions is given in:

“Data conversion for function shipping, distributed program link and asynchronous processing” on page 156
“Data conversion for transaction routing” on page 164
“Data conversion for distributed transaction processing (DTP)” on page 169

Code page support

Table 32 shows the different names for code pages on each of the operating systems that TXSeries for Multiplatforms supports. So, for example, the code page name for English (Latin-1) EBCDIC is *IBM-037* on, Windows Systems, and Solaris, and *american_e* on HP. You use these code page names when you are configuring the transaction routing **RemoteCodePageTR** attribute in the Communications Definitions (CD) entry. This task is discussed in “Data conversion for transaction routing” on page 164.

You use the CICS Short Codes that are shown in the table when you are coding DFHCNV macros for function shipping data conversion. See “Data conversion for function shipping, distributed program link and asynchronous processing” on page 156 and “Standard data conversion for function shipping, DPL and asynchronous processing” on page 159 for further information. The short code is the same, no matter which type of CICS region you are using.

A CICS region displays its local code page and the corresponding short code in one of the messages that are written to the **console.***nnnnnn* file during region start-up.

Table 32. CICS Shortcodes and code pages that TXSeries for Multiplatforms supports

CICS short code name	AIX and Windows code page	HP–UX code page name	Solaris code page name	Description
37	IBM-037	american-e	IBM-037	IBMLatin-1 EBCDIC

Table 32. CICS Shortcodes and code pages that TXSeries for Multiplatforms supports (continued)

8859-1	ISO8859-1	iso8859_1	8859	Latin-1 ASCII (ISO)
819	ISO8859-1	iso8859_1	8859	Latin-1 ASCII (IBM/ISO)
850	IBM-850	roman8	IBM-850	Latin-1 ASCII
437	IBM-437	iso8859_1	IBM-437	Latin-1 (PC) ASCII
930	IBM-930	cp930	IBM-930	Japanese EBCDIC
931	IBM-931	japanese_e	IBM-931	Japanese EBCDIC
939	IBM-939	cp939	IBM-939	Japanese EBCDIC
932	IBM-932	sjis	ja_JP.pck	Japanese ASCII
EUCJP	IBM-eucJP	eucJP	eucJP	Japanese ASCII (ISO)
942	IBM-942	IBM-942	IBM-942	Japanese ASCII
EUCKR	IBM-eucKR	eucKR	eucKR	Korean ASCII (ISO)
934	IBM-934	IMB-934	IBM-934	Korean ASCII
944	IBM-944	IBM-944	IBM-944	Korean ASCII
949	IBM-949	korean15	IBM-949	Korean ASCII
933	IBM-933	korean_e	IBM-933	Korean EBCDIC
EUCTW	IBM-eucTW	IBM-eucTW	eucTW	Traditional Chinese
938	IBM-938	IBM-938	IBM-938	Traditional Chinese ASCII
948	IBM-948	IBM-948	IBM-948	Traditional Chinese ASCII
937	IBM-937	chinese-t_e	IBM-937	Traditional Chinese EBCDIC
BIG5	Zh_TW.big5	big5	zh_TW.BIG5	Traditional Chinese BIG5
946	IBM-946	IBM-946	IBM-946	Simplified Chinese ASCII
1381	IBM-1381	hp15CN	IBM-1381	Simplified Chinese ASCII
935	IBM-935	chinese-s_e	IBM-935	Simplified Chinese EBCDIC
EUCN	IBM-eucCN	chinese-s_e	eucCN	Simplified Chinese ASCII (ISO)
GB18030	GB18030	gb18030	GB18030	Simplified Chinese GB18030
864	IBM-864	arabic8	IBM-864	Arabic ASCII
8859-6	ISO8859-6	iso8859_6	ISO8859-6	Arabic ASCII (ISO)
1089	ISO8859-6	iso8859_6	ISO8859-6	Arabic ASCII (IBM/ISO)
420	IBM-420	arabic_e	IBM-420	Arabic EBCDIC
855	IBM-855	IBM-855	IBM-855	Cyrillic ASCII
866	IBM-866	IBM-866	IBM-866	Cyrillic ASCII
8859-5	ISO8859-5	iso8859_5	ISO8859-5	Cyrillic ASCII (ISO)

Table 32. CICS Shortcodes and code pages that TXSeries for Multiplatforms supports (continued)

915	ISO8859-5	iso8859_5	ISO8859-5	Cyrillic ASCII (IBM/ISO)
1025	IBM-1025	IBM-1025	IBM-1025	Multilingual Cyrillic EBCDIC
869	IBM-869	greek8	IBM-869	Greek ASCII
8859-7	ISO8859-7	iso8859_7	ISO8859-7	Greek ASCII (ISO)
813	ISO8859-7	iso8859_7	ISO8859-7	Greek ASCII (IBM/ISO)
875	IBM-875	greek_e	IBM-875	Greek EBCDIC
856	IBM-856	hebrew8	IBM-856	Hebrew ASCII
8859-8	ISO8859-8	iso8859_8	ISO8859-8	Hebrew ASCII (ISO)
916	ISO8859-8	iso8859_8	ISO8859-8	Hebrew ASCII (IBM/ISO)
424	IBM-424	hebrew_e	IBM-424	Hebrew EBCDIC
273	IBM-273	german_e	IBM-273	Austria, Germany EBCDIC
277	IBM-277	danish_e	IBM-277	Denmark, Norway EBCDIC
278	IBM-278	finnish_e	IBM-278	Finland, Sweden EBCDIC
280	IBM-280	italian_e	IBM-280	Italy EBCDIC
284	IBM-284	spanish_e	IBM-284	Spain, Latin Am.(Sp) EBCDIC
285	IBM-285	english_e	IBM-285	UK EBCDIC
297	IBM-297	french_e	IBM-297	France EBCDIC
500	IBM-500	IBM-500	IBM-500	International latin-1 EBCDIC
871	IBM-871	icelandic_e	IBM-871	Iceland EBCDIC
852	IBM-852	IBM-852	IBM-852	Latin-2 ASCII
8859-2	ISO8859-2	iso8859_2	ISO8859-2	Latin-2 ASCII (ISO)
912	ISO8859-2	iso8859_2	ISO8859-2	Latin-2 ASCII (IBM/ISO)
870	IBM-870	IBM-870	IBM-870	Latin-2 EBCDIC
857	IBM-857	turkish8	IBM-857	Turkey ASCII
8859-9	ISO8859-9	iso8859_9	ISO8859-9	Turkey ASCII (ISO)
920	ISO8859-9	iso8859_9	ISO8859-9	Turkey ASCII (IBM/ISO)
1026	IBM-1026	turkish_e	IBM-1026	Turkey EBCDIC
UTF-8	UTF-8 (only)	UTF-8	UTF-8	Unicode file code set
UCS-2	UCS-2 (only)	UCS-2	UCS-2	Unicode processing code set

Data conversion for function shipping, distributed program link and asynchronous processing

The following sections describe the factors that you must considered before you configure data conversion for function shipping:

- "Introduction to data conversion" on page 147

  This introduces the subject of data conversion.
- "Which system does the conversion"

  This explains which of the interconnected systems is responsible for the data conversion.
- "How code page information is exchanged" on page 157

  This explains how the systems exchange information on the code pages that they are using.
- "When TXSeries for Multiplatforms does not convert the data" on page 157

  This explains how data conversion is configured when the TXSeries for Multiplatforms region is not the one that is responsible for the data conversion.
- "Standard data conversion for function shipping, DPL and asynchronous processing" on page 159

  This explains how data conversion is configured when the standard CICS-supplied data conversion programs can be used.
- "Non-standard data conversion (DFHUCNV) for function shipping, DPL and asynchronous processing" on page 160

  This explains how data conversion is configured when the standard CICS-supplied data conversion programs cannot be used, and a user exit has to be created.

Which system does the conversion

Function shipping distributed program link and asynchronous processing involves two categories of data that might need data conversion:

- The name of the CICS resource
- The application data

CICS resource names must always flow across the link in EBCDIC. The system that is not normally using EBCDIC is responsible for the translation, but this translation is an internal CICS function and does not require user setup.

The application data is converted on the system that owns the resource (described in "Standard data conversion for function shipping, DPL and asynchronous processing" on page 159). Figure 55 on page 157 shows this.

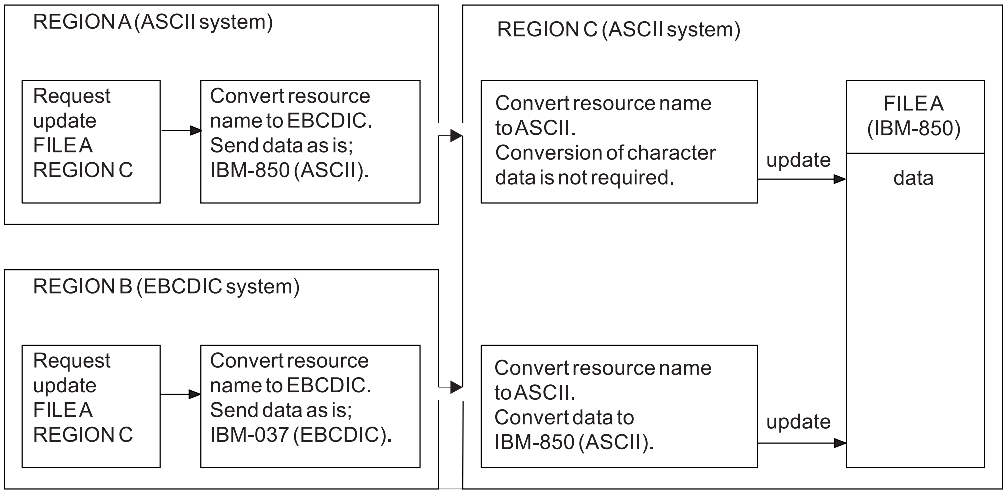

REGIONA(ASCII system)
Request update FILE A REGION C
Convert resource name to EBCDIC. Send data as is; IBM-850 (ASCII).
REGION B (EBCDIC system)
Request update FILE A REGION C
Convert resource name to EBCDIC. Send data as is; IBM-037 (EBCDIC).
REGION C (ASCII system)
Convert resource name to ASCII. Conversion of character data is not required.
update
Convert resource name to ASCII. Convert data to IBM-850 (ASCII).
update
FILE A (IBM-850)
data

Figure 55. Data conversion for function shipping

In Figure 55, REGIONC owns FILE A. FILE A is encoded with code page IBM-850 (ASCII). A transaction on REGIONA requests that FILE A be updated and the data is sent encoded with code page IBM-850. Because the transaction data is encoded in the same code page as is the resource, a conversion of the application data is not required. However, because CICS resource names are always flowed across the connection in EBCDIC (IBM-037), a conversion is required at both ends of the connection.

REGIONB is an EBCDIC system. The transaction data is sent in EBCDIC and converted on REGIONC before FILE A is updated. The resource name is also converted on REGIONC. But, because REGIONB is an EBCDIC system, the resource name is not converted before it is sent across the connection.

How code page information is exchanged

When a request is received from another TXSeries for Multiplatforms region, a *shortcode* is included in the flow that identifies the code page that the transaction is using.

If the transaction on the sending system is using a code page that is not specified in the table, a null value is flowed. Also, non-TXSeries for Multiplatforms regions do not flow code page information. In either case, the default code page that is specified in the conversion template is used. If a shortcode is received that is not in the table, CICS uses the shortcode as the code page.

When TXSeries for Multiplatforms does not convert the data

If your region does not need to convert application data, you do not need to set it up for data conversion. Although you might not be concerned with data conversion on a non-TXSeries for Multiplatforms region, you might need an understanding of what the differences are. Table 33 on page 158 highlights those differences, and lists the other intercommunication manuals that describe data conversion of the other platform.

Table 33. Comparison of data conversion with other CICS systems

Data conversion on:	Differs from data conversion on TXSeries for Multiplatforms in the following ways:	Refer to these books:
IBM mainframe-based CICS	1. Does not send or use code page or byte ordering information flowed over the network. See note for exceptions. 2. CICS does the conversion. Operating system facilities, such as **iconv** in TXSeries for Multiplatforms, are not used.	*CICS Family: API Structure.*
CICS OS/2	Does not send or use code page or byte ordering information flowed over the network.	*CICS for OS/2 Intercommunication Guide*
CICS/400	Does not send or use code page or byte ordering information flowed over the network.	*Communicating from CICS/400*

Note: APAR PN75374 provides code page support for CICS Transaction Server for z/OS 3.3. CICS systems with this APAR recognize code page information in the PIP data on function-shipping flows. For CICS/VSE 2.2 refer to APAR PN75374, and for CICS/MVS 2.1.2 refer to PN61020.

Avoiding data conversion

It is usually assumed that conversion is always required between systems that use different encoding; for example, between EBCDIC and ASCII processors. However, if requests for a particular resource are always in the same code page, you can store that resource on your system in that code page, therefore avoiding the need to convert.Figure 56 shows this.

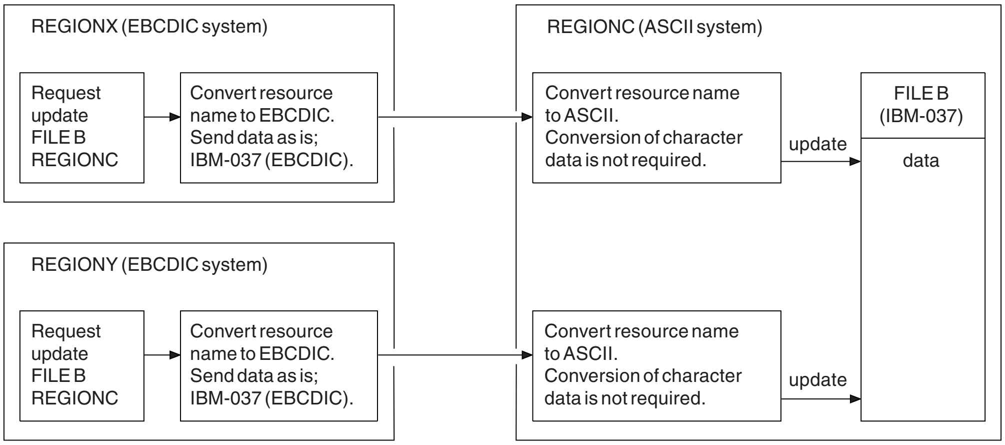

REGIONX (EBCDIC system)
Request update FILE B REGIONC
Convert resource name to EBCDIC. Send data as is; IBM-037 (EBCDIC).
REGIONY (EBCDIC system)
Request update FILE B REGIONC
Convert resource name to EBCDIC. Send data as is; IBM-037 (EBCDIC).
REGIONC (ASCII system)
Convert resource name to ASCII. Conversion of character data is not required.
update
Convert resource name to ASCII. Conversion of character data is not required.
update
FILE B (IBM-037)
data

Figure 56. Avoiding data conversion

In this example, the only systems that need access to FILE B on REGIONC are EBCDIC systems that are using IBM-037. Because of this, FILE B can be encoded in IBM-037 on REGIONC, although REGIONC is an ASCII processor, and data conversion is avoided.

Standard data conversion for function shipping, DPL and asynchronous processing

This section describes standard data conversion in which the CICS-supplied data conversion can be used. "Non-standard data conversion (DFHUCNV) for function shipping, DPL and asynchronous processing" on page 160 describes the non-standard conditions in which the CICS-supplied data conversion cannot be used and user exits must be used.

When a request to access a resource is function shipped, the resource definition for that resource is examined. If **TemplateDefined=yes**, CICS looks for a *conversion template* that is defined for the resource to determine whether conversion is required and how conversion is to occur.

Each resource that requires conversion must have a conversion template defined for it. These conversion templates specify:

- The code page that is used to encode the resource character data when it is stored on the resource owning system
- The default code page that is to be assumed for the incoming request if the incoming request does not state how the data is encoded
- Field lengths
- What parts of the field contain character data, binary data, MBCS (graphic), or packed decimal data

Conversion templates are user-defined. To define a conversion template:

1. You code a *macro source conversion table*, using the DFHCNV macros, to specify how you want the resources converted. One macro source conversion table can contain specifications for several resources. (See Appendix A, "DFHCNV - The data conversion macros," on page 321.)
2. You process your macro source conversion table, using a CICS-supplied command, **cicscvt**, to create the conversion templates. One conversion template is built for each resource. (See "Using cicscvt to build the conversion templates.")

Note: DFHCNV macros are generally portable between CICS systems. Some minor changes might be required. See "When TXSeries for Multiplatforms does not convert the data" on page 157 for information about the differences between CICS systems.

Using cicscvt to build the conversion templates

The **cicscvt** program is a standalone utility that converts a macro source conversion table to individual compiled conversion templates for each resource that is defined in the table. To produce the conversion templates, enter:

```
cicscvt fileName
```

where *fileName* is the name of the coded DFHCNV conversion table.

cicscvt names the conversion templates based on the resource type and resource name that are specified with the DFHCNV macro. The source macro format uses abbreviations for the resource classes. Those abbreviations are shown in Table 34 on page 160:

Table 34. Resource class abbreviations and resource types

RTYPE	resource_class
FC	File Definitions (FD)
TD	Transient Data Definitions (TDD)
TS	Temporary Storage Definitions (TSD)
IC	Transaction Definitions (TD)
PC	Program Definitions (PD)

The output of **cicscvt** is:

```
resource_name.resource_type.cnv
```

For example, **cicscvt** produces a file named "ABCD.TSD.cnv" when the following is coded:

```
        DFHCNV    TYPE=ENTRY,RTYPE=TS,RNAME=ABCD
```

To use this conversion template, you need to install it as either:

```
\var\cics_regions\regionName\database\TSD\ABCD.cnv
/var/cics_regions/regionName/database/TSD/ABCD.cnv
```

Notice that the file needed to be renamed to "ABCD.cnv".

Updating a conversion template in the runtime environment: If you want to update a conversion template in the runtime environment you should:
1. Install the conversion template in the appropriate directory.
2. Delete the resource and reinstall it.

This forces CICS to reload the conversion template the next time that the resource is accessed.

For TS Queues, you should also execute an EXEC CICS DELETEQ TS of the queue, before carrying out the above procedure.

Non-standard data conversion (DFHUCNV) for function shipping, DPL and asynchronous processing

"Standard data conversion for function shipping, DPL and asynchronous processing" on page 159 describes how the CICS-supplied data conversion program uses the conversion templates to perform standard data conversions. When a field requires nonstandard conversion, you can specify, at the resource level, that a user exit is to be used (USREXIT=YES in the TYPE=ENTRY macro).

For example, a user exit can be used for data conversion when:

- You need to convert uppercase EBCDIC to lowercase ASCII
- The conversion logic is dependent on the data itself
- You have some fields that require standard conversion, and some that require nonstandard conversion
- You want to provide your own complete conversion mechanism from with DFHUCNV

A default function-shipping user conversion program, DFHUCNV, is supplied for nonstandard conversions. This program is provided to showe how to interface with CICS to perform nonstandard data conversions when function shipping. In its unmodified form, it converts all uppercase characters to lowercase if the user conversion type is a particular hexadecimal value.

You can implement DFHUCNV in its unmodified form, or you can add user-dependent processing to meet your particular requirements. However, before you change the default user-conversion program, try it first; if you do decide to change the functions of DFHUCNV, take a copy of the program source so that you have a backup in case you encounter problems and need to revert to the original version.

You can write the program in C or COBOL, and you can use EXEC CICS commands.

You must define the program in the Program Definitions (PD) entry DFHUCNV.

Performance and data conversion

CICS could call the user-conversion program every time a function shipping request is processed. Therefore, the performance of the user program is critical to the performance of the function-shipping mirror transaction.

When the user-conversion program has returned, CICS performs standard conversions. CICS examines the conversion template to obtain the first SELECT item in the linked list. If the user data matches the comparison data in the SELECT item, CICS traverses the FIELD linked list, and converts the data as described by the FIELD entry. CICS repeats this process until all matching user data is converted.

Performance might be degraded if your conversion templates are poorly organized, and result in extended searches for matching fields. You should ensure that the most frequently matched fields are defined early in the conversion template, therefore shortening the search times.

Coding a nonstandard data conversion program: CICS uses the COMMAREA to pass a parameter list to the user-replaceable conversion program that is providing access to the data conversion table and the data that is to be converted. All pointers point to an area in the COMMAREA.

The following lists and describes the parameters. The names shown are those that are used in the CICS-supplied sample header. A C language header file for these parameters is in:

```
prodDir/include/cics_fscv.h
```

The C language source for the supplied version of DFHUCNV is in:

```
prodDir/src/samples/ucnv/cics_fscv.ccs
```

Signature
: This is the eight-character signature (ERZ27CVU) for the parameters that CICS passes to the user-conversion program.

ConvDir
: CICS passes this 32-bit parameter to the user conversion program. The parameter indicates in which direction conversion should be performed, and has one of these two values:
  0 = Conversion from local to remote
  1 = Conversion from remote to local

ResourceName
: This is the name of the resource, and is **NULL** if CICS has not initialized the conversion template.

ResourceNameLength
This is an unsigned 16-bit integer that specifies the length of the resource named. The length is zero if CICS does not initialize the conversion template.

ResourceT
This 32-bit parameter defines the type of resource that the user data conversion function can use as a key for obtaining a conversion template. It has one of these values:

0 = file control
1 = transient data
2 = temporary storage
3 = interval control
4 = NOCHECK interval control
5 = distributed program link (DPL)

UserData
This 32-bit parameter points to the user data for conversion. It might be **NULL** if no data exists for conversion.

UserDataLength
This is an unsigned 16-bit integer that specifies the length of the user data. The length is zero if no user data exists.

KeyData
This 32-bit parameter points to file control key data for conversion. It might be **NULL** if no key data exists for conversion.

KeyDataLength
This is an unsigned 16-bit integer that specifies the length of key data. The length is zero if no key data exists.

LocalToRemote
This is a code page descriptor for conversion from the local code page to the remote code page.

RemoteToLocal
This is a code page descriptor for conversion from the remote code page to the local code page.

ByteOrder
The byte ordering of the remote system. (See "Numeric data conversion considerations" on page 149 for further details about byte ordering.)

0 = Byte order for IBM mainframes, , HP-UX, and Solaris

(Called Network order, or Big Endian order

1 = Byte order for OS/2 and Windows Systems

(Called Little Endian)

ConvTable
This is a pointer to the root of a linked list that details selected data from the conversion template for the resource. Each element in the list contains the root of a linked list of field data for the conversion template. All elements in the linked lists are for input only and should not be modified. The linked list of data for a conversion template consists of:

Signature
This is the eight-character signature ("ERZ27CVS") for the structure.

Next This is a pointer to the next select element in the linked list. It is set to **NULL** if it is at the end of the list.

SelectType
This 32-bit parameter specifies the type of select. Refer to the following values:

0 = Default. OPTION=DEFAULT has been specified.

1 = Data. Indicates that the data to compare is in character format.

2 = Hex Data. Indicates that the data is to be compared without conversion.

3 = Key. Indicates the start of conversions that are to be applied to file control keys.

CompareOffset
This is an unsigned 16-bit integer that specifies the offset, from the start of the data area, at which the comparison is to begin.

CompareLength
This is an unsigned 16-bit integer that specifies the length of the comparison data.

CompareData
This is a pointer to the data that is to be compared against.

FieldRoot
This is a pointer to the root of a linked list of field data for a conversion template. It describes what to convert if a match was found and is part of a linked list such that several conversions can be made. This field data consists of:

Signature
This is the eight-character signature ("ERZ27CVF") for the structure.

Next This is a pointer to the next conversion template element. It contains **NULL** if it is at the end of the list.

ConversionOffset
This is an unsigned 16-bit integer that specifies the offset, from the start of the data, at which the conversion is to begin.

ConvertType
This 32-bit parameter specifies the type of data to convert, and has one of these seven values:

0 = Data is character. Convert as a character, using the **iconv** library.

1 = Data is graphic (DBCS string without SOSI characters).

2 = No conversion required.

3 = User data.

4 = Data is 16-bit numeric.

5 = Data is 32-bit numeric.

ConvertLength
This is an unsigned 16-bit integer that specifies the length of the conversion data.

UserEscapeType
This is the user escape type unsigned 16-bit integer that is passed to DFHUCNV.

Data conversion for transaction routing

The data that flows between two CICS systems when a CICS transaction is using a remote terminal consists of:

- The data that is displayed on the screen. This is sent as 3270 data streams.
- Any COMMAREA and TCTUA that is saved by a pseudo-conversational transaction when it returns.

Data conversion for transaction routing is required when the terminal-owning region (TOR) and application-owning region (AOR) use a different code page or byte ordering.

CICS on Open Systems and CICS on Windows Systems can be configured to convert automatically the screen data (3270 data streams) that is sent to, and received from, a remote CICS system. Data conversion is triggered by the **RemoteCodePageTR** attribute of the Communications Definition (CD) entry for a connection. If this attribute is set to a code page that is different from the local code page, TXSeries for Multiplatforms ensures that the screen data that is sent to the remote system is converted to the code page that is specified in **RemoteCodePageTR**. TXSeries for Multiplatforms also assumes that screen data that is received from the remote system is in the code page that is specified in **RemoteCodePageTR** and will convert it to the local code page. So the code page that is specified in **RemoteCodePageTR** can be viewed as the code page of all transaction-routed screen data that is flowing between the two CICS systems.

Choosing an appropriate value for **RemoteCodePageTR** depends upon the particular remote CICS system.

- IBM mainframe-based CICS does not perform any data conversion for transaction routing. Therefore, if your TXSeries for Multiplatforms region is communicating with an IBM mainframe-based CICS system, set **RemoteCodePageTR** to the EBCDIC code page that is used in the IBM mainframe-based CICS system. Your TXSeries for Multiplatforms region does all the data conversion and the screen data flows in the code page of the IBM mainframe-based CICS system.
- CICS OS/2 can perform data conversion for transaction routing. However, it assumes that all screen data is flowed in EBCDIC. If your TXSeries for Multiplatforms region is communicating with a CICS OS/2 region, set **RemoteCodePageTR** to the same EBCDIC code page as that which is specified in the CICS OS/2 **Partner Code Page** attribute of the Connection and Session Table (TCS) entry for the connection.
- TXSeries for Multiplatforms can receive screen data in both ASCII and EBCDIC. If your TXSeries for Multiplatforms region is communicating with another TXSeries for Multiplatforms region, set the **RemoteCodePageTR** to the same value ***in both regions***. For performance reasons it is sensible to make the code page that is specified in **RemoteCodePageTR** the same as at least one of the region’s local code pages, to reduce the amount of data conversion. This is not essential, however, and you can choose to use a code page between which both regions can convert.

RemoteCodePageTR is expressed by use of the code page names that are defined in your local machine. Table 32 on page 153 shows the code page names for your operating system. In addition, ensure that an **iconv** data conversion table that can convert both ways between the code page that is specified in **RemoteCodePageTR** and the local code page, is installed on your machine. The default local code page for the region is displayed in one of the messages that is output in the console.*nnnnnn* file during region startup.

TXSeries for Multiplatforms can convert transaction routing screen data automatically because the structure of the 3270 data streams is well understood. TXSeries for Multiplatforms cannot automatically convert the data in a TCTUA or a COMMAREA that is flowed during transaction routing because its contents are application defined. It might contain a mixture of character and binary data that must be converted by different techniques. TXSeries for Multiplatforms therefore provides a user exit called **DFHTRUC**, which can be customized to your region's applications. The default version of DFHTRUC that is supplied with your TXSeries for Multiplatforms region does nothing. You might have to change it if:

- Transaction routing is being used between your region and a remote CICS system that uses a different code page ***and***
- Pseudo-conversational transactions are being routed that save a TCTUA or COMMAREA when they return ***and***
- Some of the transactions that are accessing this TCTUA and/or COMMAREA run on one system, while others run on the other

The information that follows discusses how to write your own version of DFHTRUC.

When is DFHTRUC called

DFHTRUC is called under the following conditions:

- The code page that is in **RemoteCodePageTR** is different from the local code page ***and***
- a TCTUA and/or COMMAREA is included in the transaction routing data

It can be called in the following places:

- When you region is the terminal-owning region (TOR) and a TCTUA and/or COMMAREA is being sent to the application-owning region at the beginning of a routed transaction
- When your region is the application-owning region (AOR) and a TCTUA and/or COMMAREA has been received with a routed transaction request
- When your region is the application-owning region (AOR) and a TCTUA and/or COMMAREA is to be sent with the last data for the routed transaction
- When you region is the terminal-owning region (TOR) and a TCTUA and/or COMMAREA has been received with the last data for the routed transaction

Only one DFHTRUC is in your region. It must therefore be coded to handle all transactions that require conversion of the TCTUA or COMMAREA. "Parameters passed to DFHTRUC" describes the information that is passed to DFHTRUC and that allows it to determine the conversion required.

Parameters passed to DFHTRUC

DFHTRUC is passed a COMMAREA that contains the following information:

SysIdLength
This is an unsigned 16-bit integer that defines the length of the SYSID that is stored in the SysId field below. The value must be 1 through 4.

SysId This is a five-character field that contains the SYSID of the remote system, padded on the right will NULLs (0x00). The SYSID is the name of the Communications Definition (CD) entry for the remote system.

COMMAREA
This is a pointer to the routed transaction's COMMAREA. If the routed transaction did not have a COMMAREA, this field is NULLs (0x00).

COMMAREALength
This is an unsigned 16-bit integer that defines the length of the COMMAREA.

TCTUA
This is a pointer to the routed transaction's TCTUA. If the routed transaction did not have a TCTUA, this field is NULLs (0x00).

TCTUALength
This is an unsigned 16-bit integer that defines the length of the TCTUA.

Direction
This parameter indicates whether the data is being sent to (outbound), or received from (inbound), the remote system. It has the following values:

0	Outbound
1	Inbound

If the data is outbound, DFHTRUC needs to convert from the local code page to the remote code page. If the data is inbound, DFHTRUC needs to convert from the remote code page to the local code page.

LocalCodePageLength
This unsigned 16-bit integer contains the length of the local code page that is stored in the LocalCodePage parameter.

LocalCodePage
This is a 255-byte character array that contains the local code page that is extracted from the operating system.

RemoteCodePageLength
This unsigned 16-bit integer contains the length of the remote code page that is stored in the RemoteCodePage parameter.

RemoteCodePage
This is a 255-byte character array that contains the code page from the **RemoteCodePageTR** attribute of the connection's CD entry.

A C language definition of these parameters is in file *prodDir*/include/cics_truc.h.

DFHTRUC can look at the EXEC Interface Block (EIB) to determine the transaction that DFHTRUC is converting for (EIBTRNID). Finally, DFHTRUC can use EXEC CICS commands such as ASSIGN to determine additional information about the transaction.

Writing your own version of DFHTRUC

The source for the supplied version of DFHTRUC (cics_truc.ccs) along with a Makefile to build it are in directory:

```
prodDir/samples/truc
```

In addition, it uses the header file called `prodDir/include/cics_truc.h`, which defines the parameters that are passed to DFHTRUC. cics_truc.ccs is written in the C programming language and includes not only the supplied version of

DFHTRUC (see the function called main), but also two example functions that might help you write your own DFHTRUC. The function called Convert uses the **iconv** utility to convert a buffer of character data from one code page to another. The function called SampleDFHTRUC is a sample version of DFHTRUC that converts the COMMAREA and TCTUA for all transactions that do not begin with 'c' (or 'C'). The conversion logic assumes that the COMMAREA and TCTUA contain only character data, and that the process of conversion does not change the number of bytes in the COMMAREA/TCTUA.

Although SampleDFHTRUC is a simple version of DFHTRUC, it does show several important features of a DFHTRUC program. These are described below.

The first part of a DFHTRUC program should determine whether it understands the format of the COMMAREA or TCTUA. It can do this by looking at EIBTRNID to determine whether the transaction is one for which DFHTRUC has been coded to convert. If it is no, DFHTRUC should return immediately. DFHTRUC must never convert the COMMAREA/TCTUA of a CICS-supplied transaction because the format of these data areas is not published. So SampleDFHTRUC looks for a 'C' at the beginning of EIBTRNID and returns if it occurs:

```
EXEC CICS ADDRESS EIB(EIB);
if (((EIB->eibtrnid)[0] == 'C') || ((EIB->eibtrnid)[0] == 'c'))
{
    EXEC CICS RETURN;
}
```

A safer test would be to test explicitly for the transactions for which DFHTRUC is coded to convert. The code fragment below returns if EIBTRNID is ***not*** PT01, PT02 or PT05.

```
EXEC CICS ADDRESS EIB(EIB);
if ((memcmp(EIB->eibtrnid, "PT01", 4) != 0) &&;amp;
    (memcmp(EIB->eibtrnid, "PT02", 4) != 0) &&;amp;
    (memcmp(EIB->eibtrnid, "PT05", 4) != 0))
{
    EXEC CICS RETURN;
}
```

It can also be useful to test EIBTRMID becauses the terminal name might, for example, identify the type of TCTUA that the application uses.

When DFHTRUC has identified the type of COMMAREA/TCTUA that it is converting, it should examine the Direction parameter that is passed to it. This indicates whether data is being sent to, or received from, the remote system. Its significance is primarily to determine whether the data conversion is from the local code page to the remote code page, or from the remote code page to the local code page. In the code fragment below, the Direction parameter is used to set two local variables, FromCodePage and ToCodePage that are used later in DFHTRUC when calling **iconv**.

```
EXEC CICS ADDRESS COMMAREA((cics_char_t **)&Param);
if (Param->Direction == CICS_DFHTRUC_DIR_OUTBOUND)
{
    FromCodePage = Param->LocalCodePage;
    ToCodePage = Param->RemoteCodePage
}
else
{
    FromCodePage = Param->RemoteCodePage;
    ToCodePage = Param->LocalCodePage;
}
```

The final stage of DFHTRUC is to do the conversion. Before you attempt to convert either the TCTUA or the COMMAREA, check whether one exists. (For example, an application might be using a TCTUA and not a COMMAREA. DFHTRUC will be called to convert the TCTUA only.) The code fragment below tests that the transaction had a COMMAREA before calling the conversion routine.

```
EXEC CICS ADDRESS COMMAREA((cics_char_t **)&Param);

if (Param->COMMAREALength != 0)
{
    RetCode = ConvertPT01CommArea(Param->COMMAREA,
                                  FromCodePage,
                                  ToCodePage);
}
```

Character data can be converted by use of **iconv**. Converting binary data (numbers) depends on the machine architecture of both the local and the remote system. No conversion might be necessary. However, some machine architectures use a different byte ordering for storing numbers and DFHTRUC might need to swap the bytes around. Use the `SysId` parameter to identify the remote system and convert according to the remote system type.

When the conversion is finished, DFHTRUC should return to CICS by using `EXEC CICS RETURN;`.

Note: If you want to use the functions in the supplied cics_truc.ccs file, copy cics_truc.ccs to your own directory before modifying it so that you can refer to the original version if necessary. If you want to use the sample Makefile, copy that to you own directory also, and modify the `PROGRAM` variable to the path name to which your complied DFHTRUC should be written. The value of `PROGRAM` in the supplied Makefile overwritesthe supplied DFHTRUC in *prodDir*/bin.

Installing your version of DFHTRUC into the region

The location of DFHTRUC is specified in the DFHTRUC Program Definition (PD) entry. The **PathName** attribute of this entry is set to **DFHTRUC**,which means CICS usesthe first version of the file DFHTRUC that it finds in the directories that are specified in the region's PATH. To install your own version of DFHTRUC either:

- Copy your version as DFHTRUC to a directory that is specified before *prodDir*/bin in your region's PATH. For example, `/var/cics_regions/`*`regionname`*`/bin`. This means that CICS will find your version of DFHTRUC before the supplied version in *prodDir*/bin.

  Alternatively, change the PD entry for DFHTRUC so that it specifies the location of your version of DFHTRUC in the PathName attribute. This PD entry is protected (Permanent=yes) so you will have to switch it to unprotected (Permanent=no) while you update it. The example below shows the PD entry for

DFHTRUC being updated in both the permanent and running database of CICS region cicsopen. The existing version is deleted from the running region. Then, the new value of **PathName** is set along with **Permanent=no**. Finally the process is repeated to switch the PD entry back to **Permanent=yes**.

```
cicsdelete -c pd -r cicsopen -R DFHTRUC
cicsupdate -c pd -r cicsopen -B DFHTRUC PathName=/cicsbin/DFHTRUC \
Permanent=no
cicsdelete -c pd -r cicsopen -R DFHTRUC
cicsupdate -c pd -r cicsopen -B DFHTRUC Permanent=yes
```

When the PD entry points to the location of your version of DFHTRUC, either:
- Restart your region or
- Use CEMT SET PROGRAM(DFHTRUC) NEW

to bring the new DFHTRUC into use.

Data conversion for distributed transaction processing (DTP)

DTP programs use application-specific data areas and conversion flows. Because of this, CICS cannot provide a general procedure for data conversion for DTP. It is, therefore, the application's responsibility to perform data conversion.

Data conversion for DTP programs can be coded in several ways. Each has its advantages and disadvantages, so your choice of technique is determined by the design of your particular DTP program. Your goal should be to minimize data conversion and, where it must occur, arrange for it to be done by the system that has the most spare capacity.

If a DTP program will be communicating with partner programs that are, for example, sometimes on an ASCII system and sometimes on an EBCDIC system, you could code your DTP programs to flow data always in a common code page, such as an EBCDIC code page. Then, each program will always know the code page of the data that it is receiving.

Another strategy is for each program to send details of its code page in the first flow of data, or use PIP data. The partner can convert, to the correct code page, the data that it is sending.

Finally, it might be possible for all the DTP programs to work in one code page, therefore eliminating data conversion completely.

Summary of data conversion

The systems to which your TXSeries for Multiplatforms region can connect can store data in different character encodings. For example, TXSeries for Multiplatforms uses ASCII, and CICS Transaction Server for z/OS uses EBCDIC. You therefore need to use data conversion for integer and text when using function shipping, transaction routing and distributed transaction processing.

Data conversion for function shipping

The CICS products define that conversion always takes place in the resource-owning system. That is, in the system that owns the file, queue, program or transaction that is being accessed by function shipping.

To perform data conversion for function shipping in TXSeries for Multiplatforms:

1. Define a conversion template in the form of a file that contains DFHCNV macros, and process the conversion template source file by using the TXSeries for Multiplatforms **cicscvt** program. ("Standard data conversion for function shipping, DPL and asynchronous processing" on page 159 describes how the conversion template source files are coded and converted.)
2. Install the processed files into the appropriate TXSeries for Multiplatforms class subdirectory for your region.
3. Set the **TemplateDefined** attribute to yes in the definition for the resource that is being converted. This tells TXSeries for Multiplatforms that conversion of the data is required, and that a conversion template should be loaded.

The conversion templates describe the format of the data that is to be converted, and the code page and byte order between which the data is to be converted. Because only one conversion template can be defined per resource, all systems that are accessing that resource must have the same conversion performed on it. That is, if a TXSeries for Multiplatforms region owns a file named ACCOUNTS, and a conversion template exists for it that defines that all data in the file is to be converted from US ASCII to US EBCDIC, that conversion is performed regardless of what system is accessing the file. This means that if both a TXSeries for Multiplatforms region and a CICS Transaction Server for z/OS system function ship a read of that file, they will both receive the data in US EBCDIC. This is fine for the CICS Transaction Server for z/OS region, but the CICS on Open Systems or CICS on Windows Systems region will not get the expected data.

To avoid this problem, some CICS products (including TXSeries for Multiplatforms) flow their code page and byte order in addition to the normal function shipping information. This allows the resource owning system to determine in what environment the remote (requesting) system is executing, and to override the code page and byte order that is specified in the conversion template. This allows two or more disparate systems to access the same data, and get the appropriate results. If the requesting system does not support this, the default action is always to convert the data as it is defined by the conversion template.

Note: Binary queue names should not be used for remote queues, even when the queues are using the same operating system as the local host. In addition, IBM mainframe-based CICS does not permit the use of binary queue names.

Data conversion for transaction routing

Transaction routing from CICS Transaction Server for z/OS to TXSeries for Multiplatforms presents similar problems to those of function shipping. That is, a panel that is displayed by TXSeries for Multiplatforms on a terminal that is attached to CICS Transaction Server for z/OS would not be usable, because TXSeries for Multiplatforms would have flowed its data in ASCII, and the CICS Transaction Server for z/OS terminal would try to display it in EBCDIC.

To perform data conversion for transaction routing, define, in the **RemoteCodePageTR** attribute of the CD entry for the remote system, the code page that you want the transaction data to use. This causes TXSeries for Multiplatforms to convert terminal data into that code page.

In addition, a COMMAREA or TCTUA that flows when the transaction routed transaction starts or finishes can be converted by the Transaction Routing User Conversion Program (DFHTRUC). This is described in "Data conversion for transaction routing" on page 164.

Data conversion for distributed transaction processing

DTP programs use application-specific data areas and conversion flows. Because of this, CICS cannot provide a general procedure for data conversion for DTP. It is, therefore, the application’s responsibility to perform data conversion.

This is discussed in “Data conversion for distributed transaction processing (DTP)” on page 169.

Part 3. Operating an SNA intercommunication environment

This part describes how to manage your CICS region, PPC Gateway server, and SNA product, and reviews the problem determination procedures for these products.

Table 35. Road map

If you want to...	Refer to...
Read about managing a PPC Gateway server.	Chapter 8, “Creating and using a PPC Gateway server in a CICS environment,” on page 175
Read about the procedures that you should follow when investigating CICS intersystem communication problems	“Intersystem problem solving process” on page 213
Read about PPC Gateway server problem determination	“PPC Gateway server problem determination” on page 228

Chapter 8. Creating and using a PPC Gateway server in a CICS environment

A TXSeries for Multiplatforms region can use a Peer-to-Peer Communications (PPC) Gateway server to communicate with remote systems across a Systems Network Architecture (SNA) network with synchronization level 2 (SL2) support. (For descriptions of synchronization levels, see "Ensuring data integrity with synchronization support" on page 16.) This chapter discusses the basics of creating and using a PPC Gateway server with CICS. The following topics are covered:

- "Overview of the PPC Gateway server"
- "Choosing a management method" on page 181
- "Using the CICS control program (cicscp) configuration tool to manage a PPC Gateway server" on page 182
- "Using CICS commands to manage a PPC Gateway server" on page 184
- "Using the IBM TXSeries Administration Tool to manage a PPC Gateway server" on page 190
- "Using SMIT to manage a PPC Gateway server" on page 195
- "Using ppcadmin commands" on page 200
- "Changing CICS intercommunication definitions for use with a PPC Gateway server" on page 210

Overview of the PPC Gateway server

A PPC Gateway server runs independently of a CICS region. A CICS region is connected to the PPC Gateway server through TCP/IP. The PPC Gateway server provides a link to the SNA network.

To use PPC Gateway server SNA support, you must install and configure an appropriate SNA product on the same machine as is the PPC Gateway server. The appropriate communications product depends upon the platform:

Table 36. Communications products for various platforms

Platform	Communications product
Windows	IBM Communications Server or Microsoft SNA Server
AIX	IBM Communications Server
Solaris	SNAP-IX
HP-UX	SNAplus2

To use CICS commands to create or change a PPC Gateway server, CICS must exist on the same machine as does the server. To use CICS commands only to communicate with a PPC Gateway server, CICS does not have to exist on the same machine.

Note: CICS does not support the use of a PPC Gateway server on Solaris. However, a CICS for Solaris region can use a PPC Gateway server that is running on a non-Solaris platform.

A CICS region can use more than one PPC Gateway server. This type of configuration provides the following performance benefits:

- If one PPC Gateway server fails, another is available as a backup.
- The processing load is spread across more than one PPC Gateway server and across multiple SNA products.

A single PPC Gateway server can also be shared by more than one CICS region. This can be an economical alternative if your CICS regions do not make many SNA intercommunication requests. However, such a configuration can overload a PPC Gateway server.

Each PPC Gateway server requires an operating system user ID and logical volume, which you must create before creating the PPC Gateway server. The logical volume stores the server's log file, which contains its recoverable information. As a result, the user ID for the PPC Gateway server must have read and write permission to this logical volume.

CICS uses a Gateway Server Definitions (GSD) entry to specify the characteristics that are required to start and stop the PPC Gateway server. This GSD entry is created automatically when the PPC Gateway server is created, and deleted when the server is destroyed.

The PPC Gateway server has a directory to store its working files, which is created automatically when the PPC Gateway server is created. It is of the form:

```
/var/cics_servers/GSD/cics/ppc/gateway/server_name
```

on Open Systems and

```
C:\var\cics_servers\GSD\cics\ppc\gateway\server_name
```

on Windows systems.

In both cases, *server_name* is the one- to eight-character unique identifying name of the particular PPC Gateway server.

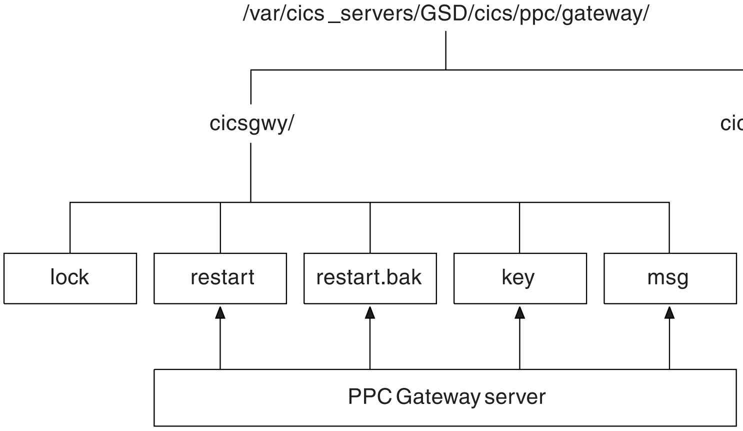


Figure 57. Directory structure of the PPC Gateway server (assumes an Open Systems platform)

Within the directory **cicsgwy** are working files that the PPC Gateway server uses, such as:

lock This file is used by the **cicsppcgwylock** command to indicate whether the

PPC Gateway server is running. (The **cicsppcgwylock** command is used by the **cicsppcgwycreate**, **cicsppcgwy**, **cicsppcgwyshut**, and **cicsppcgwydestroy** commands, and is not required under normal conditions.)

restart and restart.bak
These files tell the PPC Gateway server during an autostart where its logical volume is and how it is formatted. They are created by the PPC Gateway server and should be read and changed only by the PPC Gateway server.

msg The PPC Gateway server writes its messages to this file. These messages are described in "PPC Gateway server problem determination" on page 228.

Sharing server names between a CICS region and a PPC Gateway server

A PPC Gateway server and a CICS region communicate by using a mixture of Remote Procedure Calls (RPCs) and native TCP/IP. For a PPC Gateway server and a CICS region to communicate, they must know one another's server name. Figure 58 shows how the server names are stored and passed between them.

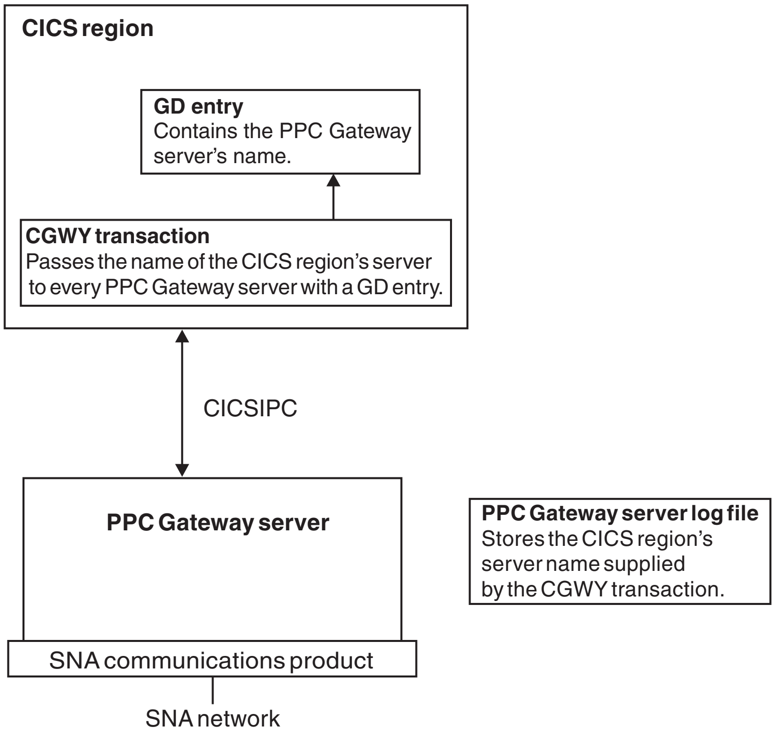

CICS region
GD entry
Contains the PPC Gateway server's name.
CGWY transaction
Passes the name of the CICS region's server to every PPC Gateway server with a GD entry.
CICSIPC
PPC Gateway server
PPC Gateway server log file
Stores the CICS region's server name supplied by the CGWY transaction.
SNA communications product
SNA network

Figure 58. How server names are shared between a CICS region and a PPC Gateway server

The server name of the PPC Gateway server is configured in the CICS region by using a Gateway Definitions (GD) entry. (See "Configuring CICS for PPC Gateway server SNA support" on page 89 for information about how to create a GD entry.) The CICS region has a CICS-supplied private transaction called CGWY that passes the server name of the CICS region (and other relevant information) to each of the

PPC Gateway servers that are defined in the GD entries. This transaction is run when the region starts up. You can also run the CGWY transaction while the CICS region is running, by using an **EXEC CICS START TRANSID(CGWY)** command.

When the PPC Gateway server receives the information from the CGWY transaction, it stores it in its log file. This information is unaffected by stopping and restarting the PPC Gateway server.

As the CGWY transaction runs, it writes messages to the console log for the region. Here are some example messages:

```
ERZ030060I/3101 time date region : PPC Gateway server configuration
                has started
ERZ030062I/3103 time date region : Configuring PPC Gateway server 'GWY'
                with details of the region
ERZ030063I/3104 time date region : Configuring PPC Gateway server 'GWY'
                with details of local transactions
ERZ030064I/3109 time date region : Configuring PPC Gateway server 'GWY'
                has ended
ERZ030061I/3102 time date region : PPC Gateway server configuration
                has ended
```

If the CGWY transaction reports an error message, refer to the description of the error message that is given in *TXSeries for Multiplatforms Messages and Codes*. Any SNA communications that are using the PPC Gateway server that is specified in the message can fail until the problem is solved.

Process for making an intersystem request from a CICS region through a PPC Gateway server

A CICS transaction can communicate with a remote system that is connected through a PPC Gateway server. Figure 59 on page 179 outlines the process that occurs when such a request is made.

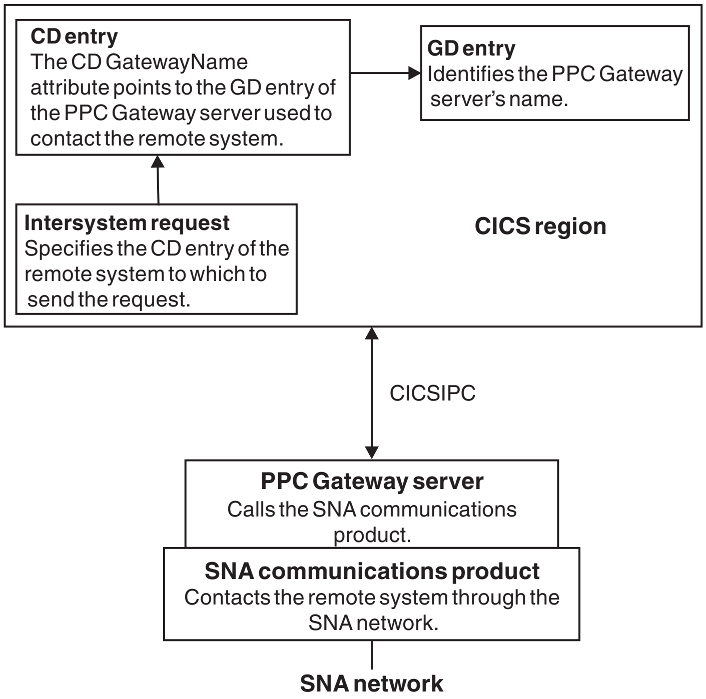

CD entry
The CD GatewayName attribute points to the GD entry of the PPC Gateway server used to contact the remote system.
GD entry
Identifies the PPC Gateway server's name.
Intersystem request
Specifies the CD entry of the remote system to which to send the request.
CICS region
CICSIPC
PPC Gateway server
Calls the SNA communications product.
SNA communications product
Contacts the remote system through the SNA network.
SNA network

Figure 59. Process for making an intersystem request from a CICS region through a PPC Gateway server

When a CICS transaction issues an intersystem request to a remote system, it specifies the name of the Communications Definitions (CD) entry that represents that remote system. The CD entry points to the relevant GD entry, using the **GatewayName** attribute. The GD entry contains the server name of the PPC Gateway server, which enables the region to pass the intersystem request to the PPC Gateway server. The PPC Gateway server then calls the SNA communications product to contact the remote system. (For information about how to configure CD and GD entries, refer to "Configuring CICS for PPC Gateway server SNA support" on page 89.)

Process for making an intersystem request to a CICS region through a PPC Gateway server

A remote system can request communications with the local CICS region. The process that occurs in this condition is shown in Figure 60 on page 180.

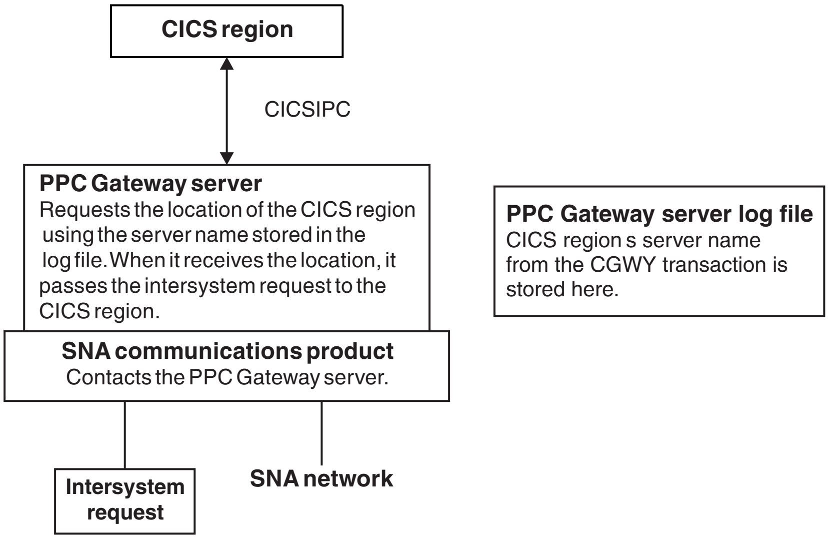

CICS region
CICSIPC
PPC Gateway server
Requests the location of the CICS region using the server name stored in the log file. When it receives the location, it passes the intersystem request to the CICS region.
PPC Gateway server log file
CICS region s server name from the CGWY transaction is stored here.
SNA communications product
Contacts the PPC Gateway server.
Intersystem request
SNA network

Figure 60. Process for making an intersystem request to a CICS region through a PPC Gateway server

When a PPC Gateway server receives an intersystem request for a CICS region from a remote SNA system, it uses the CICS region's server name. (The region's server name was passed to the PPC Gateway server previously by the CGWY transaction, and is stored in the PPC Gateway server's log file.) It can then pass the intersystem request to the CICS region.

If the PPC Gateway server detects that an intersystem request has failed, or that the SNA network is unavailable, it writes error messages to its message file **/var/cics_servers/GSD/cics/ppc/gateway/***server_name***/msg** on Open Systems or **C:\var\cics_servers\GSD\cics\ppc\gateway\***server_name***\msg** on Windows systems. These messages are described in "PPC Gateway server problem determination" on page 228.

Exchange log names (XLN) process

The PPC Gateway server supports synchronization level 2 (SL2). (Refer to "Ensuring data integrity with synchronization support" on page 16 for information about SL2.) As a result, it must record information in its log file about the CICS transactions that are using SL2 intersystem requests.

This information is used if a failure occurs during an SL2 intersystem request. It enables the PPC Gateway server to negotiate with the remote SNA system on behalf of the CICS region to ensure that the data on each system remains consistent. This negotiation is called *resynchronization*.

The PPC Gateway server must also record the names of the log files that are used by the remote SNA systems with which it is communicating. This information is required during resynchronization to ensure that the remote SNA system is using the same log file for the resynchronization process as it was using when the intersystem request failed. If the remote system (or the PPC Gateway server) has changed log files (because of a cold start, for instance), the resynchronization process is not reliable. In this case, administrator intervention is required to correct the data on the affected systems. (See "Using ppcadmin commands" on page 200 for more information about **ppcadmin** commands.)

The process by which the PPC Gateway server sends its log file name to a remote SNA system and receives the name of the remote SNA system's log file is called *exchange log names (XLN)*. The XLN process occurs every time a connection is established between the remote SNA system and the PPC Gateway server. SL2 intersystem requests are not permitted until the XLN process has been successful with the appropriate remote SNA system.

Choosing a management method

You can use several methods to install, configure, and run a PPC Gateway server. The following list shows the methods that are available and the tasks that can be performed with each:

- **cicscp** commands: The **cicscp** configuration tool provides an easy-to-use command line interface that automates configuration as much as possible by using default values where necessary and imposing some naming conventions.
  - "Creating a PPC Gateway server" on page 182
  - "Starting a PPC Gateway server" on page 183
  - "Stopping a PPC Gateway server" on page 184
  - "Destroying a PPC Gateway server" on page 184
- CICS commands: Low-level CICS commands allow the creation of more complex configurations than can be created by using the **cicscp** configuration tool.
  - "Creating a PPC Gateway server" on page 184
  - "Starting a PPC Gateway server" on page 186
  - "Stopping a PPC Gateway server" on page 187
  - "Destroying a PPC Gateway server" on page 188
  - "Changing the attributes of a PPC Gateway server" on page 188
  - "Listing the PPC Gateway servers defined on a machine" on page 188
  - "Viewing the attributes of a PPC Gateway server" on page 188
  - "Listing the PPC Gateway servers running on a machine" on page 189
  - "Releasing the lock of a PPC Gateway server" on page 189
- The IBM TXSeries Administration Tool: This tool provides a Graphical User Interface (GUI) method of configuring a PPC Gateway server on a Windows system.
  - "Creating a PPC Gateway server" on page 190
  - "Starting a PPC Gateway server" on page 191
  - "Stopping a PPC Gateway server" on page 193
  - "Modifying the attributes of a PPC Gateway server" on page 193
  - "Destroying a PPC Gateway server" on page 194
- SMIT: This tool provides a GUI method of configuring a PPC Gateway server on an AIX system.
  - "Creating a PPC Gateway server" on page 195
  - "Starting a PPC Gateway server" on page 197
  - "Stopping a PPC Gateway server" on page 198
  - "Viewing and changing the attributes of a PPC Gateway server" on page 199
  - "Destroying a PPC Gateway server" on page 200
- **ppcadmin** commands: These commands provide details about the status of transactions and the XLN process.
  - "Viewing CICS configuration in the PPC Gateway server" on page 203
  - "Viewing XLN process status in the PPC Gateway server" on page 204
  - "Requesting the XLN process with a remote SNA system" on page 205

- "Viewing pending resynchronizations in the PPC Gateway server" on page 206
- "Canceling pending resynchronizations in the PPC Gateway server" on page 206
- "Viewing intersystem requests running in the PPC Gateway server" on page 207
- "Viewing LUWs in the PPC Gateway server" on page 208

Using the CICS control program (cicscp) configuration tool to manage a PPC Gateway server

The **cicscp** configuration tool provides an easy-to-use command-line interface that automates the creation and configuration of a PPC Gateway server by using default values where necessary and imposing some naming conventions. For more information about the **cicscp** command set, see the *TXSeries for Multiplatforms Administration Reference*.

Use **cicscp** commands to perform the following system management tasks. For each task, it is assumed that you are logged on as **root** on Open Systems or with administrative privileges on Windows systems.

- "Creating a PPC Gateway server"
- "Starting a PPC Gateway server" on page 183
- "Stopping a PPC Gateway server" on page 184
- "Destroying a PPC Gateway server" on page 184

Creating a PPC Gateway server

To use the **cicscp create ppcgwy_server** command to create a PPC Gateway server:

1. Decide on a one- to eight-character *server_name* for the PPC Gateway server (for example, **cicsgwy**).
2. Enter the following command:

```
cicscp -v create ppcgwy_server server_name ShortName=shortName
```

This command creates the underlying structure and file base of the PPC Gateway server automatically. The following processes are completed:

- If the GSD stanza file does not exist, the **cicscp create ppcgwy_server** command creates it.
- If the **UserID** specified in the GSD for the server does not exist, the **cicscp create ppcgwy_server** command creates it with the appropriate home directory.
- If you do not specify a logical volume name in the **LogVolume** attribute of the GSD, the **cicscp create ppcgwy_server** command uses a default name of **log_%S**. If the logical volume does not exist, the **cicscp create ppcgwy_server** command creates it with a default size of 4 MB and a default location based on the platform. (As a result, the CICS_PPCGWY_VG and CICS_PPCGWY_SIZE environment variables do not need to be set.) If the logical volume already exists and is owned by the correct user, the **cicscp create ppcgwy_server** command issues a warning message. If it exists and is owned by a different user, the **cicscp create ppcgwy_server** command issues an error message.
- The **cicscp create ppcgwy_server** command creates a value for the server's GSD **ShortName** attribute of the form `P`*`baseName`*, where *baseName* is the first part of the created server name, truncated to seven characters if necessary. The new server name must be unique in its first seven characters to use the default **ShortName** value. If it is not, and you try to create a second server name with

the same first seven characters, an error occurs. For example, if you run the **cicscp create ppcgwy_server** command to create a server called `ppcgwysrv1`, it also creates a default **ShortName** value of `Pppcgwys`. If you then use the **cicscp create ppcgwy_server** command to create a second PPC Gateway server called `ppcgwysrv2`, the command attempts to create another **ShortName** value called `Pppcgwys`. Because the resulting **ShortName** value is identical to an existing one, the command issues an error. You can override the **ShortName** attribute default by specifying a value for the **ShortName** attribute when you issue the **cicscp create ppcgwy_server** command, as shown in step 2 on page 182.

- the **cicscp create ppcgwy_server** command adds an entry for the server to the **server_bindings** file that is in the **/var/cics_servers** directory on Open Systems or the **C:\var\cics_servers** directory on Windows systems. If this file does not exist, the **cicscp create ppcgwy_server** command creates it.
- If SNA is not configured on the machine, the **cicscp create ppcgwy_server** command issues a warning because the server cannot be started until SNA is configured and started.

Note: If you want to simultaneously create and start a new PPC Gateway server, use the **cicscp start ppcgwy_server** command as follows:

```
cicscp -v start ppcgwy_server server_name StartType=cold
```

Starting a PPC Gateway server

The **cicscp start ppcgwy_server** command offers two modes for starting a PPC Gateway server. A PPC Gateway server can be *cold started*. In this mode, the server is passed the name of its logical volume by the **cicscp start ppcgwy_server** command, and formats its logical volume with a new log file, destroying the recoverable information in the existing log file. Therefore, use a cold start only the first time that you start a PPC Gateway server. More frequently, a PPC Gateway server is started in *autostart* mode. In this mode, the server reads the recoverable information from the existing log file and proceeds as if it had not been shut down.

The following examples show how to use the **cicscp start ppcgwy_server** command to start a PPC Gateway server. If you accept the default **StartType** attribute value of **auto**, issue the command:

```
cicscp -v start ppcgwy_server server_name
```

If you want to change the **StartType** attribute value to **cold**, enter the command:

```
cicscp -v start ppcgwy_server server_name StartType=cold
```

It is possible to change the **ThreadPoolSize** and **SNADefaultModeName** attributes of the PPC Gateway server during either a cold start or an autostart. This PPC Gateway server continues to use the changed attributes on all subsequent autostarts. Changing a parameter during a start affects only the runtime database. Because the change is not placed in the permanent database, this attribute value is lost on subsequent cold starts.

Note: If you want to simultaneously create and start a new PPC Gateway server, use the **cicscp start ppcgwy_server** command as follows:

```
cicscp -v start ppcgwy_server server_name StartType=cold
```

In this case, because the *server_name* that you specify to start with the **cicscp start ppcgwy_server** command does not exist, this command creates the PPC Gateway server automatically using default values for all attributes. It then starts this server and applies any attribute values that you specify on the command line as changes to the defaults. Use this feature carefully because any changes to attributes that you specify as part of the **cicscp start ppcgwy_server** command do not take effect until the server is created. If the changes that you specify conflict with the default attributes, the command can fail.

If SNA is not configured, the **cicscp start ppcgwy_server** command issues an error message, and the PPC Gateway server is not started.

Stopping a PPC Gateway server

The **cicscp stop ppcgwy_server** command provides two modes for stopping a PPC Gateway server. A PPC Gateway server can be shut down in *Normal* mode, which means that it waits for all intersystem requests to complete before shutting down. Or it can be shut down in *Forced* mode, which causes it to close its files and stop immediately without waiting for intersystem requests to complete.

To stop a PPC Gateway server, issue the **cicscp stop ppcgwy_server** command. The following is an example of a normal shutdown (the default):

```
cicscp -v stop ppcgwy_server server_name
```

If you want to force the stop, enter:

```
cicscp -v stop ppcgwy_server server_name -f
```

Destroying a PPC Gateway server

To destroy a PPC Gateway server, issue the **cicscp destroy ppcgwy_server** command as shown in the following example:

```
cicscp -v destroy ppcgwy_server server_name
```

The **cicscp destroy ppcgwy_server** command stops the server if it is running, removes the server definition from the Gateway Definitions (GD), but does *not* remove the user ID or the logical volume.

Using CICS commands to manage a PPC Gateway server

CICS commands are used to perform the following system management tasks. For each task, it is assumed that you are logged on as **root** on Open Systems or with administrative privileges on Windows systems.

- “Creating a PPC Gateway server”
- “Starting a PPC Gateway server” on page 186
- “Stopping a PPC Gateway server” on page 187
- “Destroying a PPC Gateway server” on page 188
- “Changing the attributes of a PPC Gateway server” on page 188
- “Listing the PPC Gateway servers defined on a machine” on page 188
- “Viewing the attributes of a PPC Gateway server” on page 188
- “Listing the PPC Gateway servers running on a machine” on page 189
- “Releasing the lock of a PPC Gateway server” on page 189

Creating a PPC Gateway server

You create a PPC Gateway server by using the **cicsppcgwycreate** command. The following steps show how to use the **cicsppcgwycreate** command and suggest a simple naming convention for the PPC Gateway server’s resource definitions.

1. Decide on a one- to eight-character *server_name* for the PPC Gateway server (for example, **cicsgwy**).
2. Create an operating system user ID with this *server_name* and a logical volume of about one partition (4 MB) called `log_server_name` (for example, `log_cicsgwy`). The procedures that are used to create a user ID and a logical volume differ according to platform. See the bulleted points listed under this step for specific instructions on how to create a user ID and logical volume for your platform. In these steps, *server_name* represents the name of the PPC Gateway server.
   - To set up the PPC Gateway server user ID and logical volume on the AIX platform:
     a. Enter the following command to create the user ID:
        ```
        mkuser pgrp=cics home="/var/cics_servers/GSD/cics/ppc/gateway/sv_name"
        core=2097152 server_name
        ```
        This example shows the creation of a user ID for a PPC Gateway server named `server_name`. (Setting the attribute value `core=2097152` increases the size of the dumps that the PPC Gateway server is allowed to create.) The user ID must have the primary group **cics** and the home directory **/var/cics_servers/GSD/cics/ppc/gateway/***sv_name*.
     b. Enter the following command to create the logical volume:
        ```
        mklv -y log_server_name rootvg 4
        ```
        This command places a logical volume named `log_server_name` in the **/dev** directory. If possible, mirror this volume across more than one physical disk.
     c. Grant the PPC Gateway server user ID read and write permission to the logical volume and the associated raw device by issuing the following commands:
        ```
        chown server_name:cics /dev/log_server_name
        chown server_name:cics /dev/rlog_server_name
        ```
   - To set up the PPC Gateway server user ID and logical volume on the HP-UX platform, consult your HP documentation for information about how to use the HP System Administration Manager (SAM). If the logical volume is created on a volume group other than vg00, export the environment variable CICS_PPCGWY_VG to the name of the volume group that was used before starting the PPC Gateway server.
   - To set up the PPC Gateway server user ID and logical volume on the Windows platform:
     a. Change to the directory **C:\var\cics_servers\GSD\cics\ppc\gateway\** by entering the command:
        ```
        cd C:\var\cics_servers\GSD\cics\ppc\gateway\
        ```
     b. Make a new directory called *server_name* by entering the command:
        ```
        mkdir server_name
        ```
     c. Click **Administrative Tools (Common)>User Manager** to create a new account for the PPC Gateway server.
     d. Click **User>New User**. The New User screen is displayed.
     e. If you intend to specify a value for the **ShortName** attribute when creating your PPC Gateway server, enter that value in the **Username** field.
     f. Select the check box that is next to the **Password Never Expires** option.
     g. Click the **Profile** button.

h. In the **Home Directory** part of the screen, select the radio button that is next to the **Local Path** option and enter **C:\var\cics_servers\GSD\cics\ppc\gateway\***server_name* in the box to the right of this option.

i. Click **OK**.

j. Click the **Groups** button.

k. Click the **cicsgroup** entry under the **Not member of** portion of the screen.

l. Click the **Add** button.

m. Click **OK**.

n. Enter the following command to create the logical volume:

```
cicsmakelv -v log_svr_name -s volumeSize -p C:\var\log_Pserver_name
```

3. Follow the instructions under "Viewing the attributes of a PPC Gateway server" on page 188 to view the default values of the GSD attributes that will be assigned to the new PPC Gateway server.
4. Use one of the methods that are outlined under "Changing the attributes of a PPC Gateway server" on page 188 to change the default values of the following attributes:
   - Change the **ShortName** attribute of the GSD from the default of **PPCGWY** to the *server_name* that is defined in step 1 on page 185 because:
     - The value for the **ShortName** attribute must be unique among all PPC Gateway servers that are defined on the machine.
     - The value for the **ShortName** attribute affects the default values of the **UserID** and **LogVolume** attributes. If you do not change the **ShortName** attribute value, you must change the default values for the **UserID** and **LogVolume** attributes to the name of the user ID and logical volume that were created in step 2 on page 185.
5. Issue the **cicsppcgwycreate** command to create the PPC Gateway server:

```
cicsppcgwycreate /.:/cics/ppc/gateway/server_name
```

This command takes the name of the PPC Gateway server, which is always of the format **/.:/cics/ppc/gateway/***server_name*.

Note: You can specify changes to GSD attributes with this command. For example, if you want to change the **ShortName** attribute, the command is:

```
cicsppcgwycreate /.:/cics/ppc/gateway/server_name ShortName=svr_name
```

When using HP-UX SNAplus2

The PPC Gateway server must be configured with the modename that is associated with the Partner LU alias. Specify this value when the PPC Gateway server is created. The following example shows the PPC Gateway server called **/.:/cics/ppc/gateway/cicsgwy**, with the **ShortName** attribute changed to cicsgwy and the **SNADefaultModeName** attribute defined as CICSISC0:

```
cicsppcgwycreate /.:/cics/ppc/gateway/cicsgwy ShortName=cicsgwy  \
                 SNADefaultModeName="CICSISC0"
```

Starting a PPC Gateway server

The **cicsppcgwy** command offers two modes for starting a PPC Gateway server. A PPC Gateway server can be *cold started*. In this mode, the server is passed the name of its logical volume by the **cicsppcgwy** command and formats its logical volume with a new log file, destroying the recoverable information in the existing log file. Therefore, use a cold start only the first time you start a PPC Gateway server. More frequently, a PPC Gateway server is started in *autostart* mode. In this mode, the server reads the recoverable information from the existing log file and proceeds as if it had not been shut down.

To start a PPC Gateway server, issue the **cicsppcgwy** command. If you accept the default **StartType** attribute value of **auto**, enter the command:

```
cicsppcgwy /.:/cics/ppc/gateway/server_name
```

Note: In this example, the PPC Gateway server *server_name* is specified explicitly on the command line. You can also pass the server name on an autostart by using the CICS_PPCGWY_SERVER environment variable. In this case, issue the command to specify the variable before issuing the **cicsppcgwy** command. For example:

```
export CICS_PPCGWY_SERVER=/.:/cics/ppc/gateway/server_name
```

(This example assumes that you are using the Korn shell on an Open Systems platform; if you are using a different shell or platform, change the command accordingly.)

If you want to change the **StartType** attribute value to **cold**, enter the command:

```
cicsppcgwy /.:/cics/ppc/gateway/server_name StartType=cold
```

It is possible to change the PPC Gateway server's **ThreadPoolSize** and **SNADefaultModeName** attributes during either a cold start or an autostart.

After the PPC Gateway server is running, your CICS region can contact it. The region uses a Gateway Definitions (GD) entry to locate the PPC Gateway server. The PPC Gateway server then calls the SNA product to contact the remote system. (For information about how to configure GD entries, refer to "Configuring CICS for PPC Gateway server SNA support" on page 89.)

Stopping a PPC Gateway server

The **cicsppcgwyshut** command offers three modes for stopping a PPC Gateway server. A PPC Gateway server can be shut down in *Normal* mode, which means that it waits for all intersystem requests to complete before shutting down. It can be shut down in *Forced* mode, which causes it to close its files and stop immediately without waiting for intersystem requests to complete. Or it can be shut down in *Cancel* mode, which simply kills it. A *Cancel* mode shutdown leaves the PPC Gateway server in an undesirable state. Use it only if you have no other choice.

To stop a PPC Gateway server, issue the **cicsppcgwyshut** command. The following example command results in a normal shutdown (the default shutdown mode is `normal`):

```
cicsppcgwyshut /.:/cics/ppc/gateway/server_name
```

The following example specifies a forced shutdown:

```
cicsppcgwyshut -f /.:/cics/ppc/gateway/server_name
```

The following example specifies a cancel:

```
cicsppcgwyshut -c /.:/cics/ppc/gateway/server_name
```

Destroying a PPC Gateway server

The **cicsppcgwydestroy** command destroys a PPC Gateway server. Use this command with care because when the server is destroyed, all the data that is associated with it is lost.

To destroy a PPC Gateway server, issue the **cicsppcgwydestroy** command as shown in the following example:

```
cicsppcgwydestroy /.:/cics/ppc/gateway/server_name
```

Changing the attributes of a PPC Gateway server

Three wayare possible to update the attributes in a PPC Gateway server's GSD entry:

- You can use the **cicsppcgwy** command to change the **StartType**, **ThreadPoolSize**, and **SNADefaultModeName** attributes. (This process is described in "Starting a PPC Gateway server" on page 186.)
- You can use the **cicsppcgwydestroy** command, followed by the **cicsppcgwycreate** command, to destroy and re-create a PPC Gateway server. (These commands are described in "Destroying a PPC Gateway server" and "Creating a PPC Gateway server" on page 184, respectively.) This method allows you to change any of the GSD attributes. Use this method with care because, although you can re-create a PPC Gateway server with the **cicsppcgwycreate** command, all the data that is associated with the original is lost when you destroy the original server with the **cicsppcgwydestroy** command. When you start this new server after running the **cicsppcgwycreate** command, it is started as a cold start. (For more information about cold starts, refer to "Starting a PPC Gateway server" on page 186.)
- You can use the **cicsupdate -c gsd** command to change any GSD attribute.

Listing the PPC Gateway servers defined on a machine

You can list the PPC Gateway servers that have been created on a particular machine by using the **cicsget -c gsd -l** command. The **cicsget** command output shows the server name and the description (from the **ResourceDescription** attribute of the relevant GSD entry) for each PPC Gateway server. Figure 61 shows the output for a machine that has two PPC Gateway servers defined.

```
cicsget -c gsd -l
/.:/cics/ppc/gateway/cicsgwy    PPC Gateway server Definition
/.:/cics/ppc/gateway/cicsgwy2   PPC Gateway server Definition
```

Figure 61. Listing PPC Gateway servers with the cicsget command

Viewing the attributes of a PPC Gateway server

The attributes that are required to start and stop a PPC Gateway server are specified in a GSD entry. You can view the attributes of a PPC Gateway server by using the **cicsget -c gsd** command.

Figure 62 on page 189 shows the attributes of an example PPC Gateway server that has a server name of **/.:/cics/ppc/gateway/cicsgwy**.

```
cicsget -c gsd /.:/cics/ppc/gateway/cicsgwy

/.:/cics/ppc/gateway/cicsgwy:
ResourceDescription="PPC Gateway server Definition"
AmendCounter=0
Permanent=no
StartType=auto
ThreadPoolSize=10
ShortName="cicsgwy"
UserID="%S"
LogVolume="log_%S"
SNADefaultModeName=""
```

Figure 62. PPC Gateway server attributes for cicsgwy

The system substitutes the value that you enter in the **ShortName** field in the other fields where the **%S** value is displayed. For example, in Figure 62, if the value in the **UserID** field displays ″**%S**″, the system reads this value as *server_name*. Similarly, if the value in the **LogVolume** field shows ″**log_%S**″, the system reads this value as **log_***server_name*.

You can view the default values for the GSD attributes by using the **cicsget -c gsd ""** command. These default values are assigned to a PPC Gateway server when it is created unless you change them as described in “Changing the attributes of a PPC Gateway server” on page 188. Figure 63 shows an example of this command.

```
cicsget -c gsd ""

:ml3
ResourceDescription="PPC Gateway server Definition"
AmendCounter=0
Permanent=no
StartType=auto
ThreadPoolSize=10
ShortName="PPCGWY"
UserID="%S"
LogVolume="log_%S"
SNADefaultModeName=""
```

Figure 63. Default PPC Gateway server attributes

Listing the PPC Gateway servers running on a machine

You can list the PPC Gateway servers that are running on a machine, by using the **ps -ef | grep ppcgwy** command. Figure 64 shows a machine on which two PPC Gateway servers are running.

```
ps -ef | grep ppcgwy
cicsgwy 22377  2977  0 12:06:43 - 0:02  \
/opt/ibm.cics/bin/ppcgwy -n /.:/cics/ppc/gateway/cicsgwy -v /var/c
cicsgwy2 27009  2977 0 15:19:24 - 1:04  \
/opt/ibm/cics/bin/ppcgwy -n /.:/cics/ppc/gateway/cicsgwy2 -v /var/c
```

Figure 64. ps -ef | grep ppcgwy command

Releasing the lock of a PPC Gateway server

If a PPC Gateway server is stopped without the use of the **cicsppcgwyshut** command, the PPC Gateway server can be left in a locked state. This state prevents it from being restarted. If you are sure that the PPC Gateway server is not running, you can release the lock for it by using the **cicsppcgwylock** command with the **-u** option. After the lock is released, the **cicsppcgwy** command can be used to start the PPC Gateway server. The following example shows the lock being released for a PPC Gateway server named **/.:/cics/ppc/gateway/cicsgwy**:

```
cicsppcgwylock -u /.:/cics/ppc/gateway/cicsgwy
```

Using the IBM TXSeries Administration Tool to manage a PPC Gateway server

You can use the IBM TXSeries Administration Tool to perform the following tasks for PPC Gateway server system management on the Windows platform. For each task, it is assumed that you are logged onto Windows with administrative privileges.

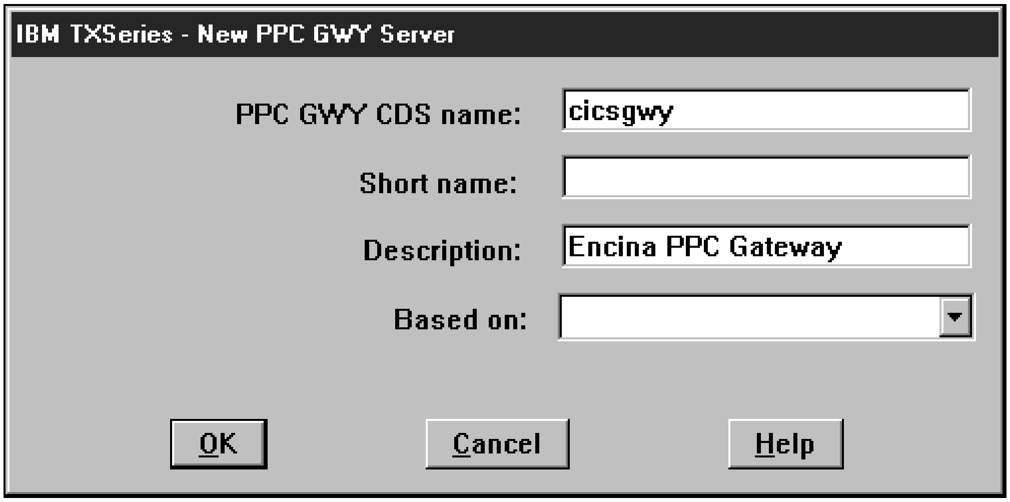
- "Creating a PPC Gateway server"
- "Starting a PPC Gateway server" on page 191
- "Stopping a PPC Gateway server" on page 193
- "Modifying the attributes of a PPC Gateway server" on page 193
- "Destroying a PPC Gateway server" on page 194

Creating a PPC Gateway server

Follow these steps to create a PPC Gateway server:

1. Start the IBM TXSeries Administration Tool. A list is displayed of the existing CICS regions, SFS file servers, and PPC Gateway servers that are on the host.
2. Click **Subsystem>New>PPC**. The New PPC GWY Server dialog box appears.
3. Enter the name for the new PPC Gateway server in the **PPC GWY CDS name** field.
4. Enter a value in the **Short name** field if you do not want to use the system-generated default value. You cannot modify the value of this attribute later.

   The default short name that is generated for the PPC Gateway server takes the form of **P***basename*, where *basename* is the PPC Gateway server namethat you entered in the **PPC GWY CDS name** field, truncated to seven characters if necessary. Therefore, you must use PPC Gateway server names that are unique in their first seven characters.
5. Change or accept the default description.
6. If you want to use an existing PPC Gateway server definition as the basis for this new PPC Gateway server, enter its name in the **Based on** field, or select an entry from this field's drop-down list. An example configuration is shown in Figure 65 on page 191. (In Figure 65 on page 191, leaving the **Short name** field blank causes the default value to be used, as described in step 4.)



IBM TXSeries - New PPC GWY Server

PPC GWY CDS name: cicsgwy

Short name:

Description: Encina PPC Gateway

Based on:

Figure 65. New PPC GWY Server screen with example configuration

7. Click **OK** to create the new PPC Gateway server. A message box informs you whether the PPC Gateway server has been created successfully.
8. Click **OK** to exit the message box.

The following tasks are completed for you when you use the above procedure:

- The **Short name** attribute is set.
- The **Server user ID** is defined if it does not currently exist.
- A **Log volume** that is used for recovery information is created in the **\var** subdirectory, if none already exists.
- If the PPC Gateway server is to run in an RPC-only environment, the necessary server binding-file entry is set up. If the **server_bindings** file does not exist, it is created.

When you have created your PPC Gateway server, consider whether to use the default values for the following attributes. If you decide to change them, use the procedure that is described in “Modifying the attributes of a PPC Gateway server” on page 193.

Note: The following attributes appear on various tabs of the Properties screen.

- **Size of Thread Pool**

  This attribute determines the number of operations that can run concurrently. It can be set in the range 1 through 20. The default is **10**.
- **Protect resource**

  This attribute specifies whether CICS permits you to change or delete the permanent database entry. If you check the check box, you cannot change or delete the entry. The default is unselected (the check box is cleared), meaning that your entry is not protected from being changed or deleted.

Starting a PPC Gateway server

The IBM TXSeries Administration Tool supports two modes for starting the PPC Gateway server:

- **Cold start**

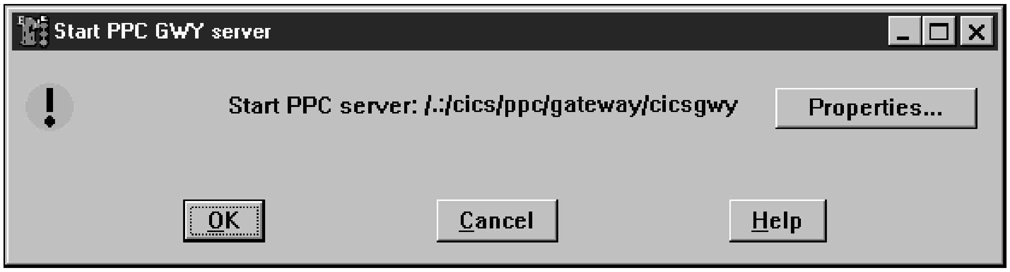
  In this mode, the PPC Gateway server performs as if it has never been started. It is passed the name of its logical volume and formats this logical volume with a new log file. If the server has been started before, this action destroys the recoverable information in the existing log file. Therefore, use a cold start only the first time that the PPC Gateway server is started.

- **Autostart**

  In this mode, the PPC Gateway server reads the recoverable information from the log file and proceeds as if it had not been shut down.

To start a PPC Gateway server:

1. Start the IBM TXSeries Administration Tool. A list displays the existing CICS regions, SFS file servers, and PPC Gateway servers on the host.
2. From the list of servers on the host, select the entry for the required PPC Gateway server.
3. Click **Subsystem>Start** to display the Start PPC GWY server dialog box, as shown in Figure 66.



Start PPC GWY server

Start PPC server: /.:/cics/ppc/gateway/cicsgwy

Figure 66. Start PPC GWY server screen

   Note: If you click the **Start All** option, you can select **Servers**, **CICS Regions**, **SFS Servers**, or **PPC Gateways**. A dialog box is displayed, prompting you to confirm or cancel the start of the systems you have selected.

4. By default, the **Start type** attribute for a PPC Gateway server is set to **auto**, causing CICS to restart it from the state in which it was last shut down. To verify the attribute setting, or to change it to **cold** to invoke a cold start, click the **Properties** button and make the change before proceeding to the next step.
5. Click **OK** to confirm the action. A message box informs you whether the PPC Gateway server has started successfully.
6. Click **OK** to exit the message box.

When starting the PPC Gateway server, you might see one of the following messages:

PCS6002A
Some of your configuration settings were not applied successfully. You can ignore this message; SNA will be started successfully.

PCS6007A
Cannot retrieve the default configuration file name. If this message is encountered when SNA is starting, it might be because you need to set a default SNA configuration file that will be used each time you use the Administration Console to start the PPC Gateway server. To set this default, issue the following command from the command line:

```
csstart -d path_to_sna_config_file
```

For example:

```
csstart -d C:\ibmcs\private\mySNAconfig.acg
```

You will now be able to start the PPC Gateway server.

Stopping a PPC Gateway server

The IBM TXSeries Administration Tool supports two modes for stopping the PPC Gateway server:

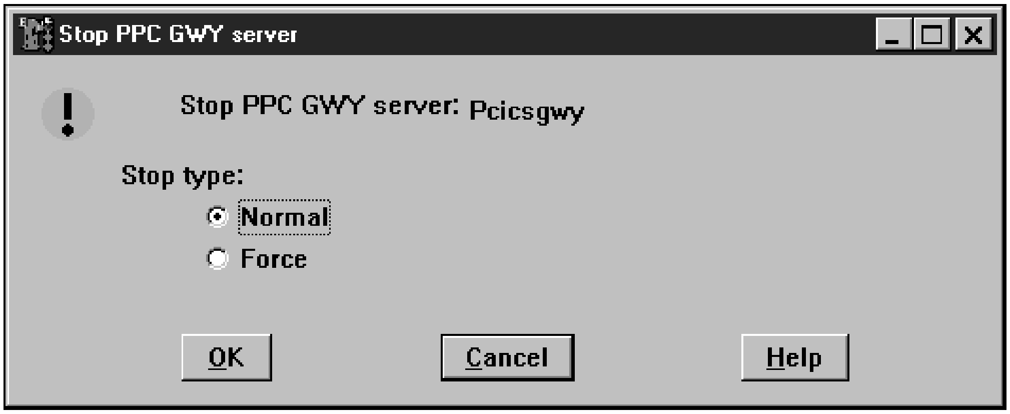
- **Normal**

  The server waits for all intersystem requests to complete before it shuts down. This is the default.
- **Forced**

  The server closes its files and stops immediately without waiting for intersystem requests to complete.

To shut down a PPC Gateway server:

1. Start the IBM TXSeries Administration Tool. A list displays the existing CICS regions, SFS file servers, and PPC Gateway servers that are on the host.
2. From the list of servers on the host, select the entry for the required PPC Gateway server.
3. Click **Subsystem>Stop**. The Stop PPC GWY server dialog box is displayed, as shown in Figure 67.



Stop PPC GWY server: Pcicsgwy

Stop type:

● Normal

○ Force

Figure 67. Stop PPC GWY server screen

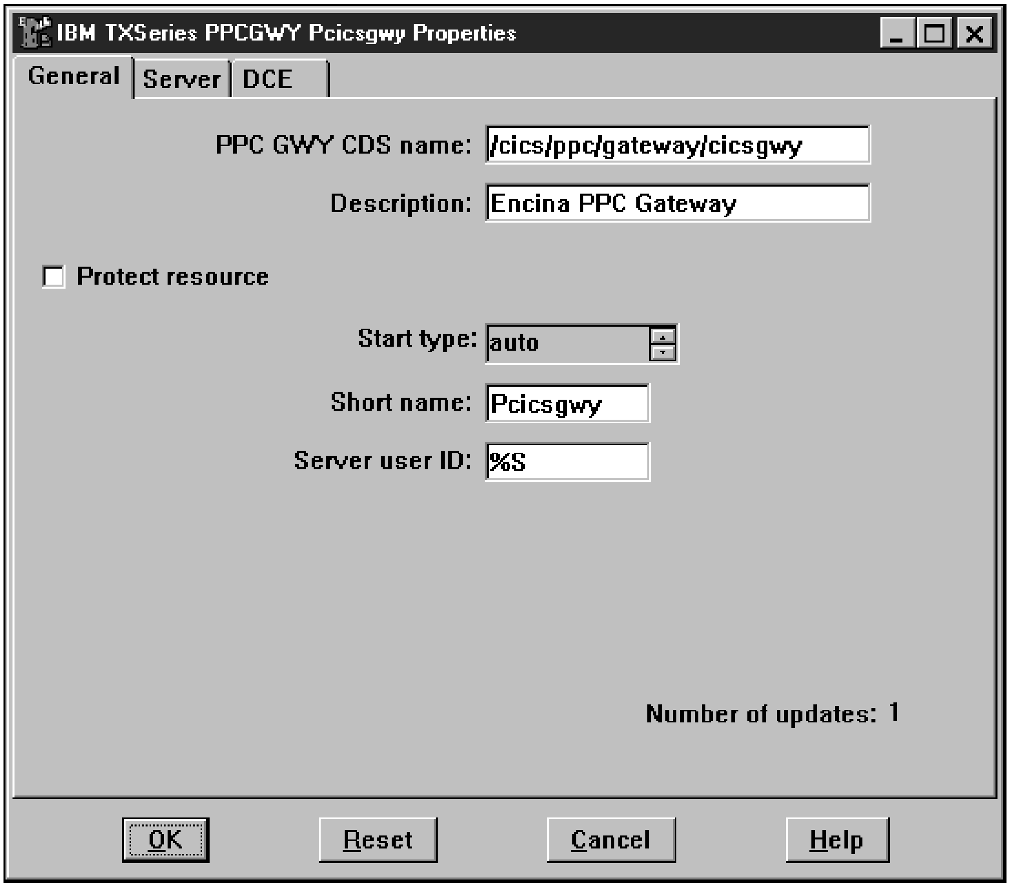
   Note: If you select the **Stop All** option, you can select **Servers**, **CICS Regions**, **SFS Servers**, or **PPC Gateways**. A dialog box is displayed, prompting you to confirm or cancel the action to stop the systems that you have selected.
4. If you require an immediate stop, click the **Force** radio button.
5. Click **OK** to stop the PPC Gateway server. A message box informs you whether the PPC Gateway server has stopped successfully.
6. Click **OK** to exit the message box.

Modifying the attributes of a PPC Gateway server

To change the values of an existing PPC Gateway server:

1. Start the IBM TXSeries Administration Tool. A list displays the existing CICS regions, SFS file servers, and PPC Gateway servers on the host.
2. From the list of servers on the host, select the entry for the PPC Gateway server that you want to modify.

3. Click **Subsystem>Properties**. The Properties screen for the PPC Gateway server is displayed. An example is shown in Figure 68.



PPC GWY CDS name: /cics/ppc/gateway/cicsgwy

Description: Encina PPC Gateway

□ Protect resource

Start type: auto

Short name: Pcicsgwy

Server user ID: %S

Number of updates: 1

Figure 68. PPC Gateway server Properties screen—General tab

4. From the **General** or **Server** tabs, select or enter new values for the attributes that you want to change.
5. Click **OK** to implement the changes.

Destroying a PPC Gateway server

Note: Before destroying a PPC Gateway server, ensure that it is not still required for any other CICS regions. Use this facility with care because all the data that is associated with the PPC Gateway server is lost when the server is destroyed.

To destroy a PPC Gateway server:

1. Start the IBM TXSeries Administration Tool. A list displays the existing CICS regions, SFS file servers, and PPC Gateway servers on the host.
2. From the list of servers on the host, select the entry for the required PPC Gateway server.
3. Click **Subsystem>Destroy**. The Destroy PPC GWY Server dialog box displays, as shown in Figure 69 on page 195.

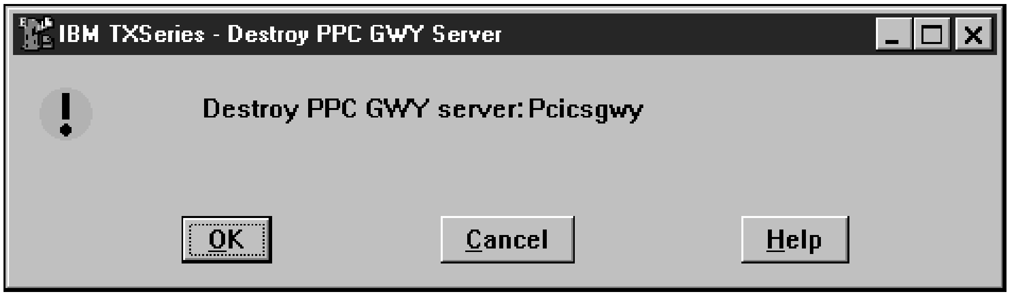

IBM TXSeries - Destroy PPC GWY Server
Destroy PPC GWY server: Pcicsgwy
OK
Cancel
Help

Figure 69. Destroy PPC GWY Server screen

Note: If you select the **Destroy All** option, you can indicate **Servers**, **CICS Regions**, **SFS Servers**, or **PPC Gateways**. A dialog box is displayed, prompting you to confirm or cancel the action that you have selected.

4. Click **OK** to destroy the PPC Gateway server. This action both stops the PPC Gateway server and deletes the associated GSD entry. A message box informs you that the PPC Gateway server has been destroyed successfully.
5. Click **OK** to close the message box.

Note: When you destroy a PPC Gateway server, the user ID and logical volume are not destroyed. You must delete the user ID and logical volume separately (for example, by using the Windows User Manager Tool and Windows Explorer, respectively).

Using SMIT to manage a PPC Gateway server

The System Management Interface Tool (SMIT) can be used to perform the following tasks for PPC Gateway server system management on an AIX platform. For each task, it is assumed that you are logged on as **root**.

- "Creating a PPC Gateway server"
- "Starting a PPC Gateway server" on page 197
- "Stopping a PPC Gateway server" on page 198
- "Viewing and changing the attributes of a PPC Gateway server" on page 199
- "Destroying a PPC Gateway server" on page 200

Creating a PPC Gateway server

The following steps show how to use SMIT to create a PPC Gateway server:

1. Decide on a one- to eight-character *server_name* for the PPC Gateway server (for example, **cicsgwy**).
2. Create an operating system user ID with this *server_name*. It must have the primary group **cics** and the home directory **/var/cics_servers/GSD/cics/ppc/gateway/***server_name*.

   The following example shows the creation of a user ID for a PPC Gateway server named *server_name*. (Setting the attribute value `core=2097152` increases the size of the dumps that the PPC Gateway server is allowed to create.)

   ```
   mkuser pgrp=cics home="/var/cics_servers/GSD/cics/ppc/gateway/server_name"
   core=2097152 server_name
   ```
3. Create a logical volume of about one partition (4 MB) called log_*server_name* (for example, `log_cicsgwy`). If possible, mirror this volume across more than one physical disk. The following example places a logical volume named log_*server_name* in the **/dev** directory.

   ```
   mklv -y log_server_name rootvg 4
   ```

4. Change the ownership of the logical volume to give read and write permission to the new user ID that was in step 2 on page 195. The following example gives the user ID for *server_name* access to the logical volume:

   ```
   chown server_name:cics /dev/log_server_name
   chown server_name:cics /dev/rlog_server_name
   ```

5. Start SMIT by entering the following command:

   ```
   smitty cics
   ```

6. Select the following options:

   ```
   ► Manage  PPC Gateway Servers
       ► Define  PPC Gateway Servers
           ► Create
   ```

   The Create PPC Gateway Server screen is displayed.

7. Enter a value or accept the default value of ("") in the **Model PPC Gateway Server Identifier** field and press **Enter**.
8. On the expanded screen that is displayed, enter a value in the **PPC Gateway Server Identifier** field in the form:

   ```
   /.:/cics/ppc/gateway/server_name
   ```

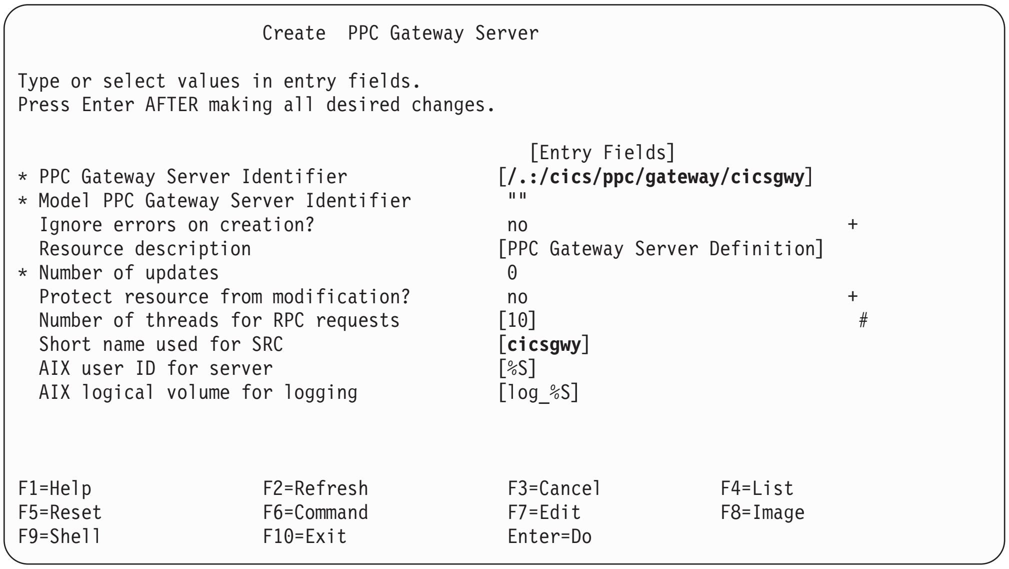
   where *server_name* is the server name that you defined in step 1 on page 195.

9. Enter this *server_name* as the value for the field **Short name used for SRC**. The system substitutes the value that you enter in this field, in the other fields where the **%S** value is displayed. For instance, if the value in the **AIX user ID for server** field displays **%S**, the system reads this value as *server_name*. Similarly, if the value in the **AIX logical volume for logging** field displays **log_%S**, the system reads this value as **log_***server_name*. These values match the user ID and logical volume that were created in step 2 on page 195 and step 3 on page 195.
10. Consider whether to use the default values for the following attributes:
    - **Number of threads for RPC requests**

      This attribute determines the number of operations that can run concurrently. It can be set in the range 1 through 20. The default is **10**.
    - **Protect resource from modification?**

      This attribute specifies whether CICS permits you to change or delete the permanent database entry. If you enter the value **yes** in this field, you cannot change or delete the entry in the permanent database. The default is **no**, meaning that your entry is not protected from being changed or deleted.

    An example Create PPC Gateway Server screen is shown in Figure 70 on page 197.



```
                    Create  PPC Gateway Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                              [Entry Fields]
* PPC Gateway Server Identifier            [/.:/cics/ppc/gateway/cicsgwy]
* Model PPC Gateway Server Identifier       ""
  Ignore errors on creation?                no                           +
  Resource description                     [PPC Gateway Server Definition]
* Number of updates                         0
  Protect resource from modification?       no                           +
  Number of threads for RPC requests       [10]                           #
  Short name used for SRC                  [cicsgwy]
  AIX user ID for server                   [%S]
  AIX logical volume for logging           [log_%S]



F1=Help               F2=Refresh            F3=Cancel           F4=List
F5=Reset              F6=Command            F7=Edit             F8=Image
F9=Shell              F10=Exit              Enter=Do
```

Figure 70. Create PPC Gateway Server screen

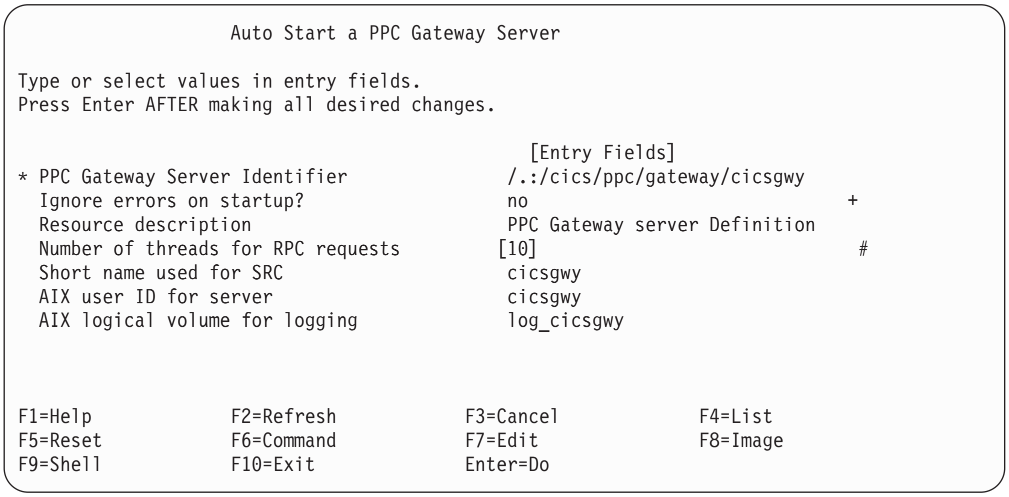
11. When all the fields are set, press **Enter** to create the PPC Gateway server.

Starting a PPC Gateway server

SMIT supports two modes for starting a PPC Gateway server. A PPC Gateway server can be *cold started*. In this mode, the server formats its logical volume with a new log file, and destroys the recoverable information that is in the existing log file. Therefore, use a cold start only the first time that you start a PPC Gateway server. More frequently, a PPC Gateway server is started in *autostart* mode. In this mode, the server reads the recoverable information from the existing log file and proceeds as if it had not been shut down.

The following steps show how to use SMIT to start a PPC Gateway server:

1. Optionally, set the environment variable CICS_PPCGWY_SERVER to the name of the PPC Gateway server. The following example command assumes that you are using the Korn shell; if you are using a different shell, change the command accordingly:
   ```
   export CICS_PPCGWY_SERVER=/.:/cics/ppc/gateway/server_name
   ```
2. Start SMIT by entering the following command:
   ```
   smitty cics
   ```
3. Select the following option:
   ```
   ► Manage PPC Gateway Servers
   ```
4. If you set the CICS_PPCGWY_SERVER environment variable before starting SMIT, go to step 7. (Step 4, step 5, and step 6 are required only if you have not set this environment variable before starting SMIT.) If you have not set this environment variable, select the **Change Working PPC Gateway Server** option to select which PPC Gateway server to start.
5. Select the server that you want to start, and press **Enter**. The COMMAND STATUS screen confirms your selection.
6. Press F3 to return to the Manage PPC Gateway Servers screen.
7. Select either **Cold Start a PPC Gateway Server** or **Auto Start a PPC Gateway Server**, depending on the type of start that you require. An example Auto Start a PPC Gateway Server screen is shown in Figure 71 on page 198.



```
                   Auto Start a PPC Gateway Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
* PPC Gateway Server Identifier               /.:/cics/ppc/gateway/cicsgwy
  Ignore errors on startup?                   no                             +
  Resource description                        PPC Gateway server Definition
  Number of threads for RPC requests         [10]                             #
  Short name used for SRC                     cicsgwy
  AIX user ID for server                      cicsgwy
  AIX logical volume for logging              log_cicsgwy



F1=Help             F2=Refresh          F3=Cancel          F4=List
F5=Reset            F6=Command          F7=Edit            F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 71. Auto Start a PPC Gateway Server screen

8. In this screen, you can change the values for the **Number of threads for RPC requests** attribute. Any changes that are made are remembered for subsequent autostarts.
9. Press **Enter** to start the PPC Gateway server.

After the PPC Gateway server is running, your CICS region can contact it. The region uses a Gateway Definitions (GD) entry to locate the PPC Gateway server. Refer to “Configuring CICS for PPC Gateway server SNA support” on page 89 for information about how to configure a GD entry.

Stopping a PPC Gateway server

SMIT supports three modes for stopping a PPC Gateway server. A PPC Gateway server can be shut down in *Normal* mode, which means that it waits for all intersystem requests to complete before shutting down. It can be shut down in *Immediate* mode, which causes the server to close its files and stop immediately without waiting for intersystem requests to complete. Or it can be shut down in *Cancel* mode, which simply kills the PPC Gateway server. A Cancel mode shutdown leaves the PPC Gateway server in an undesirable state. Use it only if you have no other choice.

The following steps show how to use SMIT to stop a PPC Gateway server:

1. Optionally, set the CICS_PPCGWY_SERVER environment variable to the name of the PPC Gateway server. The following example command assumes that you are using the Korn shell; if you are using a different shell, change the command accordingly:

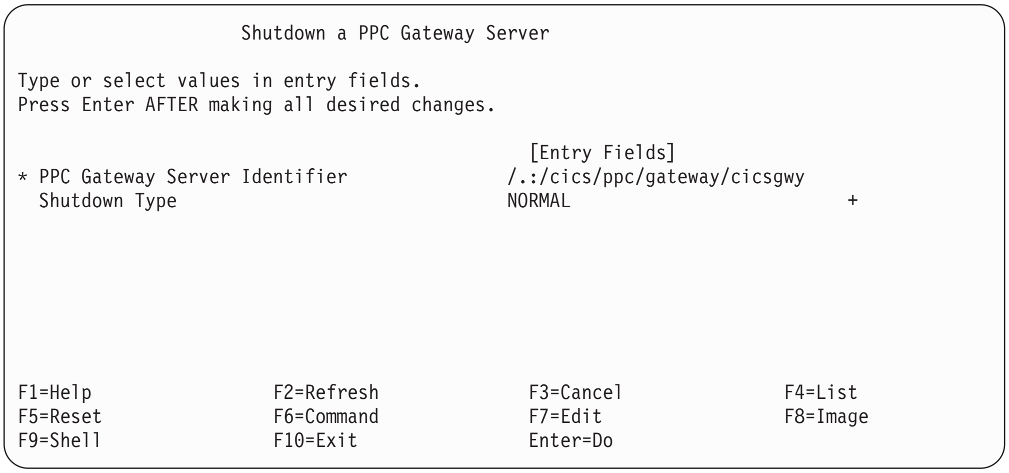
   ```
   export CICS_PPCGWY_SERVER=/.:/cics/ppc/gateway/server_name
   ```
2. Start SMIT by entering the following command:

   ```
   smitty cics
   ```
3. Select the following option:

   ► `Manage PPC Gateway Servers`
4. If you set the CICS_PPCGWY_SERVER environment variable before starting SMIT, go to step 7 on page 199. (Step 4, step 5 on page 199, and step 6 on page 199 are required only if you have not set the CICS_PPCGWY_SERVER environment variable before starting SMIT.) If you have not set this environment variable, select the **Change Working PPC Gateway Server** option to select which PPC Gateway server to stop.

5. Select the server that you want to stop, and press **Enter**. The COMMAND STATUS screen confirms your selection.
6. Press F3 to return to the Manage PPC Gateway Servers screen.
7. Select the **Shutdown a PPC Gateway Server** option. The Shutdown a PPC Gateway Server screen is displayed.
8. In the **Shutdown Type** field, select the type of shutdown that you require:
   - NORMAL: A normal shutdown
   - IMMEDIATE: A forced shutdown
   - CANCEL: To cancel the PPC Gateway server

   An example Shutdown a PPC Gateway Server screen is shown in Figure 72.



```
                     Shutdown a PPC Gateway Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                 [Entry Fields]
* PPC Gateway Server Identifier               /.:/cics/ppc/gateway/cicsgwy
  Shutdown Type                               NORMAL                          +









F1=Help                F2=Refresh             F3=Cancel              F4=List
F5=Reset               F6=Command             F7=Edit                F8=Image
F9=Shell               F10=Exit               Enter=Do
```

Figure 72. Shutdown a PPC Gateway Server screen

9. Press **Enter** to stop the PPC Gateway server.

Viewing and changing the attributes of a PPC Gateway server

Two ways are possible use SMIT to view and update the attributes of a PPC Gateway server:

- Use the Cold Start of a PPC Gateway Server screen or the Auto Start of a PPC Gateway Server SMIT screen. Only **Number of threads for RPC requests** field can be changed in these screens. (For more information about how to start a PPC Gateway server, refer to "Starting a PPC Gateway server" on page 197.)
- Use the PPC Gateway server's Show/Change screen. You can change any of the server's attributes from this screen. However, the update process first destroys, then re-creates the PPC Gateway server. Use this method with care because all the data that is associated with the server is lost. As a result, the next time the server is started, it is started as a cold start. (For more information about cold starts, refer to "Starting a PPC Gateway server" on page 197.)

The following steps show how to use SMIT to view and change a PPC Gateway server on the server's Show/Change screen:

1. Start SMIT by entering the following command:

   ```
   smitty cics
   ```
2. Select the following options:

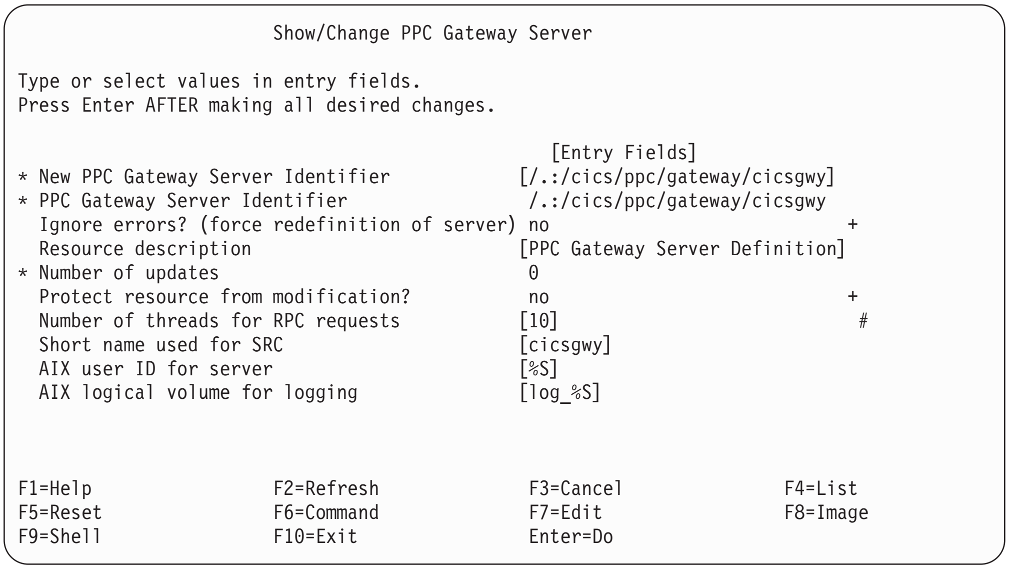
   ```
   ► Manage  PPC Gateway Servers
       ► Define PPC Gateway Servers
           ► Show/Change
   ```

The Select a PPC Gateway Server for Show/Change screen is displayed.

3. In the **PPC Gateway Server Identifier** field, enter a PPC Gateway server identifier in the form:

   `/.:/cics/ppc/gateway/`*`server_name`*

4. Press **Enter**. The Show/Change PPC Gateway Server screen is displayed. An example screen is shown in Figure 73.



```
                        Show/Change PPC Gateway Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                  [Entry Fields]
* New PPC Gateway Server Identifier            [/.:/cics/ppc/gateway/cicsgwy]
* PPC Gateway Server Identifier                 /.:/cics/ppc/gateway/cicsgwy
  Ignore errors? (force redefinition of server) no                            +
  Resource description                         [PPC Gateway Server Definition]
* Number of updates                             0
  Protect resource from modification?           no                            +
  Number of threads for RPC requests           [10]                            #
  Short name used for SRC                      [cicsgwy]
  AIX user ID for server                       [%S]
  AIX logical volume for logging               [log_%S]



F1=Help                 F2=Refresh              F3=Cancel               F4=List
F5=Reset                F6=Command              F7=Edit                 F8=Image
F9=Shell                F10=Exit                Enter=Do
```

Figure 73. Show/Change PPC Gateway Server screen

5. If you do not want to change the PPC Gateway server, press F3 to return to the Define PPC Gateway Servers screen. Otherwise, change the required attributes, and press **Enter** to destroy and re-create the server with the new values.

Destroying a PPC Gateway server

SMIT can be used to destroy a PPC Gateway server. Use this facility with care because all the data that is associated with the PPC Gateway server is lost when the server is destroyed.

The following steps show how to use SMIT to destroy a PPC Gateway server:

1. Start SMIT by entering the following command:

   ```
   smitty cics
   ```

2. Select the following options:

   ```
   ► Manage PPC Gateway Servers
       ► Define PPC Gateway Servers
           ► Destroy
   ```

   The Destroy a PPC Gateway Server screen is displayed.

3. In the **PPC Gateway Server Identifier** field, enter a PPC Gateway server identifier in the form:

   `/.:/cics/ppc/gateway/`*`server_name`*

4. Press **Enter** to destroy the PPC Gateway server.

Using ppcadmin commands

The PPC Gateway server has a set of administration commands called **ppcadmin** that allow you to view its status. The set of commands can display:

- The CICS regions that have passed configuration data to the PPC Gateway server, by using the CGWY transaction
- The names of the remote SNA systems in which the exchange log names (XLN) process is required or has been successful
- The remote SNA systems with which the PPC Gateway server is waiting to resynchronize, and the CICS transactions that are affected
- The current intersystem requests that are using the PPC Gateway server

Table 37 lists the **ppcadmin** commands that are applicable to the CICS environment.

Table 37. ppcadmin commands that are applicable to the CICS environment

Command	Use
ppcadmin help	Show syntax of a **ppcadmin** command (This command lists all the **ppcadmin** commands, some of which are not applicable to the CICS environment.)
ppcadmin list luentries	List all CICS regions that have configured the PPC Gateway server
ppcadmin query luentry	View a CICS region's configuration in the PPC Gateway server
ppcadmin list xlns	List XLN status for all connections
ppcadmin query xln	Query XLN status for the specified connection
ppcadmin force xln	Force an XLN for the specified connection
ppcadmin list convs	List all active intersystem requests
ppcadmin query conv	Query an active intersystem request
ppcadmin query stats	Query the conversation statistics
ppcadmin list luws	List all active Logical Units of Work (LUWs)
ppcadmin query luw	Query an active LUW
ppcadmin list transactions	List all active transactions
ppcadmin query transaction	Query an active transaction
ppcadmin list resyncs	List all pending resynchronizations
ppcadmin query resync	Query the specified resynchronization
ppcadmin cancel resync	Cancel all pending resynchronizations for a connection

Note: If the CICS region runs on a different operating system from that on which the PPC Gateway server runs, you can issue a **ppcadmin** command from either the machine on which the PPC Gateway server runs, or from the machine on which the CICS region runs.

Use the following procedure to use a **ppcadmin** command (this procedure assumes that you are using the Korn shell on an Open Systems platform; if you are using a different platform or shell, change the **export** commands accordingly):

1. Decide on the specific **ppcadmin** command that you require from the list that is shown in Table 37.
2. Determine the PPC Gateway server's name. The **ppcadmin** commands require the PPC Gateway server's name, which is passed to the **ppcadmin** command by using the command's **-server** *server_name* option or the environment variable CICS_PPCGWY_SERVER. If you are planning to issue several

ppcadmin commands to a PPC Gateway server, it is easier to set the CICS_PPCGWY_SERVER environment variable before calling the first **ppcadmin** command as follows:

```
export CICS_PPCGWY_SERVER=/.:/cics/ppc/gateway/server_name
```

3. Issue the **ppcadmin** command. The **ppcadmin** commands can be run in a command mode or in an interactive mode, as shown in Figure 74 and Figure 75. Figure 74 shows a **ppcadmin** command issued in interactive mode.

```
export CICS_PPCGWY_SERVER=/.:/cics/ppc/gateway/cicsgwy
ppcadmin
 PPC administration tool.
Type "help" for help, "exit" to exit.

ppcadmin> help list luentries
NAME
    ppcadmin list luentries -- List all registered executive LU entries

SYNOPSIS
    ppcadmin list luentries [-server server]

ppcadmin> list luentries

Command executed at: Mon Oct 11 12:44:18 1999


total local LU entries: 1

        Executive LU: CICSOPEN

ppcadmin> exit
```

Figure 74. Using ppcadmin commands in interactive mode

To enter this mode, enter the command suite name with no arguments (for example, `ppcadmin`). When you do so, the prompt becomes the name of the command suite, which allows you to enter only the verb, object, and arguments for subsequent commands.

Figure 75 shows the **ppcadmin** command issued in command mode:

```
export CICS_PPCGWY_SERVER=/.:/cics/ppc/gateway/cicsgwy
ppcadmin help list luentries
  NAME
      ppcadmin list luentries -- List all registered executive LU entries

  SYNOPSIS
      ppcadmin list luentries [-server server]
ppcadmin list luentries

Command executed at: Mon Oct 11 12:45:46 1999


total local LU entries: 1

        Executive LU: CICSOPEN
```

Figure 75. Using ppcadmin commands in command mode

The sections that follow provide information about selected **ppcadmin** commands. For information about the whole **ppcadmin** command set, see *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration*.

Viewing CICS configuration in the PPC Gateway server

Use the **ppcadmin list luentries** command to list the Logical Unit (LU) names that are configured in the PPC Gateway server. An example of this command and its output is shown in Figure 76.

```
ppcadmin list luentries

Command executed at: Mon Oct 11 09:43:36 1999


total local LU entries: 2

        Executive LU: CICSOPEN
        Executive LU: CICSLUGW
```

Figure 76. ppcadmin list luentries command

In Figure 76, a CICS region's LU name is shown as an `Executive LU`. This is the LU name that is configured in the CICS region's Gateway Definitions (GD) **GatewayLUName** attribute.

The PPC Gateway server, **/.:/cics/ppc/gateway/cicsgwy**, has two LUs configured: `CICSOPEN` and `CICSLUGW`. The LU `OPENCICS` is for the CICS region **cicsopen**, and the LU `CICSLUGW` is for the CICS region **cicsaix**.

If you are using a PPC Gateway server on the AIX platform with the IBM Communications Server for AIX, you can use the **ppcadmin query luentry** command to show the CICS transactions for which the PPC Gateway server is listening on behalf of the region's LU name. To use this command, you must provide the LU name for the CICS region about which you are querying. Figure 77 on page 204 shows an example of this command and its output.

```
ppcadmin query luentry CICSLUGW

Command executed at: Mon Oct 11 10:48:05 1999


        Executive LU: CICSLUGW
                Total TPN profiles: 1
                TPN Profile: CICSTPN
                Total TP Names: 30
                 DTP1
                 CEMT
                 CVMI
                 CRSR
                 RECV
                 TEST
                 ACCT
                 BANK
                 CSM1
                 \01
                 CSM2
                 \02
                 CSM3
                 \03
                 CESF
                 ACC0
                 CSM5
                 \05
                 ACC1
                 CSSF
                 ADDR
                 CECI
                 CESN
                 CEBR
                 CEDF
                 CPMI
                 CALF
                 CECS
                 CSMI
                 CRTE
```

Figure 77. ppcadmin query luentry command

In Figure 77, the PPC Gateway server is queried about the CICS transactions that are associated with LU name `CICSLUGW`. The listed transactions have been passed to the PPC Gateway server because the **cicsaix** region's Transaction Definitions (TD) **TPNSNAProfile** attribute is set to the name of an IBM Communications Server for AIX TPN profile. In this example, the profile is called `CICSTPN`. The PPC Gateway server can receive requests only from remote SNA systems for the transactions that are listed by this command.

The transactions `\01`, `\02`, `\03`, and `\05` are hexadecimal versions of the CICS transactions CSM1, CSM2, CSM3 and CSM5. These hexadecimal transaction names are used by some CICS systems for function shipping requests. The hexadecimal transaction names do not have TD entries. Each is given to the PPC Gateway server whenever its CSM*x* equivalent transaction name is passed.

Note: When switching between local SNA and a PPC Gateway server, be sure to purge the old LU entry in the region's Gateway Definitions (GD) to avoid duplicate LU names. Refer to "Configuring CICS for PPC Gateway server SNA support" on page 89 for information about configuring GD entries.

Viewing XLN process status in the PPC Gateway server

Use the **ppcadmin list xlns** command to view the status of the XLN processes that are occurring between the CICS region and the remote SNA systems with which it is communicating. (See "Exchange log names (XLN) process" on page 180 for an overview of the XLN process.) An example of this command and its output is shown in Figure 78.

```
ppcadmin list xlns

Command executed at: Mon Oct 11 12:48:16 1999


Number of LU pairs: 2

Executive LU: CICSLUGW
SNA Partner LU: CICSESA
Log Identifier Name Exchange Complete: YES

Executive LU: CICSLUGW
SNA Partner LU: CICSMVS
Log Identifier Name Exchange Complete: NO
```

Figure 78. ppcadmin list xlns command

In Figure 78, the PPC Gateway server: **/.:/cics/ppc/gateway/cicsgwy** has made requests for the XLN process to occur between the local LU `CICSLUGW` and both partner LUs `CICSESA` and `CICSMVS`. The XLN process, which is represented as `Log Identifier Name Exchange Complete`, is successful between `CICSLUGW` and `CICSESA`, but unsuccessful between `CICSLUGW` and `CICSMVS`.

An unsuccessful XLN process can occur as a result of network or operations failures. Check your SNA product's messages for information about network availability. Check your PPC Gateway server's messages for operations problems. For more information about PPC Gateway server messages, refer to "PPC Gateway server problem determination" on page 228. For information about the related **ppcadmin query xln** command, see *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration*.

Requesting the XLN process with a remote SNA system

Use the **ppcadmin force xln** command to request that the XLN process occur between the CICS region and a remote SNA system. (See "Exchange log names (XLN) process" on page 180 for an overview of the XLN process.)

To use this command, you must provide the local LU name for the local CICS region and the partner LU name for the remote system. An example of this command and its output is shown in Figure 79.

```
ppcadmin force xln CICSLUGW CICSMVS

Command executed at: Mon Oct 11 15:39:22 1999
```

Figure 79. ppcadmin force xln command with successful results

In Figure 79, the PPC Gateway server **/.:/cics/ppc/gateway/cicsgwy** has requested that the XLN process occur between local LU `CICSLUGW` and partner LU `CICSMVS`. The request is successful.

If the command fails, as shown in the example in Figure 80 on page 206, check the messages that the PPC Gateway server produces. Refer to "PPC Gateway server problem determination" on page 228 for information about PPC Gateway server messages.

```
ppcadmin force xln CICSLUGW CICSMVS
Error: ENC-ppc-0016: Connection failure, no retry
```

Figure 80. ppcadmin force xln command with unsuccessful results

Viewing pending resynchronizations in the PPC Gateway server

Use the **ppcadmin query resync** command to view the resynchronization requests that the PPC Gateway server requires to resolve the outcome of distributed CICS LUWs. (See “Exchange log names (XLN) process” on page 180 for an overview of the resynchronization process.)

To use this command, you must provide the local LU name for the local CICS region and the partner LU name for the remote system. An example of this command and its output is shown in Figure 81.

```
ppcadmin query resync CICSLUGW CICSMVS

Command executed at: Mon Oct 11 16:15:53 1999


Log Identifier Name Exchange Complete: NO
Number of transactions with pending resynchronizations: 1

Executive LU: CICSLUGW  SNA Partner LU: CICSMVS
        Logical Unit of Work Id: MYSNANET.TMVS412:1c2f3594dea2:0001
        Transaction Id: 2
        Global tid: 00 00 00 02 06 12 01 01 e0 1f 0a 00 b3 f9 a4 1e 8c c0 08
        convId: 20493824
        Conversation Correlator: 60 f9 f8 d3
        Session Id: f0 1c 0f ec 37 51 68 01
        SNA Mode Name: CICSISC0
        Local Tran State: PPC_TRAN_STATE_PREPARED
        Peer Tran State: PPC_TRAN_STATE_PENDING

Transaction is prepared and awaiting resolution at this site. (Not finished).

A prepare has been sent by the peer. The peer has resynchronization
responsibility towards this site.
```

Figure 81. ppcadmin query resync command

In Figure 81, the PPC Gateway server **/.:/cics/ppc/gateway/cicsgwy** is requested to show the resynchronization requests that are pending between local LU `CICSLUGW` and partner LU `CICSMVS`. Currently, only one resynchronization request is required.

For information about the related **ppcadmin list resyncs** command, see *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration*.

Canceling pending resynchronizations in the PPC Gateway server

Use the **ppcadmin cancel resync** command to delete the resynchronization requests that the PPC Gateway server requires to resolve the outcome of distributed CICS LUWs. (See “Exchange log names (XLN) process” on page 180 for an overview of the resynchronization process.)

To use this command, you must provide the local LU name for the local CICS region and the partner LU name for the remote system. An example of this command and its output is shown in Figure 82.

```
ppcadmin cancel resync CICSLUGW CICSMVS

Command executed at: Mon Oct 11 16:15:53 1999
```

Figure 82. ppcadmin cancel resync command

In Figure 82, the PPC Gateway server **/.:/cics/ppc/gateway/cicsgwy** is requested to delete the resynchronization requests that are required between local LU `CICSLUGW` and partner LU `CICSMVS`.

Note: Use this command only if the remote SNA system has been cold started, or the remote system will not contact the PPC Gateway server again. If a transaction in the local CICS region is waiting for the result of any of the deleted resynchronization requests, you must issue a **FORCEPURGE** against it before it will complete. If the transaction is in doubt, it will commit or back out as specified in the **InDoubt** attribute of its Transaction Definitions (TD) entry.

Viewing intersystem requests running in the PPC Gateway server

Use the **ppcadmin list convs** command to view the intersystem requests that are running in a PPC Gateway server. An example of this command and its output is shown in Figure 83.

```
ppcadmin list convs

Command executed at: Mon Oct 11 16:08:24 1999


total convs: 2

convId: 20493824
        Logical Unit of Work Id: MYSNANET.TMVS412:1c2f3594dea2:0001
        Transaction Id: 2
        Global tid: 00 00 00 02 06 12 01 01 e0 1f 0a 00 b3 f9 a4 1e 8c c0 08
        TP Name: ACCT
        Executive LU: CICSLUGW <-- SNA Partner LU: CICSMVS

convId: 204898c4
        Logical Unit of Work Id: MYSNANET.TESA232:1c2bb7ccd158:0002
        Transaction Id: 1
        Global tid: 00 00 00 c6 01 0c 73 6e 61 74 65 73 74 31 00 00 00 08
        TP Name: CEMT
        Executive LU: CICSLUGW <-- SNA Partner LU: CICSESA
```

Figure 83. ppcadmin list convs command

In Figure 83, the PPC Gateway server **/.:/cics/ppc/gateway/cicsgwy** is requested to list the current conversations. Two intersystem requests running. One intersystem request is for the CICS transaction `ACCT` and is between the local LU `CICSLUGW` and the partner LU `CICSMVS`. The other intersystem request is for the CICS transaction `CEMT` and is between the local LU `CICSLUGW` and the partner LU `CICSESA`.

If you need more information about a particular conversation, use the **ppcadmin query conv** command. This command requires the `convId` number for the particular conversation, which you can obtain from the results of the **ppcadmin list**

convs command. An example of the **ppcadmin query conv** command and its output is displayed in Figure 84.

```
ppcadmin query conv 204898c4

Command executed at: Mon Oct 11 17:55:51 1999


convId: 204898c4

Conversation Parameters:
------------------------
Logical Unit of Work Id: MYSNANET.TESA232:1c2bb7ccd158:0002
Transaction Id: 1
Global tid: 00 00 00 c6 01 0c 73 6e 61 74 65 73 74 31 00 00 00 08
TP Name: CEMT
Sync Level: CM_SYNC_POINT
Conversation Type: CM_MAPPED_CONVERSATION
Conversation State: CM_SEND_STATE
Last Sync point State: CM_SEND_STATE

This is an SNA conversation.
        Conversation Correlator: 60 f9 f8 e5
        Session Id: f0 1c 10 19 23 86 ea 02

Peer Information:
-----------------
        Executive LU: CICSLUGW <-- SNA Partner LU: CICSESA
        Local Tran State: PPC_TRAN_STATE_ACTIVE
        Perceived Peer (Tran) State: PPC_TRAN_STATE_ACTIVE

SNA Parameters:
---------------
        SNA Mode Name: CICSISC0
        SNA Security Type: CM_SECURITY_NONE
        SNA User Id:

Conversation Statistics:
-----------------------
        Bytes sent:             933
        Bytes Received:         686
        Error Count:            0
        Number of Sync points:   0
        Number of Backouts:     0
```

Figure 84. ppcadmin query conv command

For information about the related **ppcadmin query stats** command, see *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration*.

Viewing LUWs in the PPC Gateway server

Use the **ppcadmin list luws** command to view the LUWs that are active in a PPC Gateway server. An example of this command and its output is shown in Figure 85 on page 209.

```
ppcadmin list luws

Command executed at: Mon Oct 11 15:56:58 1999


Total LUWs: 2

Logical Unit of Work Id: MYSNANET.TMVS412:1c2f3594dea2:0001
        Transaction Id: 2
        Global tid: 00 00 00 02 06 12 01 01 e0 1f 0a 00 b3 f9 a4 1e 8c c0 08

Logical Unit of Work Id: MYSNANET.TESA232:1c2bb7ccd158:0002
        Transaction Id: 1
        Global tid: 00 00 00 c6 01 0c 73 6e 61 74 65 73 74 31 00 00 00 08
```

Figure 85. ppcadmin list luws command

In Figure 85, the command is issued to the PPC Gateway server **/.:/cics/ppc/gateway/cicsgwy**. This PPC Gateway server has two LUWs. For each LUW, the **ppcadmin list luws** command displays the following:

- The `Logical Unit of Work Id` (or *LUWId*): the SNA unique name for the LUW.
- The `Transaction Id` (or *tid*): the local identifier of the LUW that the PPC Gateway server uses.
- The `Global tid`: the identifier for the LUW that all involved servers use, such as a Structured File Server (SFS) server.

If you need more information about a particular LUW, use the **ppcadmin query luw** command. This command requires the `Logical Unit of Work Id` value for the particular LUW, which you can obtain from the results of the **ppcadmin list luws** command. An example of the **ppcadmin query luw** command and its output is shown in Figure 86.

```
ppcadmin query luw MYSNANET.TESA232:1c2bb7ccd158:0002

Command executed at: Mon Oct 11 15:57:39 1999


Logical Unit of Work Id: MYSNANET.TESA232:1c2bb7ccd158:0002
Total transactions: 1

Transaction Id: 1
Global tid: 00 00 00 c6 01 0c 73 6e 61 74 65 73 74 31 00 00 00 08
Number of conversations associated with this transaction: 1

        convId: 204898c4
        Executive LU: CICSLUGW
        SNA Partner LU: CICSESA
        Conversation Correlator: 60 f9 f8 e5
        Session Id: f0 1c 10 19 23 86 ea 02
        Local Tran State: PPC_TRAN_STATE_ACTIVE
        Peer Tran State: PPC_TRAN_STATE_ACTIVE

Transaction is currently active at this site. (Not yet resolved).

Transaction is currently active at the peer. (Not yet resolved).
```

Figure 86. ppcadmin query luw command

For information about the related **ppcadmin list transactions** and **ppcadmin query transaction** commands, see *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration.*

Changing CICS intercommunication definitions for use with a PPC Gateway server

If you need to change the routing of the SNA flow, you possibly need to add, change, or delete CICS intercommunication resource definitions such as:

- Listener Definitions (LD) entries
- Gateway Definitions (GD) entries
- Communications Definitions (CD) entries

This section includes information about when and how these resource definitions can be updated, and the effects of updating them on the SNA communications product and the PPC Gateway server.

CICS resource definitions exist in two places:

- In the permanent database that contains the CICS resource definitions that are available to the region during a cold start.
- In the runtime database that contains the CICS resource definitions that are available to the region during an autostart. The runtime database is created as part of a region cold start. It can be changed only when the region is running.

Use the **cicsadd** command to add intercommunication resource definitions to the permanent database, and, if the region is running, to the runtime database. For example, the following command adds a Listener Definitions (LD) entry called `LOCALSNA` to a region. The **-B** option causes the entry to be added to the permanent database, and, if the region is running, to the runtime database:

```
cicsadd -c ld -r regionName -B LOCALSNA Protocol=SNA
```

Use the **cicsinstall** command to add, to a running region, resources that are defined in the permanent database. For example, the following command adds to your region all resources that have the attribute **ActivateOnStartup** equal to **yes**:

```
cicsinstall -r regionName -a
```

Use the **cicsdelete** command to delete resources that are defined in the permanent database, and sometimes, in a running region. For example, the following command deletes a Listener Definitions (LD) entry called `LOCALSNA` from a region's permanent database only (indicated by the **-P** option):

```
cicsdelete -c ld -r regionName -P LOCALSNA
```

The following table summarizes the effects of changing CICS intercommunication resource definitions on the region's permanent and runtime databases.

Table 38. The effect of updating CICS intercommunication resource definitions

Action	Listener Definitions (LD) entry	Gateway Definitions (GD) entry	Communications Definitions (CD) entry
Add new entry to permanent database.	This can be done at any time. The entry is then available at the next cold start.	Same as the LD entry.	Same as the LD entry.

Table 38. The effect of updating CICS intercommunication resource definitions (continued)

Action	Listener Definitions (LD) entry	Gateway Definitions (GD) entry	Communications Definitions (CD) entry
Add new entry to runtime database while region is running.	The region accepts a new LD entry, but it has no effect. You must restart the region before it uses the new LD entry.	When the GD entry is added to the region, CICS immediately configures the PPC Gateway server that is named in the GD entry so that it is available for use by the region.	The CD entry is immediately available for use by CICS transactions.
Update existing entry in permanent database.	This can be done at any time. The updated entry is then available at the next cold start.	Same as the LD entry.	Same as the LD entry.
Update existing entry in runtime database while region is running.	This is not allowed.	The entry must be deleted by using the **cicsdelete -R** command, then updated.	The entry must be deleted by using the **cicsdelete -R** command, then updated.
Delete existing entry from permanent database.	This can be done at any time. The entry will no longer be available at the next cold start.	Same as the LD entry.	Same as the LD entry.
Delete existing entry from runtime database while region is running.	The LD entry cannot be deleted from the runtime database.	The GD entry can be deleted at any time. Before deleting the entry, CICS removes its configuration from the PPC Gateway server.	The CD entry can be deleted from the region only if a CICS transaction is not using it for an intersystem request.

Transaction Definitions (TD) entries also contain intercommunication information in the **TPNSNAProfile**, **SNAModeName**, and **IsBackEndDTP** attributes. These attributes affect the way that intersystem requests are processed that involve the transaction that is named in the TD entry. In addition, TD entries that have the **TPNSNAProfile** attribute configured can affect the PPC Gateway servers that are configured in the Gateway Definitions (GD) and the CICS local SNA support.

If you add a GD entry to your region while it is running, CICS attempts to contact the PPC Gateway server that is configured in the GD entry in order to pass it information about the local system. If the PPC Gateway server is not running, CICS cannot contact it. If this occurs, start the PPC Gateway server, then run the CGWY transaction. This is described in “Sharing server names between a CICS region and a PPC Gateway server” on page 177.

If you delete a GD entry from your region, you *must* ensure that all information about your region has been removed from the PPC Gateway server. You can do this by:

- Using the command:

```
ppcadmin delete luentry -server server_name LUentry
```

where *server_name* is the name of the PPC Gateway server (for example, **/.:/cics/ppc/gateway/cicsgwy**) and *LUentry* is the local LU name taht is configured in the **GatewayLUName** attribute of the deleted GD entry. Refer to *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration* for further information.

- Stopping, then cold starting the PPC Gateway server.
- Stopping and destroying the PPC Gateway server.

Similarly, if you change the local LU name that is configured in the GD entry attribute **GatewayLUName**, you *must* ensure that the information that associates the original LU name with your region is removed from the PPC Gateway server. You can do this by stopping and cold starting the PPC Gateway server, or by issuing the **ppcadmin delete luentry** command. After the original information is removed from the PPC Gateway server, you can run the CGWY transaction to add new information. Refer to "Sharing server names between a CICS region and a PPC Gateway server" on page 177 for further information.

When you add, update, or delete a Transaction Definitions (TD) entry that has a non-blank **TPNSNAProfile** attribute in the running region, the region passes information about the changes to every PPC Gateway server that is configured in the GD. If a PPC Gateway server is not running, it does not receive the updates. This is a concern only if you are adding a TD entry and the PPC Gateway server is running on AIX. Until this PPC Gateway server is given the information about the new transaction, it cannot receive intersystem requests for the transaction from remote SNA systems. If this condition occurs, start the PPC Gateway server and perform one of the following actions to pass the configuration to the PPC Gateway server:

- Run the CGWY transaction. This action configures all PPC Gateway servers that are defined in the GD, with details of the local region and all transactions that have a TD entry where the **TPNSNAProfile** attribute is configured.
- Delete, then add the TD entry for the transaction. This action configures all PPC Gateway servers with the transaction.
- Delete, then add the GD entry for the PPC Gateway server. This action configures the PPC Gateway server with details of the local region and all transactions that have a TD entry where the **TPNSNAProfile** attribute is configured.

Chapter 9. Intersystem problem determination

This chapter contains information about problem determination in an intercommunication environment. It is provided for CICS users who are new to SNA. The information in this chapter is not intended to replace the information that is provided in the administration and diagnosis books that are available with the SNA products that you are using. Refer to *TXSeries for Multiplatforms Concepts and Planning* for a list of relevant SNA books.

This chapter describes:
- The steps to take to track down and solve an intersystem problem
- Common intercommunication problems
- CICS intercommunication error codes that appear in CICS messages
- PPC Gateway server problem determination procedures

Intersystem problem solving process

Problem solving in the CICS intersystem environment might seem a difficult task because of the number and variety of products that are involved. It requires a systematic approach that enables you to find the component in the network that is causing the problem. It is also helpful if you have:
- Knowledge of what each product does
- Familiarity with the diagnostics
- Experience to recognize the symptoms of common problems

The information that follows guides you through the process of intersystem problem determination and shows you where to look for clues and how to interpret those clues. It is sometimes difficult in this description to be specific because each problem is different. However, by following the process that is suggested, you will be able to isolate your particular problem to one component in the network and most likely fix it yourself. If you cannot solve the problem, the information that you have collected will help the service representative to solve the problem.

To solve intersystem problems, do the following steps:
- “Step 1. Understand the failure scenario”
- “Step 2. Identify the failing component” on page 214
- “Step 3. Fix the problem” on page 215

Step 1. Understand the failure scenario

The first step in problem determination is to have a clear picture of exactly what is failing. Here are some questions to ask yourself:
- What was the system doing when the failure occurred? For example, was it initializing or had it been running successfully for some time?
- What transactions were running at the time of the failure? What should these transactions do?
- When does it fail? Does it fail all the time, or is it intermittent?
- How does it fail? Is it a hang, loop, or abend?
- Did it start failing after a particular event such as a change to the configuration or a system crash?
- What steps are needed to re-create the problem?

If possible, try a little experimentation to either narrow the failure down to a specific scenario, or to discover the extent of the problem. (The CICS-supplied CECI and CRTE transactions might be useful.)

This experimentation process is best demonstrated with an example. Consider the problem in which a function shipped temporary storage (TS) queue request that is issued by a CICS transaction fails. You should try a few requests to see if all function shipped TS queue requests fail or if it is only the one that was issued by the failing transaction. If all function shipped TS queue requests fail, do function shipping requests to files work, for example? If function shipping in general fails, is transaction routing working or do you have a scenario in which all outbound requests are failing? If all outbound requests are failing, do inbound requests work?

By answering these questions you are generating a clear picture of the problem that you are trying to solve, an important first step to identifying the cause.

Step 2. Identify the failing component

When you are sure of the exact nature and extent of the problem that you are trying to solve, you must identify which component in the network is failing. You might have strong suspicions about this already, but if not, or you want to confirm your suspicions, continue as follows:

1. Identify the components of the network that are involved in the request.

   For example, the request might flow from your CICS region, across TCP/IP to a PPC gateway, then from the PPC gateway through an SNA network to arrive in a mainframe CICS/ESA system where it is to be run.
2. Check whether each component is running. If a component is running, check whether it has produced any error or attention messages. (Refer to Table 39 for some suggestions about where to look for diagnostics.)

Table 39. Sources of information for following the path of a request

Type of request	Local CICS messages (console.msg and CSMT.out)	PPC Gateway server messages	SNA link trace	Remote system's message
TCP/IP intersystem request	Yes			Yes
PPC Gateway server configuration	Yes	Yes		
PPC Gateway server/SNA intersystem request	Yes	Yes	Yes	Yes
Local SNA intersystem request	Yes		Yes	Yes

Note: CICS messages are described in *TXSeries for Multiplatforms Messages and Codes* . They are very useful in intersystem PD because:

- They log information on the failing request.

- During region startup, messages are logged that describe the network name and protocols that the region uses and any errors that are detected in the communication configuration (such as LD, CD and GD entries).

PPC Gateway server messages are described in “PPC Gateway server message descriptions” on page 232.

It is possible that several components have reported the error. However, it is usually the component that initially detects the error that gives the most precise diagnostics. Refer to the message descriptions because they might give you the exact cause of the error.

If the messages do not identify the problem, try turning on the trace in each component and rerunning the failing request. Traces show the calls and responses that are being passed between the different components that might highlight a bad parameter or return code. They tend to be designed for developers of the product and so are not easy to read without the product design information. However, the trace will tell you whether the request even reached a particular component. If you find that the request failed before reaching the remote system, concentrate your suspicions around the component that rejected the request. Check the configuration of this component. Also refer to the information that is given in “Common intercommunication errors” on page 216 because it might describe the failure that you are seeing.

If you still cannot determine what is wrong, refer to “Getting further help” because you need help from the service representative.

Step 3. Fix the problem

The message descriptions and other diagnostics should provide you with the information that you need in order to solve the problem.

When the problem is fixed, think about whether it could happen again, and whether you can take any steps to prevent it. This can save you time and trouble in the future.

Getting further help

If you really cannot solve the problem yourself, collect the appropriate information, as shown in Table 40, and contact the service representative.

Table 40. Required information for the service representative

Information Required	TCP/IP request	Local SNA request	PPC Gateway server request	PPC Gateway server configuration
Description of the problem	Yes	Yes	Yes	Yes
CICS messages, trace and resource definitions	Yes	Yes	Yes	Yes
PPC Gateway server messages			Yes	Yes
PPC Gateway server GSD entry			Yes	Yes
PPC Gateway server trace			Yes	Yes
PPC Gateway server configuration			Yes	Yes
SNA product configuration, messages and link trace		Yes	Yes	Yes

Table 40. Required information for the service representative (continued)

Information Required	TCP/IP request	Local SNA request	PPC Gateway server request	PPC Gateway server configuration
Messages and other diagnostics on the remote system	Yes	Yes	Yes	

Common intercommunication errors

This section describes how to solve common intercommunication problems for the following symptoms:

- "Symptom: cannot start (acquire) SNA sessions"
- "Symptom: the PPC Gateway server fails to start" on page 217
- "Symptom: CICS will not configure the PPC Gateway server" on page 217
- "Symptom: CICS will not configure my transactions in the PPC Gateway server" on page 217
- "Symptom: CICS cannot configure the PPC Gateway server" on page 218
- "Symptom: CICS for MVS/ESA transactions abend AXFX" on page 219
- "Symptom: SNA exchange log names (XLN) fails" on page 219
- "Symptom: all inbound requests fail" on page 220
- "Symptom: all outbound requests fail" on page 220
- "Symptom: invalid or unrecognizable data is passed to applications" on page 221
- "Symptom: applications fail because of security violations" on page 221
- "Symptom: Transaction routing to CICS Transaction Server for z/OS fail with communications error 15a00007/a0000100" on page 221
- "Symptom: Transaction routing causes screen errors" on page 222
- "Symptom: CICS fails to detect a predefined CD entry, and autoinstalls another" on page 222

Symptom: cannot start (acquire) SNA sessions

These problems are usually because of:

- A configuration problem
- The remote system, or part of the network, being unavailable

Check whether all the components of your network, including the remote system, are running. Check whether the configuration in these components is properly installed.

If the link will not start and the relevant machines are running, either your local machine, VTAM, or the remote machine does not have correct node, or machine address information.

If the link is running but the session will not start, experiment to see whether the session can be activated from one side only. For example, the session might not start if the request is issued locally, but will start if activated from the remote system. If this occurs, the local and partner Logical Unit (LU) names are correctly configured but either,

- The modename definition that is used does not allow both systems to bind contention winners.

- The side that cannot start any sessions cannot locate the remote system. This could be because the remote system is defined as located on the wrong machine, or using the wrong link.

If the session will not start in either direction, check:

- The local and partner LUs are correctly defined on all systems.

  It is often easy to misspell LU names; for example, replacing a numeric 0 (zero) with the letter O, or the number one (1) with the letter l.
- If VTAM is used, the PU and LU definitions that are used are correct.
- The modenames that are used are consistent both in the local definitions and in the remote system's definitions. In addition, if VTAM is used, **MODEENT** definitions are required for each modename.

Refer to the specific SNA product information for details of local configuration and how to display the status of your SNA product.

Symptom: the PPC Gateway server fails to start

The PPC Gateway server is started using the **cicsppcgwy** command. If this fails to work, check the message descriptions that this command produces. Also check the PPC Gateway server's message file, because the PPC Gateway server might have started, detected a bad condition, then exited. More information about PPC Gateway server messages is given in "PPC Gateway server message descriptions" on page 232.

If RPC-only is being used, check the server_bindings file CICS_HOSTS environment variable.

Symptom: CICS will not configure the PPC Gateway server

Your CICS region requires a Gateway Definitions (GD) entry for your PPC Gateway server in order to pass it configuration information. Check whether you have defined a GD entry for the PPC Gateway server and this entry is installed in your CICS region. If your GD entry is correct, refer to "Symptom: CICS cannot configure the PPC Gateway server" on page 218.

Symptom: CICS will not configure my transactions in the PPC Gateway server

When configuring a PPC Gateway server, your CICS region attempts to pass the names of all CICS transactions that have a value configured in the **TPNSNAProfile** attribute of their Transaction Definitions (TD) entry. Only the AIX PPC Gateway Server requires this information because it cannot receive an intersystem request from Communications Server for AIX unless:

- The TD entry for the CICS transaction has the **TPNSNAProfile** attribute set to the name of a valid AIX TPN profile.
- This profile is in Communications Server for AIX's runtime database.
- CICS has successfully configured the AIX PPC Gateway Server with the name of the CICS transaction.

You can view the CICS transactions that are configured in the PPC Gateway server by using **ppcadmin**. If your transaction is not configured in the PPC Gateway server, check whether:

- The **TPNSNAProfile** attribute in the TD entries is set up correctly.

- Your CICS region can successfully configure the PPC Gateway server. (For more information, refer to "Symptom: CICS cannot configure the PPC Gateway server.")

Note: When using a non-AIX PPC Gateway Server, it is usual to leave the **TPNSNAProfile** attributes set to their default value of "". However, if the **TPNSNAProfile** attributes are changed and CICS passes some transaction names to a non-AIX PPC Gateway Server, it responds to CICS by indicating that it does not need this information, and CICS stops the configuration. Therefore, if you do configure the **TPNSNAProfile** attribute in your TD entries, CICS will make one unnecessary configuration request to the non-AIX gateway each time it is configured.

Symptom: CICS cannot configure the PPC Gateway server

The CICS-supplied transaction CGWY configures the PPC Gateway servers at CICS region startup, or you can run it by using the **EXEC CICS START TRANSID(CGWY)** command. To configure the PPC Gateway servers, CGWY needs:

- A Transaction Definitions (TD) entry for CGWY that allows it to run as a background task
- A Gateway Definitions (GD) entry for each PPC Gateway server that it is to configure

Check whether these conditions are satisfied.

Your CICS region also configures a PPC Gateway server when a GD entry for that PPC Gateway server is successfully installed in the region.

When the PPC Gateway server configuration is in progress:

- The PPC Gateway server must be running.
- Your SNA product must be correctly configured.

CICS writes messages to the **console.***nnnnnn* file while the PPC Gateway server configuration is in progress. These messages show the GD entries that are being used for the configuration. In the example below, the CICS region cics6000 has two GD entries, one named GWY and the other named GWY2.

```
ERZ030060I/3101 date time region : \
PPC Gateway server configuration has started
ERZ030062I/3103 date time region : \
Configuring PPC Gateway server 'GWY' with details of the region
ERZ030063I/3104 date time region : \
Configuring PPC Gateway server 'GWY' with details of local transactions
ERZ030064I/3109 date time region : \
Configuring PPC Gateway server 'GWY' has ended
ERZ030062I/3103 date time region : \
Configuring PPC Gateway server 'GWY2' with details of the region
ERZ030063I/3104 date time region : \
Configuring PPC Gateway server 'GWY2' with details of local transactions
ERZ030064I/3109 date time region : \
Configuring PPC Gateway server 'GWY2' has ended
ERZ030061I/3102 date time region : \
PPC Gateway server configuration has ended
```

If CICS detects errors when configuring a PPC Gateway server, it writes an error message that describes the problem to **console.***nnnnnn*, and moves to the next GD entry. Examine these messages in your region's *nnnnnn* file. They are described in the *TXSeries for Multiplatforms Messages and Codes* manual. If error messages occur, follow the instructions that are given in the message descriptions. If no error messages occur, check whether the GD entries contain the correct values for the PPC Gateway server. Also check whether the required GD entries are installed in your region. Finally, check the PPC Gateway server's message file, because the PPC Gateway server may have reported the reason for the error.

Symptom: CICS for MVS/ESA transactions abend AXFX

The AXFX abend code usually occurs if a CICS for MVS/ESA transaction makes a synchronization level 2 request to CICS for Open Systems on a connection that has not successfully exchanged log names (XLN).

XLN can be successful only on a SNA connection that uses a PPC Gateway Server. So check whether the CICS for Open Systems Communications Definition (CD) entry for the connection is set up with **ConnectionType=ppc_gateway**.

If the SNA connection is using a PPC Gateway Server, refer to "Symptom: SNA exchange log names (XLN) fails" because this explains how to make the XLN work.

If the connection should use local SNA (**ConnectionType=local_sna**) and, therefore, you want CICS for MVS/ESA to use synchronization level 1 intersystem requests, change the local LU definition in your SNA product so it requests synchronization level 1 instead of synchronization level 2. For information about synchronization levels, refer to "Ensuring data integrity with synchronization support" on page 16. Alternatively:

- "Communicating across SNA connections" on page 8 compares using local SNA with using a PPC Gateway Server.
- Chapter 4, "Configuring CICS for SNA," on page 67 describes how to configure CICS to use local SNA or a PPC Gateway Server.

Symptom: SNA exchange log names (XLN) fails

SNA Synchronization level 2 intersystem requests can be sent only to a remote system that has successfully completed exchange log names (XLN). XLN occurs between the remote SNA system and a PPC Gateway Server. Ensure that the remote system, SNA session, and PPC Gateway Server are running. Also check whether the remote system can support synchronization level 2 requests.

Use the **ppcadmin list xln** command to list the XLN status of your SNA connections. (This is described in "Viewing XLN process status in the PPC Gateway server" on page 204.)

If XLN is not successful for a connection, check the PPC Gateway Server's message file because the PPC Gateway Server might have logged an error message that indicates why the XLN failed. Follow the instructions that are given in "PPC Gateway server message descriptions" on page 232 for any error messages that you find.

If the **ppcadmin list xln** command does not list a particular SNA connection, XLN has never been attempted for the connection since the PPC Gateway Server was last cold started. Use the **ppcadmin list luentries** command, which is described in "Viewing CICS configuration in the PPC Gateway server" on page 203, to show which CICS regions have configured the PPC Gateway Server. Follow the instructions that are given in “Symptom: CICS will not configure the PPC Gateway server” on page 217 if your CICS region has not configured the PPC Gateway Server.

When you have ensured that the PPC Gateway Server has been configured by your CICS region, use the **ppcadmin force xln** command, which is described in “Requesting the XLN process with a remote SNA system” on page 205, to issue an XLN request. If this command fails, check the PPC Gateway Server’s message file because the PPC Gateway Server will have logged an error message that indicates why the XLN failed. Follow the instructions for these messages, which are given in “PPC Gateway server message descriptions” on page 232.

For information about synchronization levels, refer to “Ensuring data integrity with synchronization support” on page 16. Alternatively:

- “Mixing the communications methods” on page 11 compares using local SNA with using a PPC Gateway Server.
- Chapter 4, “Configuring CICS for SNA,” on page 67 describes how to configure CICS to use local SNA or a PPC Gateway Server.
- “Overview of the PPC Gateway server” on page 175 describes how the PPC Gateway Server works.

Symptom: all inbound requests fail

Check whether your CICS region is available and configured correctly. Examine the CICS messages that are written to your region’s **console.***nnnnnn* and **CSMT.out** because they might explain what the problem is.

Check whether the network and the remote system are available and configured correctly. If you are using a PPC Gateway server and SNA synchronization level 2, check whether exchange log names has been successful. (Refer to “Symptom: SNA exchange log names (XLN) fails” on page 219.)

If you are using a PPC Gateway server, check whether the PPC Gateway server is configured correctly. Refer to “Symptom: CICS cannot configure the PPC Gateway server” on page 218 for more information.

Look for error messages that are written by the remote system, because the remote system might have rejected the request before it was sent into the network.

If you are using Communications Server for AIX, ensure that the **TPNSNAProfile** attribute for all your Transaction Definitions (TD) entries are set to the name of an Communications Server for AIX TPN profile, and, if you are using an AIX PPC Gateway Server, that CICS has configured these transactions in the PPC Gateway server. (Refer to “Symptom: CICS will not configure my transactions in the PPC Gateway server” on page 217.)

Symptom: all outbound requests fail

Examine the CICS messages that are written to your region’s **console.***nnnnnn* and **CSMT.out** because they might explain what the problem is.

Check whether the network and the remote system are available and configured correctly.

If you are using a PPC Gateway server, check whether the PPC Gateway server is configured correctly and running. The PPC Gateway server is configured in your

CICS region using a Gateway Definitions (GD) entry. (Refer to “Symptom: CICS cannot configure the PPC Gateway server” on page 218 for more information.) If the PPC Gateway server is successfully configured, check whether the PPC Gateway server has written any error messages. More information about PPC Gateway server messages is given in “PPC Gateway server message descriptions” on page 232.

The Communications Definitions (CD) entry for your connection should point to this GD entry by the **GatewayName** attribute.

If you are using SNA synchronization level 2, check whether exchange log names has been successful. (Refer to “Symptom: SNA exchange log names (XLN) fails” on page 219.)

Look for error messages that are written by the remote system, because the remote system might have rejected the request.

Symptom: invalid or unrecognizable data is passed to applications

If your applications are starting correctly, but the data that they are given is not what was sent (for example, the COMMAREA that is passed to a DPL program has invalid values), it could be that:

1. The data conversion templates are not installed correctly.
2. The **TemplateDefined** attribute of your local resource (for example, a program or file) is not set to **yes**.

See “Summary of data conversion” on page 169 and Chapter 7, “Data conversion,” on page 147.

Symptom: applications fail because of security violations

If intersystem requests fail for security reasons, your security configuration might be incorrect. Access to resources for intersystem requests are controlled by static configuration. When configured, the security checking operates automatically. Try to establish which part of the network is rejecting the request. For example:

- Is the request failing before it leaves the originating system? TXSeries for Multiplatforms returns **NOTAUTH** to an application if it attempts to use the **SYSID** option on a function shipping request when **RSLCheck=internal** in the application’s Transaction Definitions (TD) entry.
- Check whether any user IDs that are sent with CICS intersystem requests are recognized by the remote system. This includes the CICS default user ID, which is configured in the RD attribute **DefaultUserId**, that is used by CICS private transactions.

Refer to Chapter 6, “Configuring intersystem security,” on page 123 and “Common configuration problems with intersystem security” on page 144 for more information about security configuration and problems.

Symptom: Transaction routing to CICS Transaction Server for z/OS fail with communications error 15a00007/a0000100

If your region is communicating with CICS Transaction Server for z/OS version 4.1 or later, and all transaction routing requests fail with a communications error 15a00007/a0000100, this could be because of a mismatch in the security configuration in the two systems.

If the CONNECTION definition on CICS Transaction Server for z/OS has ATTACHSEC set to IDENTIFY or VERIFY, and USEDFLTUSER set to NO, CICS Transaction Server for z/OS requires a user ID to be sent with all transaction routing requests. If a user ID is not received, CICS Transaction Server for z/OS unbinds the session that is used by the transaction routing request with a sensecode of X’080F6051’. Your region is informed of a conversation failure and reports communications error 15a00007/a0000100.

To solve this problem you can either:

- Set **OutboundUserIds=sent** in the Communications Definition (CD) entry for the remote system so that your region sends user IDs to CICS Transaction Server for z/OS, or
- Change the settings of ATTACHSEC or USEDFLTUSER in the CICS Transaction Server for z/OS CONNECTION definition so that it does not require your region to send user IDs.

Symptom: Transaction routing causes screen errors

Problems with the data that is displayed at a terminal by a transaction routed transaction can be caused by:

- Incorrect code page defined in the **RemoteCodePageTR** attribute of the Communications Definitions (CD) entry. Refer to “Data conversion for transaction routing” on page 164 for information about how to code this attribute.
- Incompatible terminal definitions in the local or remote regions. Check whether the terminal definitions that are used by each system are compatible. TXSeries for Multiplatforms uses a Terminal Definitions (WD) entry to define a terminal.
- Errors in the application program. Check whether the application runs with a local terminal.

Symptom: CICS fails to detect a predefined CD entry, and autoinstalls another

If you have previously defined a Communications Definitions (CD) entry for a remote system that is connected over CICS family TCP/IP, CICS might fail to use it when a connection request is received, and it might autoinstall another CD entry.

When CICS receives a connection request from a remote CICS system, it searches the CD entries in your region to determine whether one already exists for that connection.

The CD attributes that it checks are:

- **ConnectionType**
- **RemoteLUName**
- **RemoteTCPAddress**
- **RemoteTCPPort**
- **ListenerName**

Also, the CD entry must not be in use by another connection.

If you have predefined a CD entry for the remote system, but CICS does not use it and proceeds to build an autoinstall entry, check the following:

- Verify that the attributes that are listed above have been defined correctly.
- Check whether both systems are trying to acquire the connection simultaneously.

When your local CICS region attempts to acquire a connection to a remote system, it will use its CD entry for that connection. If the remote system attempts to acquire the connection at the same time, your local CICS region will not be able to use that same CD entry when it receives the connection request, because that entry will be in use. Therefore your CICS region will autoinstall another CD entry.

- Check the definition of the connection in the remote system, and ensure that when your local region sends a connection request, the remote system selects the correct definition.

  If the remote system is a CICS OS/2 system, it should issue an *EXEC CICS INQUIRE CONNECTION( ) NETNAME( )* command before your local CICS region issues its connection request. This command ensures that the correct LU Name is set into its definition of the connection, and so it will use that definition when it receives the connection request from your CICS region.

SNA product-specific errors

For SNA product-specific errors, you should refer to the appropriate SNA product book:

- *TXSeries for Multiplatforms Using IBM Communications Server for AIX with CICS*
- *TXSeries for Multiplatforms Using IBM Communications Server for Windows Systems with CICS*
- *TXSeries for Multiplatforms Using Microsoft SNA Server with CICS*
- *TXSeries for Multiplatforms Using SNAP-IX for Solaris with CICS*
- *TXSeries for Multiplatforms Using HP-UX SNAplus2 with CICS*

CICS error codes

Some CICS error messages contain:

```
Communications error primaryCode/secondaryCode
```

where *primaryCode* is the primary error code, and where *secondaryCode* is the secondary error code. The following list shows the primary and secondary error codes. Refer to this list for problem determination when you get a CICS communications error message that contains these codes:

15a0000a/15a0010a Unsupported

Explanation: An attempt has been made to use a communications function that is not supported by the communications protocol.

System action: The communications conversation might fail.

User response: Look for additional messages in the console.*nnnnnn* file and the CSMT log. These might describe the cause of the problem, and tell you how to proceed.

If the problem remains, contact your service representative.

15a00001/8890000 Program Error Purging

Explanation: The remote program called EXEC CICS ISSUE ERROR.

System action: The transaction might be abnormally terminated.

User response: If you are using distributed transaction processing (DTP), check whether the front-end and back-end transactions that are involved in the communications conversation are behaving as expected. Refer to "Communicating errors across a conversation" on page 295 for more information. If the indicated transaction is a back-end DTP transaction, ensure that the transaction definition **IsBackEndDTP** is set to **yes**. If the problem remains, contact your service representative.

15a00002/15a00101 Allocate Busy

Explanation: CICS attempted to allocate a conversation, but no bound contention winner sessions were available.

System action: The communications conversation has not been started.

User response: Query your SNA product to determine whether any sessions are active to the remote system. If no sessions are active, start some. This might require restarting the remote system, or part of the network. If all available sessions are bound, but they are in use, considering increasing the number of sessions. This is configured in the mode (or session) definition both in your local SNA product and in the remote system. These ″session limits″ must be the consistent in both system for the connection to activate.

When sessions have been made available, rerun the intersystem request.

15a00002/15a00102 Allocate Failure

Explanation: CICS was unable to initiate an intersystem request with the remote system because either

- The remote system or intervening network is unavailable
- The communications configuration in the Communications Definitions (CD) entry for the connection is incorrect
- If the remote system is connected by SNA, the SNA configuration might be incorrect

System action: The intersystem request fails. Further messages that give more detail might follow.

User response: Check the Communications Definitions (CD) entry for the connection on the local and the equivalent definitions on the remote system. Check whether the remote system and the network path to the remote system are available and correctly configured.

If CICS Bind time security is being used, ensure that no mismatch exists in the bind passwords that are used by each system. Refer to “Authenticating systems across SNA connections” on page 125 for further information about Bind time security.

15a00003/15a00109 Allocate Timed Out

Explanation: The communications operation did not complete in the acceptable time. The timeout value is configured in the **AllocateTimeout** attribute in the Communications Definitions (CD) entry.

System action: The intersystem request has not been started, or has been terminated.

User response: Check whether the remote system is available and the connection between the local system and the remote system is working if the connection is using a PPC Gateway server.

Check the value that is specified in the CD entry for the AllocateTimeout attribute, or the Timeout on allocate if you are using the IBM TXSeries Administration Tool (on Windows systems), or Timeout on allocate (in seconds) field if you are using AIX SMIT. The configuration of the CD entries is described in:

- “Configuring CD entries for CICS PPC TCP/IP” on page 47
- “Using SMIT to configure CD entries (AIX only)” on page 52
- “Configuring CICS for PPC Gateway server SNA support” on page 89

You can change the configured value, which is described in:

- “Changing the attributes of a PPC Gateway server” on page 188

15a00004/15a0010d Synclevel Not Supported Locally

Explanation: The local system cannot support the requested synchronization level. This is usually because the transaction has requested a higher synchronization level than is supported on the connection. The type of connection is configured in the **ConnectionType** attribute of the CD entry that is specified in the SYSID option for the intersystem request. “Designing your network configuration” on page 5 describes the different types of connections and the synchronization levels that they support. “Ensuring data integrity with synchronization support” on page 16 describes the synchronization levels, and their uses in intersystem requests.

System action: The communications conversation has not been started, or has been terminated.

User response: Examine the CD entry that the request uses. Check whether the **ConnectionType** is correct. Check the type of intersystem request that the application is making, and that the synchronization level that is requested is supported on the connection. If using distributed transaction processing (DTP), check the SYNCLEVEL option that is used on CONNECT PROCESS. Refer to the *TXSeries for Multiplatforms Application Programming Reference* for more information. Make any necessary changes to the CD entry or the application, and rerun the request.

15a00004/15a00103 Invalid Convid

Explanation: An invalid conversation identifier has been used on a call to an intersystem communications function.

System action: The communications function is not processed.

User response: If using distributed transaction processing (DTP), check whether the CONVID option on EXEC CICS calls is set correctly. If the problem remains, contact your service representative.

15a00004/15a00104 Invalid Mode/Partner LU alias

Explanation: An intersystem request was unsuccessful because either the modename, or the remote system name is not correctly configured in SNA.

The modename can be specified in:

- The PROFILE option of an EXEC CICS ALLOCATE command
- The **SNAModeName** attribute of the Transaction Definitions (TD) entry for the transaction that is issuing the intersystem request
- The **DefaultSNAModeName** attribute of the Communications Definitions (CD) entry
- The local SNA product

Refer to “Default modenames” on page 110 for more information about SNA modenames.

The remote system name is specified by using the **RemoteLUName**, **RemoteNetworkName**, and **SNAConnectName** attributes of the Communications Definitions (CD) entry that the intersystem request uses. Those attributes must be consistent with the configuration in both your local SNA product and the remote system.

System action: The intersystem request is unsuccessful.

User response: Examine the CICS message logs and the definitions that are used by the issuing transaction, to determine which remote system and modename is requested. Ensure that the configuration in CICS is consistent with the configuration that is in your local SNA product and remote system. Rerun the request after you have corrected the error.

15a00004/15a00105 Invalid Parameter

Explanation: An unrecognized parameter, such as a local or partner LU name, has been used on an intersystem request. If the intersystem request was received from a remote system, the error might be that the remote system is not correctly defined in the Communications Definitions (CD). If the intersystem request is issued by a local transaction, the error might be that the application has requested invalid options. Possible reasons for this are:

- The local LU name that is used on the request is not correctly defined to your SNA product.
- The local LU name is being used by more than one PPC Gateway server or CICS region.
- The remote system names that are configured in the **RemoteLUName** and **SNAConnectName** attributes of the CD entry are incorrect or are not configured in your SNA product.
- The application has requested an invalid option.

If the CD entry for the remote system is correct and a valid intersystem request is being issued, this error could be caused by incompatible versions of the software that your region is using.

System action: The parameter is ignored and the intersystem request might abnormally terminate.

User response: Examine the console.*nnnnnn* file and the CSMT log for additional messages that might describe the cause of this error. If additional messages are produced, follow the instructions for these messages. Also look for messages from your SNA product or PPC Gateway server if you are using one. If no additional messages are produced, check whether the levels of software that are used by your region are compatible. In addition, if you are using a SNA product, a PPC Gateway server, or both, ensure that they are also compatible with your CICS region.

15a00004/15a00110 Exchange log name failed

Explanation: The PPC Gateway server attempted to set up a synchronization level 2 conversation with a remote system across SNA, but was unable to exchange log names with it.

System action: The communication conversation has not been started.

User response: See “PPC Gateway server message descriptions” on page 232 for a description of how to find the PPC Gateway server message file, and how to use it to solve the exchange log name problem. Search the message file for messages about log name exchange for the local and remote systems, and follow the instructions for the message. When the exchange has completed successfully, retry the request.

15a00005/15a00108 State Error

Explanation: The communications verb is not allowed in this particular conversation state.

System action: The communications conversation is abnormally terminated.

User response: If using distributed transaction processing (DTP) check whether the front-end and back-end transactions that are involved in the communications conversation are correct. Refer to “The state numbers” on page 331 for more information. If this problem remains, contact your service representative.

15a00005/15a00111 Backout required

Explanation: The transaction is in backout (or rollback) required state, because it received a backout request from the remote system, or because a previous synchronization level 2 intersystem request abnormally terminated. When in this state, the transaction must issue an EXEC CICS SYNCPOINT ROLLBACK command. The transaction did not issue a rollback command, and the command that it did issue failed with this error code.

System action: CICS abnormally terminates the intersystem request.

User response: Examine, then correct, the logic of the transaction program to ensure that it issues an EXEC

CICS SYNCPOINT ROLLBACK command when in backout required state. The CSMT log might show additional messages that help you identify the command that was issued and caused this error.

15a00006/15a0010c Local SNA not initialized

Explanation: An intersystem request is using a Communications Definitions (CD) entry that is configured to use local SNA. However, local SNA is not enabled in your region.

System action: CICS abnormally terminates the intersystem request.

User response: Configure your region to use local SNA (as described in "Configuring CICS for local SNA support" on page 70), and retry the request.

15a00006/15a00106 Communications Protocol Specific Error

Explanation: A communications protocol specific failure has occurred. This might occur if a transaction has been abnormally terminated.

System action: The communications conversation fails.

User response: If the problem remains, contact your service representative.

15a00007/a0000100 Conversation Failure

Explanation: The conversation to the remote system has failed.

System action: The communications conversation has been terminated.

User response: Check whether the remote system is available and whether the link between the local system and the remote system is working.

15a00007/10086021 Transaction Unknown on Remote System

Explanation: The transaction is not known by the remote system.

System action: The communications conversation has not been started, or has been terminated.

User response: Check whether the transaction is defined on the remote system.

Check whether the communication products that are used by the remote system are available.

15a00007/10086031 PIP Data Not Supported by the Remote System

Explanation: A CICS transaction sent Process Initialization Parameter (PIP) data to a remote system that does not support PIP data. The use of PIPLENGTH and PIPLIST on the EXEC CICS CONNECT PROCESS is not allowed on this conversation.

System action: The conversation state is switched to FREE.

User response: If using distributed transaction processing (DTP), check the usage of PIPLENGTH and PIPLIST on the calls to EXEC CICS CONNECT PROCESS. Refer to the *TXSeries for Multiplatforms Application Programming Reference* for more information. Also check the SNA configuration in the remote system because it might be possible to enable PIP data.

15a00007/10086032 PIP Data Incorrectly Specified

Explanation: A CICS transaction sent incorrectly specified Process Initialization Parameter (PIP) data to a remote system. The remote system rejected the PIP data. PIP data is passed by using the PIPLENGTH and PIPLIST options on the EXEC CICS CONNECT PROCESS is command.

System action: The connection will not be made.

User response: If using distributed transaction processing (DTP), check the usage of PIPLENGTH and PIPLIST on the calls to EXEC CICS CONNECT PROCESS. If the problem remains, contact your service representative. Refer to the *TXSeries for Multiplatforms Application Programming Reference* for more information.

15a00007/10086034 Conversation Type Mismatch

Explanation: An attempt has been made to allocate a mapped conversation, but the remote system was expecting a basic conversation. CICS supports only mapped conversations, so the remote system and program need to be changed.

This error can also occur if CICS receives a request from a remote system to attach a basic conversation

System action: The communications conversation has not been started, or has been terminated.

User response: Change the configuration and the program in the remote system so that it can accept a mapped conversation from your region. then rerun the request.

15a00007/10086041 Sync Not Supported by Remote Program or System

Explanation: Indicates that synchronization level 2 conversations are not supported.

System action: CICS abnormally terminates the intersystem request.

User response: If using distributed transaction processing (DTP), check the SYNCLEVEL option that is used on CONNECT PROCESS. Refer to the *TXSeries for Multiplatforms Application Programming Reference* for more information.

15a00007/80f6051 Security Violation

Explanation: The remote system has rejected an intersystem request because the local transaction, or the local region, does not have the authority to access the required resource. The reason could be:

- The Communications Definitions (CD) entry that is defined in your region for the remote system is not configured to send user IDs. (Refer to the **OutboundUserIds** attribute of the CD.)
- The user ID that is associated with the local transactions does not have enough authority on the remote system to access the resources that are requested in the intersystem request.

System action: CICS abnormally terminates the intersystem request.

User response:

- Check whether the CD entry is configured to flow user IDs (OutboundUserIds=sent).
- Check whether the security configuration in the remote system allows your local region, and the user ID of the user who is making the intersystem request, to gain access to the required resource.

Refer to Chapter 6, "Configuring intersystem security," on page 123 for more information about how to configure for security.

15a00007/84b6031 Remote Transaction Temporarily Not Available

Explanation: The transaction is known at the remote system, but cannot be started.

System action: The communications conversation has not been started, or has been terminated.

User response: Look at the errors that are logged by the remote system. Determine why the transaction is not available on the remote system at the present time.

15a00007/84c0000 Remote Transaction Not Available

Explanation: The transaction is known at the remote system, but cannot be started.

System action: The communications conversation has not been started, or has been terminated.

User response: Look at the errors that are logged by the remote system. Determine why the transaction is not available on the remote system.

15a00007/8640000 Conversation Abnormal Termination

Explanation: The remote program or system has performed an EXEC CICS ISSUE ABEND.

System action: CICS abnormally terminates the intersystem request.

User response: Look for additional error messages in both the local and remote system that might indicate the cause of the network failure. If using distributed transaction processing (DTP), check whether the front-end and back-end transactions that are involved in the communications conversation are correct. Refer to "Communicating errors across a conversation" on page 295 for more information.

If the problem remains, contact your service representative.

15a00007/8640001 Conversation Abnormal Termination

Explanation: The remote system has performed an EXEC CICS ISSUE ABEND.

System action: The communications conversation is abnormally terminated.

User response: Look for additional error messages in both the local and remote system that might indicate the cause of the network failure. If using distributed transaction processing (DTP), check whether the front-end and back-end transactions that are involved in the communications conversation are correct. Refer to "Communicating errors across a conversation" on page 295 for more information.

If the problem remains, contact your service representative.

15a00007/8640002 Conversation Abnormal Termination

Explanation: The remote system has abnormally terminated the conversation because of a timeout.

System action: The communications conversation is abnormally terminated.

User response: Look for additional error messages in both the local and remote system that might indicate the cause of the network failure. If using distributed transaction processing (DTP), check whether the front-end and back-end transactions that are involved in the communications conversation are correct. Refer to "Communicating errors across a conversation" on page 295 for more information.

If the problem remains, contact your service representative.

15a00008/15a00107 Conversation Terminated Normally

Explanation: The remote system has terminated the conversation normally.

System action: The communications conversation ends normally, but unexpectedly.

User response: Look for additional error messages in both the local and remote system that might indicate the cause of the network failure. If using distributed transaction processing (DTP), check whether the front-end and back-end transactions that are involved in the communications conversation are correct. Refer to the description of FREE in the *TXSeries for Multiplatforms Application Programming Reference* for more information.

If the problem remains, contact your service representative.

15a00009/8240000 Backed Out

Explanation: The remote system has requested that work that has been done since the last sync point be backed out. This indicates that the transaction has been abnormally terminated.

System action: The local transaction will back out to the previous sync point.

User response: Check for any error messages that are logged by the remote system. Refer to the description of the *TXSeries for Multiplatforms Application Programming Reference* for more information.

PPC Gateway server problem determination

The PPC Gateway server uses the standard CICS Toolkit trace facilities to generate its messages and its trace. This means that the style and content of its output is very different from the CICS messages and trace that is described in *TXSeries for Multiplatforms Messages and Codes* and *TXSeries for Multiplatforms Problem Determination Guide*. The following information describes the general layout of the PPC Gateway server's messages and trace and provides explanations of some of the messages that you might see.

Format of PPC Gateway server messages and trace

The PPC Gateway server produces messages and trace that are in the same format. Below is an example of a PPC Gateway server message, showing the information that it contains.

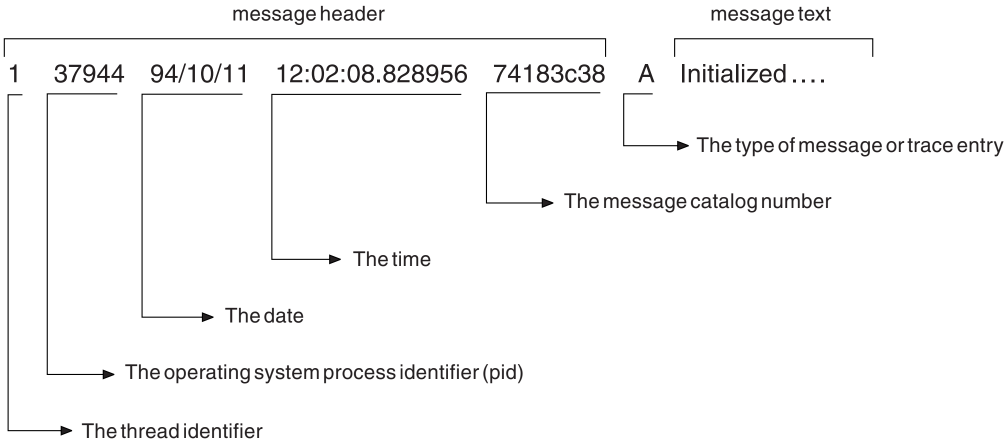

message header
message text
1 37944 94/10/11 12:02:08.828956 74183c38 A Initialized….
The type of message or trace entry
The message catalog number
The time
The date
The operating system process identifier (pid)
The thread identifier

Figure 87. Example PPC Gateway server message

Table 41 shows the different message types the PPC Gateway server uses. Refer to "PPC Gateway server message descriptions" on page 232 for descriptions of individual messages.

Table 41. Message types

Designator	Description
A	These messages are named **audit** messages. They record significant events that occur while the PPC Gateway server is running. The text that appears in the message is free format. Here is an example (this message is produced when the PPC Gateway server has completed initialization): `31    22644 96/01/01-19:43:43.744425 74182e16  A  Ready...`

Table 41. Message types (continued)

Designator	Description

W	These messages are named **attention** messages. They record unexpected events that occur while the PPC Gateway server is running. Here is an example of an attention message:
	`31    22644 96/01/01-19:43:43.744425 a0040437  W  System crash`
	`         imminent, machine low on swap space. (0x74183027) : ppc_gwy`
	The first line of text describes the error condition. The second line, that describes where in the PPC Gateway server code the message was produced, is in two parts. The first part contains the value, in parentheses, of the NLS message number (0x74183027 in the example). The second part, which is separated by a colon (:), is the name of the component within the PPC Gateway server that produced the attention (ppc_gwy in our example). The component name is useful if you have to investigate a problem, because it is possible to switch on the PPC Gateway server trace so that it is produced only by a particular component. The attention message might indicate which component is failing and, therefore, which component to trace.Need to replace this with an example that does not mention

F	These messages are named **fatal** messages and are issued just before the PPC Gateway server exits. They indicate that an event has occurred which the PPC Gateway server can not process. Here is an example of a fatal error message that was produced by the PPC Gateway server. Notice that it follows a similar format to the format of the attention message above:
	`31    22644 96/01/01-19:43:43.744425 50400415 F  unable to`
	`         create signal handler thread`
	When these messages occur, the PPC Gateway server writes as much debugging information as it has to a file named EncinaBacktrace.*pid*, where *pid* is the operating system process identifier for the failed PPC Gateway server process. This file is written to the directory in which you started the PPC Gateway server. You can format it by using the **interpretTrace** utility as follows:
	`interpretTrace < EncinaBacktrace.pid > outputfile`

The remaining types of PPC output are trace entries. This trace is designed for use by the PPC developers. Therefore, a precise definition of the contents of the PPC Gateway server trace is not published. Because it is often possible to follow the trace output, a general overview is given below:

Table 42. Trace types

Designator	Description

>	This is a function entry trace point. It is produced on entry to a function if trace is switched on. Here is an example that contains the function name, followed by the source file and component to which the function belongs:
	`31  22644 96/01/01-19:43:43.744425 30349826  > ppc_snaOS_Open`
	`       ppc_snaOS.c ppc_sna`

Table 42. Trace types (continued)

Designator	Description

<	This is a function exit trace point. It is produced on exit from a function if trace is switched on. Two types of exit trace are possible. Which trace is used depends on whether the function returns datat. An example of each is shown below:
	`31  22644 96/01/01-19:43:43.744425 00000006 <  ppc_snaOS_Init`
	`       ppc_snaOS.c ppc_sna`
	`31  22644 96/01/01-19:43:43.744425 28040c00 <R ppc_snaOS_Open`
	`       ppc_snaOS.c ppc_sna -> 7601a003`
	The first example shows the exit from ppc_snaOS_Init, which is a function that returns no data (a void function). The second example shows the exit from ppc_snaOS_Open which returns a return code. Most (but not all, so beware when reading the trace) of these PPC functions return a ppc_status_t return code, which are defined in:
	install_directory`/include/ppc/ppc_status.h`
	A value of 00000000 means that the function was successful; a hexadecimal value that begins with 7601a indicates an unsuccessful response. The **translateError** command can be used to find out what the unsuccessful response was. This is how you would find out what the **7601a003** response meant:
	`$ translateError 0x7601a003`
	`ENC-ppc-003: Allocate failure, retry`
	The output of **translateError** is in the same format as that which CICS uses to display return codes from the PPC Gateway server in the CICS messages. Note that **translateError** is NLS enabled. If you see the following output:
	`$ translateError 0x7601a003`
	`ENC-ppc-003`
	Check that your LANG environment variable is set up correctly.

P	This is a trace entry that shows the **parameters** that are passed to a function. The contents of this type of trace entry varies widely, depending on the function, and might span several lines. Here are some examples:
	`31  22644 96/01/01-19:43:43.744425 28040c23  P  address:`
	`  0x202359a8`
	`31  22644 96/01/01-19:43:43.744425 28040c01  P  *dataLengthP:`
	`  16, *requesttosendP: 0`
	`      *whatDataReceivedP: PPC_RECEIVED_DATA_COMPLETE,`
	`      *whatControlReceivedP: PPC_RECEIVED_CONTROL_SEND`

E	This final type of trace is an **event** trace. It can be used to show the data that the PPC Gateway server is processing, or error conditions. Here are some examples:
	`31  22644 96/01/01-19:43:43.744425 04800017  E  ppc Config:`
	`  Adding SNA tpn[3] : CEMT`
	`31  22644 96/01/01-19:43:43.744425 10043819  E  result mask:`
	`  00000000`

Tracing a PPC Gateway server

Tracing in a PPC Gateway server is controlled by the **-t** and **-T** parameters of the **cicsppcgwy**.

The **-t** option tells the PPC Gateway server which components in the PPC Gateway server to trace. Here are some examples of PPC Gateway server components:

ppc_sna
This component issues the calls to your SNA product. Tracing ppc_sna shows you the calls that are being made and the responses given.

ppc_gwy
This component provides the overall control for the PPC Gateway server. Tracing ppc_gwy shows you the requests that the PPC Gateway server is processing.

ppc_tcp
This component controls the communications to your CICS region across TCP/IP.

ppc_snp
This component controls sync point and backout processing.

ppc_rsn
This component controls resynchronization.

To turn tracing on in a component, specify the component name followed by **:0x1f**. For example:

```
-t ppc_sna:0x1f
```

You can specify more than one component as follows:

```
-t ppc_sna:0x1f:ppc_gwy:0x1f
```

The **-T** parameter specifies where the trace information is written to. If you do not specify a **-T** parameter, the destination for trace is the internal ring buffer in the PPC Gateway server. This is a cyclic buffer so only the most recent trace is kept. You can display the contents of the ring buffer by using the **tkadmin dump ringbuffer**. The trace information is also dumped to the **EncinaBacktrace** file that the PPC Gateway server produces when it exits unexpectedly. Using the ring buffer is useful if the PPC Gateway server has to run for a while before the problem that you are trying to trace occurs. If you want the trace to be written to the same file asare the PPC Gateway server messages, specify **-T all:stdout**. If you want the trace to be written to another file, specify **-T all:*filename***. This command also causes the PPC Gateway server messages to be written to this file.

The example below shows the:
`/.:/cics/ppc/gateway/cicsgwy`

PPC Gateway server being started with components ppc_sna and ppc_gwy tracing. The output will be written to the PPC Gateway server's message file:
`/var/cics_servers/GSD/cics/ppc/gateway/cicsgwy/msg`

```
% cicsppcgwy /.:/cics/ppc/gateway/cicsgwy -t ppc_sna:0x1f:ppc_gwy:\
         0x1f -T all:/var/cics_servers/GSD/cics/ppc/gateway/cicsgwy/msg
```

Note: Your PPC Gateway server can produce a large amount of trace output so, if you have requested that the trace be sent to a file, ensure that there is plenty of disk space available. Trace also slows down the PPC Gateway server. Therefore, use it only when you are trying to solve a problem.

PPC Gateway server message descriptions

While the PPC Gateway server is running it writes messages to a file that is in a directory under **/var/cics_servers**. For example, if the PPC Gateway server is named:

```
/.:/cics/ppc/gateway/cicsgwy
```

the message file is named:

```
/var/cics_servers/cics/ppc/gateway/cicsgwy/msg
```

Below are descriptions for common PPC Gateway server messages. They are sequenced by message type as follows:
- Audit messages (A)
- Attention messages (W)
- Fatal messages (F)

(Refer to "Format of PPC Gateway server messages and trace" on page 228 for information about PPC Gateway server message types.) Within each message type, the messages are shown in alphabetical sequence.

Audit messages

A connection down: *regionLU*

Example:

```
A  connection down: OPENCICS
```

Explanation: The PPC Gateway server cannot start a listener for your CICS region's Logical Unit (LU) *regionLU* because your SNA product is unavailable. The PPC Gateway server needs a listener in order to receive intersystem requests from remote SNA systems.

System action: The PPC Gateway server repeatedly tries to start the listener.

User response: Restart your SNA product.

A Connection Failure during Resynchronization for tid: *tId***, luwid:** *luwId***. localLu:** *regionLU* **- partnerLu:** *remoteSystem*

Example:

```
A  Connection Failure during Resynchronization for tid: 0x2, luwid: 0x207e797c
   {length: 16, luName: MYSNANET.TESA232, instance: 7f09b9643173,
    sequence: 1}. localLu: OPENCICS - partnerLu: CICSESA
```

Explanation: The PPC Gateway server needs to communicate with a remote system *remoteSystem* in order to exchange log names or resolve the outcome of a particular Logical Unit of Work (LUW). However, the gateway cannot allocate a conversation to this remote system. This could be because:
- Your SNA product is not running, or
- The link/connection to the remote system is down, or
- The remote system itself is unavailable

The *luwId* is the SNA identifier for the LUW. It is displayed in the following format:

luwid: *address* {length: *luNameLen*, luName: *luName* , instance: *instance*, sequence: *seqno*}

where:
- *address* is the address of the LUW record in the PPC Gateway server's storage
- *luName* is the Logical Unit (LU) name of the system/terminal that started the LUW
- *luNameLen* is the length of *luName*
- *instance* and *seqno* uniquely identify the LUW

The *tId* is the local internal name for the LUW.

System action: The PPC Gateway server will retry the allocate request.

User response: Restore the connection to the remote system so that the PPC Gateway server can communicate with the remote system.

A Connection Failure during Resynchronization for localLu: *regionLU* **- partnerLu:** *remoteSystem*

Example:

```
A  Connection Failure during Resynchronization for localLu:
   OPENCICS - partnerLu: CICSESA
```

Explanation: The PPC Gateway server needs to communicate with a remote system *remoteSystem* in order to exchange log names or resolve the outcome of a particular Logical Unit of Work (LUW). However, the gateway cannot allocate a conversation to this remote system. This could be because:
- Your SNA product is not running or
- The link/connection to the remote system is down or
- The remote system itself is unavailable

System action: The PPC Gateway server will retry the allocate request.

User response: Restore communications with the remote system so that the PPC Gateway server can communicate with it.

A connection not available: *regionLU*

Example:

```
A  connection not available: OPENCICS
```

Explanation: The PPC Gateway server cannot start a listener for your CICS region's Logical Unit (LU) *regionLU* because your SNA product is unavailable. The PPC Gateway server needs a listener in order to receive intersystem requests from remote SNA systems.

System action: The PPC Gateway server repeatedly tries to start the listener.

User response: Restart your SNA product.

A connection up: *regionLU*

Example:

```
A  connection up: OPENCICS
```

Explanation: The PPC Gateway server has successfully started a listener for your CICS region's Logical Unit (LU) *regionLU*. Inbound intersystem requests to the CICS region will now be passed from Communications Server for AIX to the PPC Gateway server.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A gateway configuration local lu data written to log

Explanation: A CICS region has passed configuration information to the PPC Gateway server and the PPC Gateway server has saved this to its log file stored in its logical volume.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A gateway configuration restore from log

Explanation: The PPC Gateway server has read all the configuration information from its log file that is stored in its logical volume.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A Initialized ...

Explanation: This message means that the PPC Gateway server has completed initialization and is ready for requests. When this message is displayed the PPC Gateway server will accept **acl_edit** commands, CICS configuration calls and intersystem requests from both remote SNA systems and from your local CICS region.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A local lu entry *regionLU* deleted

Example:

```
A  local lu entry OPENCICS deleted
```

Explanation: All configuration information for a particular CICS region (for example, the region name) has been deleted from the PPC Gateway server.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A local lu entry *regionLU* created

Example:

```
A  local lu entry OPENCICS created
```

Explanation: A CICS region has passed its region name to the PPC Gateway server. The PPC Gateway server needs this information to pass intersystem requests from remote SNA systems to the CICS region.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A Log Name Mismatch during Resynchronization for tid: *tId*, luwid: *luwId*. localLu: *regionLU* - partnerLu: *remoteSystem*

Example:

```
A  Log Name Mismatch during Resynchronization for tid: 0x2, luwid: 0x207e797c
   (length: 16, luName: MYSNANET.TESA232, instance: 7f09b9643173, sequence: 1)
   localLu: OPENCICS - partnerLu: CICSESA
```

Explanation: This message indicates that either the PPC Gateway server, or the remote system, has been cold started while one or more incomplete logical units of work (LUW) are outstanding. This is a potentially serious condition because when a system is cold started, its old log file is discarded and replaced by a new, empty, log file. Therefore, it loses all the information about incomplete LUWs. When this happens, the PPC Gateway server and the remote system cannot decide the outcome (commit or backout) for these incomplete LUWs, and human intervention is required.

The *luwId* is the SNA identifier for the LUW. It is displayed in the following format:

luwid: *address* {length: *luNameLen*, luName: *luName* , instance: *instance*, sequence: *seqno*}

where
- *address* is the address of the LUW record in the PPC Gateway server's storage
- *luName* is the Logical Unit (LU) name of the system or terminal that started the LUW
- *luNameLen* is the length of *luName* and
- *instance* and *seqno* uniquely identify the LUW

The *tId* is the local internal name for the LUW.

System action: The PPC Gateway server continues processing. It might retry the request to resolve the LUW outcome with the remote system. Until this condition is resolved, your CICS region cannot use synchronization level 2 intersystem requests to the remote system.

User response: Work down the instructions shown that are below. After you have completed each step, check whether the problem is solved. If it is, stop. Each step that you take is more severe and the affects might be more widespread, so your aim is to solve the problem with as little intervention as possible.

1. Stop the PPC Gateway server, then restart by using an auto start.
2. Delete the resynchronization records from the PPC Gateway server. (Refer to "Canceling pending resynchronizations in the PPC Gateway server" on page 206 for information about how to do this). Then shutdown and auto start the PPC Gateway server. (See "Starting a PPC Gateway server" on page 186, "Starting a PPC Gateway server" on page 191, or "Starting a PPC Gateway server" on page 197.)
3. Allow all current intersystem requests that are running in the PPC Gateway server to complete normally, then prevent new intersystem requests from starting. Ensure that none of the remote systems that have been communicating with the PPC Gateway server has outstanding resynchronizations. (If the remote system is CICS Transaction Server for z/OS, outstanding or pending resynchronizations exist if the **Pen** indicator is set for the connection to your CICS region when you issue CEMT INQUIRE CONNECTION.) If outstanding resynchronizations exist, activate the connection and allow the resynchronizations to complete.
4. If these resynchronizations will not complete, delete the resynchronization records from the remote system. (For example if the remote system is CICS Transaction Server for z/OS, refer to the CEMT SET CONNECTION(xxxx) NOTPENDING command.)
5. Finally, shutdown the PPC Gateway server, and restart by using a cold start. Then shutdown and restart each of your local CICS regions that use the PPC Gateway server.

A Log Name Mismatch during Resynchronization for localLu: *regionLU* **- partnerLu:** *remoteSystem*

Example:

```
A  Log Name Mismatch during Resynchronization for
   localLu: OPENCICS - partnerLu: CICSESA
```

Explanation: This message indicates that either the PPC Gateway server, or the remote system, has been cold started while one or more incomplete logical units of work (LUW) are outstanding. This is a potentially serious condition because when a system is cold started, its old log file is discarded and replaced by a new, empty, log file. Therefore, it loses all the information about incomplete LUWs. When this happens the PPC Gateway server and the remote system cannot decide the outcome (commit or backout) for these incomplete LUWs, and human intervention is required.

System action: The PPC Gateway server continues processing. It might retry the request to resolve the LUW outcome with the remote system. Until this condition is resolved, your CICS region cannot use synchronization level 2 intersystem requests with the remote system.

User response: Work down the instructions shown below. After you have completed each step, check whether the problem is resolved. If it is, stop. Each step that you take is more severe and the affects might be more widespread, so your aim is to solve the problem with as little intervention as possible.

1. Stop the PPC Gateway server, then restart by using an auto start.
2. Delete the resynchronization records from the PPC Gateway server. (Refer to "Canceling pending resynchronizations in the PPC Gateway server" on page 206 for information about how to do this). Then shutdown and auto start the PPC Gateway server. (See "Starting a PPC Gateway server" on page 186, "Starting a PPC Gateway server" on page 191, or "Starting a PPC Gateway server" on page 197.)
3. Allow all current intersystem requests that are running in the PPC Gateway server to complete normally, then prevent new intersystem requests from starting. Ensure that none of the remote systems that have been communicating with the PPC Gateway server has outstanding resynchronizations. (If the remote system is CICS Transaction Server for z/OS, outstanding or pending resynchronizations exist if the **Pen** indicator is set for the connection to your CICS region when you issue CEMT INQUIRE CONNECTION.) If outstanding resynchronizations

exist, activate the connection and allow the resynchronizations to complete.

4. If these resynchronizations will not complete, delete the resynchronization records from the remote system. (For example if the remote system is CICS Transaction Server for z/OS, refer to the CEMT SET CONNECTION(xxxx) NOTPENDING command.)
5. Finally, shutdown the PPC Gateway server, and restart by using a cold start. Then shutdown and restart each of your local CICS regions that use the PPC Gateway server.

A Logging new Log name: *logName* **for partner:** *remoteSystem*

Example:

```
A  Logging new Log name: 0BBF2D7A for partner:  CICSESA
```

Explanation: The PPC Gateway server is about to write to its local log file the name of the log file that is used by the remote system *remoteSystem*. The PPC Gateway server and the remote system send each other their log file names whenever the connection between them is acquired. By storing the name of a remote systems' log file, it is possible for the PPC Gateway server to check whether the remote system is still using the same log file when the PPC Gateway server requests the outcome of a particular LUW.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A lu pair *regionLU/remoteSystem* **will use old rule state after bo and cc**

Example:

```
A  lu pair OPENCICS/CICSMVS will use old rule state after bo and cc
```

Explanation: The remote CICS system has informed the PPC Gateway server that it was written before changes were made to the sync point part of the SNA LU 6.2 architecture. This means that:

- If a TXSeries for Multiplatforms distributed transaction processing (DTP) program has conversations with one or more DTP programs that are running on this remote system, and a rollback (backout) occurs, the state of these conversations will be:
  - **Send (state 2)** if a backout request was sent to the remote system on this conversation by use of the EXEC CICS SYNCPOINT ROLLBACK command.
  - **Receive (state 5)** if the conversation received a backout request in the EIBSYNRB flag of the EXEC Interface Block (EIB), and responded with an EXEC CICS SYNCPOINT ROLLBACK command.
- The remote system does not use a fully qualified conversation correlator (CC) in resynchronization requests. (This is part of the **S.4 sync point** option set that is described in the **SNA LU 6.2 Reference: Peer Protocols (SC31-6808)** manual.)

This behavior does not affect the PPC Gateway server's ability to participate in distributed logical units of work (LUWs) with this remote system. The only difference that you might notice is that the state of DTP conversations to this remote system after a DTP program issues an EXEC CICS SYNCPOINT ROLLBACK command might be different from what you would have expected if the remote system used the new SNA architected rules. (These new rules set the conversation state to the value that it was at the beginning of the LUW, if a backout occurs.)

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A lu pair *regionLU/remoteSystem* **will not use flag byte or bo with rip**

Example:

```
A  lu pair OPENCICS/CICSESA will not use flag byte or bo with rip
```

Explanation: The remote system has informed the PPC Gateway server that it:

- Does not support the use of the flag byte in SNA presentation headers and
- Will not send a *resynchronization in progress (RIP)* indicator with a backout request

These functions are part of the **S.4 sync point** option set that is described in the **SNA LU 6.2 Reference: Peer Protocols (SC31-6808)** manual. The absence of support for these functions in the remote system does not affect the PPC Gateway server's ability to participate in distributed logical units of work with this remote system.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A LU pair disconnect: *regionLU/remoteSystem*

Example:

```
A  LU pair disconnect: OPENCICS/CICSESA
```

Explanation: The connection between the local CICS region *regionLU* and the remote system *remoteSystem* has been released (shutdown). This might have occurred because the remote system or the network has failed, or it might be as a result of operator action.

System action: The PPC Gateway server continues processing.

User response: Determine why the connection was released. If it is because of a network failure and the connection is currently required, attempt to restore it.

A new config data

Explanation: This message means that the PPC Gateway server has been cold started and so has no record of your CICS region and its transactions. Until your CICS region has configured the PPC Gateway server, the PPC Gateway server cannot send or receive intersystem requests on behalf of your CICS region. Also, it cannot exchange log names with a remote system for your region.

System action: The PPC Gateway server continues processing.

User response: When the PPC Gateway server displays the ″Initialized...″ message, restart your CICS region. This causes CICS to pass to the PPC Gateway server all the information that is needed in order to send and receive intersystem requests on behalf of your CICS region.

A Normal gateway server shutdown scheduled

Explanation: The PPC Gateway server has been requested to shutdown normally. The PPC Gateway server exits when all the intersystem requests that are running in the PPC Gateway server have completed.

System action: The PPC Gateway server stops accepting new intersystem requests. It monitors the intersystem requests that are still running, and when they are completed, the PPC Gateway server shuts down.

User response: This message is for information only. No action is required.

A Normal gateway server shutdown with no conversations

Explanation: The PPC Gateway server is about to shutdown. No outstanding intersystem requests are running in the PPC Gateway server.

System action: The PPC Gateway server shuts down.

User response: This message is for information only. No action is required.

A Partner Detected Protocol Violation: Resynchronization for tid: *tId*, luwid: *luwId*. localLu: *regionLU* - partnerLu: *remoteSystem*

Example:

```
A Partner Detected Protocol Violation: Resynchronization for tid: 0x2, luwid: 0x207e797c
   {length: 16, luName: MYSNANET.TESA232, instance: 70f9b9463713, sequence: 1}.
   localLu: OPENCICS - partnerLu: CICSESA
```

Explanation: The remote system *remoteSystem* abnormally terminated a conversation with the PPC Gateway server that was being used to resolve the outcome of a particular LUW. The reason for this could be that the process on the remote system that was controlling the conversation was canceled, or it failed. (For example, if the remote system is CICS Transaction Server for z/OS, CICS/MVS, or CICS/VSE, this would occur if the CLS2 transaction was purged.) Alternatively, the remote system might have detected a protocol error that the PPC Gateway server made.

The *luwId* is the SNA identifier for the LUW. It is displayed in the following format:

luwid: *address* {length: *luNameLen*, luName: *luName* , instance: *instance*

where
- *address* is the address of the LUW record in the PPC Gateway server's storage
- *luName* is the Logical Unit (LU) name of the system or terminal that started the LUW
- *luNameLen* is the length of *luName*
- *instance* and *seqno* uniquely identify the LUW

The *tId* is the local internal name for the LUW.

System action: The PPC Gateway server retries the request.

User response: Look for messages on the remote system that might explain why the conversation was abnormally terminated. If this is a persistent problem, take a trace of the SNA link while the PPC Gateway server is trying the request and contact the service representative.

A Partner Detected Protocol Violation: Resynchronization for localLu: *regionLU* - partnerLu: *remoteSystem*

Example:

```
A  Partner Detected Protocol Violation: Resynchronization for
   localLu: OPENCICS - partnerLu: CICSESA
```

Explanation: The remote system *remoteSystem* abnormally terminated a conversation with the PPC Gateway server that was being used to exchange log names. The reason for this could be that the process on the remote system that was controlling the conversation was canceled, or it failed. (For example, if the remote system is CICS Transaction Server for z/OS, CICS/MVS, or CICS/VSE, this would occur if the CLS2 transaction was purged.) Alternatively, the remote system might have detected a protocol error that the PPC Gateway server made.

System action: The PPC Gateway server retries the request.

User response: Look for messages on the remote system that might explain why the conversation was abnormally terminated. If this is a persistent problem, take a trace of the SNA link while the PPC Gateway server is trying the request, and contact the service representative.

A Ready for additional configuration ...

Explanation: The PPC Gateway server is part way through initializing. When the **A Initialized ...** messages is displayed, the PPC Gateway server is ready for intersystem requests.

System action: The PPC Gateway server continues initializing.

User response: This message is for information only. No action is required.

A Resynchronization Completed Successfully: for tid: *tId*, luwid: *luwId*. localLu: *regionLU* - partnerLu: *remoteSystem*

Example:

```
A  Resynchronization Completed Successfully: for tid: 0x6, luwid:
   0x205a3990 {length: 16, luName: MYSNANET.TESA232, instance:
   eaf612febe30, sequence: 1}. localLu: OPENCICS   - partnerLu: CICSESA
```

Explanation: The PPC Gateway server has successfully completed a resynchronization request with *remoteSystem*.

The *luwId* is the SNA identifier for the LUW. It is displayed in the following format:

luwid: *address* {length: *luNameLen*, luName: *luName* , instance: *instance*

where
- *address* is the address of the LUW record in the PPC Gateway server's storage
- *luName* is the Logical Unit (LU) name of the system or terminal that started the LUW
- *luNameLen* is the length of *luName*
- *instance* and *seqno* uniquely identify the LUW

The *tId* is the local internal name for the LUW.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A Resynchronization Completed Successfully: for localLu: *regionLU* - partnerLu: *remoteSystem*

Example:

```
A  Resynchronization Completed Sucessfully: for
   localLu: OPENCICS - partnerLu: CICSESA
```

Explanation: The PPC Gateway server has successfully completed a exchange log names request with *remoteSystem*.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A security disabled

Explanation: The PPC Gateway server writes this messages during its initialization if it was started with a **ProtectionLevel** set to **none**. (Refer to section "Viewing the attributes of a PPC Gateway server" on page 188 for information about viewing the attributes of the PPC Gateway server.) This means that the PPC Gateway server does not check the identity of any CICS region (or PPC executive application such as **ppcadmin**) that makes requests to it.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

A Signal -- Forced gateway server shutdown

Explanation: The PPC Gateway server has been requested to shutdown immediately. Before exiting, it closes its log file and abnormally terminates any intersystem requests that are using the PPC Gateway server.

System action: The PPC Gateway server abnormally terminates all intersystem requests that are running in the PPC Gateway server, and shuts down.

User response: This message is for information only. No action is required.

A Signal -- Normal gateway server shutdown

Explanation: The PPC Gateway server has been requested to shutdown normally. The PPC Gateway server exits when all the intersystem requests that are running in the PPC Gateway server have completed.

System action: The PPC Gateway server stops accepting new intersystem requests. It monitors the intersystem requests that are still running, and when they are completed, the PPC Gateway server shuts down.

User response: This message is for information only. No action is required.

A SIGUSR1 received (probably from AIX SNA) and ignored

Explanation: The operating system has passed a SIGUSR1 (value 30) signal to the PPC Gateway server. This signal usually indicates that a SNA connection to a remote system has shutdown unexpectedly.

System action: The PPC Gateway server continues. Any intersystem requests that are using the SNA connection that has shutdown might abnormally terminate.

User response: Look for further messages that are produced by the PPC Gateway server. These might indicate which connection has shut down. Follow the instructions for these messages.

A sna relay allocate failure, local LU = *regionLU***, remote LU =** *remoteSystem***, tpn =** *transId***, status =** *returnCode*

Example:

```
A sna relay allocate failure, local LU = OPENCICS, remote LU = CICSESA,
  tpn = CRSR, status = ENC-ppc-0001: Allocation  failure, all sessions busy
```

Explanation: This message means that the PPC Gateway server cannot route an intersystem request from your CICS region (*regionLU*) to a remote SNA System (*remoteSystem*). The *transId* is the name of the CICS transaction that failed and the *returnCode* indicates why the request failed. Here are some examples:

- ENC-ppc-0001: Allocation failure, all sessions busy

  All the contention winner sessions that are currently bound to the remote system are busy. If this is a persistent error, and if you are using CICS for AIX, use the **sna -display sl** command to determine how many contention winner sessions are currently bound between the remote system and your region. These are shown in the Act ConW column. At least one contention winner session should be shown for each of the mode groups that are used between your CICS region and the remote system. If this problem is persistent, activate some more contention winner sessions for your connection. If this fails, check your SNA definitions to determine why contention winner sessions cannot be bound.
- ENC-ppc-0002: Allocation failure, conversation type mismatch

  The remote SNA system has rejected the request for *transId* because CICS is using an SNA mapped conversation and the remote system expected a basic conversation.
- ENC-ppc-0003: Allocation failure, no retry

  Your SNA configuration has a problem that is preventing your SNA product from starting an intersystem request on the *remoteSystem*.
- ENC-ppc-0004: Allocation failure, retry

  Your SNA product cannot start an intersystem request on the *remoteSystem*. Check whether the remote system is available. If it is running, you might have an SNA configuration problem.
- ENC-ppc-0005: Allocation failure, PIP data not supported

  CICS has sent Process Initialization Parameter (PIP) data with an intersystem request, but the remote system does not support PIP data. If the intersystem request is a:

  - Function shipping
  - Distributed program link
  - Asynchronous processing or
  - Transaction routing request

  CICS retries the request without PIP data. If the intersystem request is a distributed transaction processing (DTP) conversation, modify the CICS transaction that issued the EXEC CICS CONNECT PROCESS PROCNAME(*transId*) to start the DTP conversation, so that the EXEC CICS CONNECT PROCESS command does not specify the PIPLIST and PIPLENGTH parameters.
- ENC-ppc-0006: Allocation failure, PIP data incorrectly specified

  A CICS transaction that is running on *regionLU* has issued an EXEC CICS CONNECT PROCESS command that specifies, using the PIPLIST and PIPLENGTH parameters, Process Initialization Parameter (PIP) data that has either:
  - More than 16 PIP elements or
  - A PIP elements that is longer than 64 bytes
  - A PIP elements that contains a space character
- ENC-ppc-0008: Allocation failure, security violation

  The intersystem request does not have enough authority to run in the remote system.
- ENC-ppc-0009: Allocation failure, LU does not support synchronization level

  regionLU has requested a synchronization level 2 conversation, but the SNA remote system *remoteSystem* does not support it. If the intersystem request is a:
  - Function shipping
  - Distributed program link
  - Asynchronous processing or
  - Transaction routing request

  CICS retries the request by using synchronization level 1. If the intersystem request is a distributed transaction processing (DTP) conversation, modify the CICS transaction that issued the EXEC CICS CONNECT PROCESS PROCNAME(*transId*) SYNCLVL(2) to use synchronization level 0 or 1.
- ENC-ppc-0010: Allocation failure, TPN does not support synchronization level

  Transaction *transId* on the remote system does not support the synchronization level that is requested on the intersystem request.
- ENC-ppc-0011: Allocation failure, TPN unavailable, no retry

  A configuration error has occurred that is preventing the remote system from starting the requested transaction *transId*.
- ENC-ppc-0012: Allocation failure, TPN unavailable, retry

  The remote transaction *transId* is temporarily unavailable.
- ENC-ppc-0013: Allocation failure, TPN unknown

  The transaction *transId* is not recognized by the remote system.
- ENC-ppc-0016: Connection failure, no retry

A configuration error is either in your SNA product, or in the SNA network. This error is preventing suitable sessions from being bound between your SNA product and the remote system.

- ENC-ppc-0017: Connection failure, retry

  This could mean that:
  - The remote system is unavailable
  - The session that is being used for the intersystem request has failed
  - Not enough sessions are available for the request
  - The connection to the remote system cannot be started
- ENC-ppc-0025: Failure

  The intersystem request has failed in an unexpected way. Look for other messages that the PPC Gateway server has produced that might indicate the cause of the error.
- ENC-ppc-0032: Invalid LU name

  This could mean that either of the following:
  - Your CICS region has not configured the PPC Gateway server. This means that the PPC Gateway server is unable to process intersystem requests from your region. Run the CGWY transaction to configure the PPC Gateway server. Then rerun the intersystem request.
  - The *regionLU* or *remoteSystem* values contain incorrect characters.
- ENC-ppc-0033: Invalid mode name

  The transaction that is running on your region and that issued the intersystem request, specified a modename that the remote system does not recognize. The modename might have been explicitly specified either in the PROFILE option of the EXEC CICS ALLOCATE command, or by the **SNAModeName** attribute of the Transaction Definitions (TD) entry. Alternatively, if a modename was not specified in the PROFILE or the **SNAModeName** attribute in the TD entry, CICS would have used the default modename that was specified in the **DefaultSNAModeName** attribute of the Communications Definitions (CD) entry.

  Refer to “Default modenames” on page 110 for further information on default modenames.
- ENC-ppc-0036: Invalid security level

  Your region passed a user ID with the intersystem request. However, the remote system does not expect to receive user IDs. Either:
  - Set **OutboundUserIds=not_sent** in the Communications Definitions (CD) entry for the remote system or
  - Change the connection definition in the remote system so that it can accept user IDs. For example, if the remote system is CICS Transaction Server for z/OS, CICS/MVS, or CICS/VSE, set the ATTACHSEC=IDENTIY in the CONNECTION definition.
- ENC-ppc-0073: Connection failure, no retry BO

  A configuration error is in either your SNA product, or in the SNA network. This error is preventing sessions from being bound between your SNA product and the remote system.
- ENC-ppc-0074: Connection failure, retry BO

  This could mean that:
  - The remote system is unavailable
  - The session that is being used for the intersystem request has failed
  - Not enough sessions are available for the request
  - The connection to the remote system cannot be started
- ENC-ppc-0079: Failure BO

  The intersystem request has failed in an unexpected way. Look for other messages that the PPC Gateway server produces that might indicate the cause of the error.

A *x* [TCP & SNA] conversations active at the Gateway. Waiting for all conversations to complete ...

Example:

```
A  2 [TCP & SNA] conversations active at the Gateway.
   Waiting for all conversations to complete ...
```

Explanation: The PPC Gateway server is about to shutdown. However, Outstanding intersystem requests are running in the PPC Gateway server.

System action: The PPC Gateway server waits until:

- All intersystem requests that are using the PPC Gateway server have completed, or
- It is requested to shut down immediately (in which case it abnormally terminates all remaining intersystem requests before it shuts down).

User response: This message is for information only. No action is required.

A tcp relay allocate failure, local LU = ***regionLU*****, remote LU =** ***remoteSystem*****, tpn =** ***transId*****, status =** ***returnCode***

Example:

```
A  tcp relay allocate failure, local LU = OPENCICS,  remote LU = CICSESA,
   tpn = XEMT, status = ENC-ppc-0013: Allocation failure, TPN unknown
```

Explanation: The PPC Gateway server cannot route an intersystem request from a remote SNA system (*remoteSystem*) to your cics region (*regionLU*). The *transId* is the name of the CICS transaction that failed and the *returnCode* indicates why the request failed. Here are some examples:

- ENC-ppc-0002: Allocation failure, conversation type mismatch

  Your region has rejected the request for *transId* because the remote system is using an SNA basic conversation and your CICS region supports only mapped conversations.
- ENC-ppc-0007: Allocation failure, transaction scheduler failure

The PPC Gateway server cannot contact *regionLU*. Check whether *regionLU* is running and whether the PPC Gateway server has been set up correctly to communicate with it.

- ENC-ppc-0012: Allocation failure, TPN unavailable, retry

  Your CICS region is not running.
- ENC-ppc-0013: Allocation failure, TPN unknown

  Your CICS region does not recognize the transaction *transId*.
- ENC-ppc-0017: Connection failure, retry

  Your CICS region, or the transaction *transId* failed part way through the intersystem request.
- ENC-ppc-0025: Failure

  The intersystem request has failed in an unexpected way. Look for other messages that the PPC Gateway server produces that might indicate the cause of the error.
- ENC-ppc-0079: Failure BO

  The intersystem request has failed in an unexpected way. Look for other messages that the PPC Gateway server produces that might indicate the cause of the error.

A Time expired; Forced gateway server shutdown with *x* SNA conversations

Example:

```
A  Time expired; Forced gateway server shutdown with 3 SNA conversations
```

Explanation: The PPC Gateway server is shutting down. *x* outstanding intersystem requests are running in the PPC Gateway server, and these will be abnormally terminated when the PPC Gateway server shuts down.

System action: The PPC Gateway server abnormally terminates the remaining intersystem requests, closes its log file, and exits.

User response: This message is for information only. No action is required.

A tpn not added: *transId*

Example:

```
A  tpn not added: CEMT
```

```
A  tpn not added: \* resync *\
```

Explanation: The PPC Gateway server cannot listen for intersystem requests for the transaction *transId* on behalf of one of your CICS regions. The cause of this problem could be:

- Your SNA product is not running.
- The SNA TPN profile that is configured in the **TPNSNAProfile** attribute of the Transaction Definitions (TD) entry for the transaction is not defined and verified in your SNA product. A possible reason for this is that the tpn profile name is TDEFAULT, which was valid for CICS/6000® and SNA services, but is not valid for AIX SNA Server/6000 Version 2., or for Communications Server for AIX.
- Another operating system process is already listening for this transaction on behalf of your CICS region. (Your SNA product will not allow two processes to listen for your CICS region’s transactions if those processes are using the same local LU name, because it would not know to which process to pass inbound requests.)

Refer to “Configuring CICS for local SNA from the command line” on page 75 for information about how to configure Listener Definitions (LD) entries.

System action: The PPC Gateway server retries to start the listener for your CICS region.

User response: First ensure your SNA product is running and start it if necessary. If your SNA product is running, check whether the TPN SNA profile is defined in your SNA product. If it is, you must determine which process is also listening for the transaction on behalf of your CICS region. This could be an independent SNA application program, or another PPC Gateway server, or your CICS region’s local SNA listener.

Check whether you are using a PPC Gateway server and CICS local SNA support and whether the Gateway Definitions (GD) entry in your region for the PPC Gateway server has a different **GatewayLUName** from that which is coded in the **LocalLUName** attribute of the Region Definitions (RD). The easiest way to do this is to look at your region’s startup messages because they show the default LU name for your region (this is used by your region’s local SNA listener) and the LU name that is to be used by the PPC Gateway server.

If you have more than one CICS region, check whether each region is using a different LU name.

List the PPC Gateway servers that are running on your machine. (See “Listing the PPC Gateway servers running on a machine” on page 189.) Look for two copies of the same PPC Gateway servers running. (If you discover that two PPC Gateway servers are running for your CICS region, stop both of them by using the kill command. Then restart a new copy of the PPC Gateway server.)

Finally look for SNA applications that are running on your machine and ensure that the SNA configuration that they are using is different from the configuration that your CICS regions are using.

A Unrecognized Error Occurred during Resynchronization for tid: *tid***, luwid:** *luwid***. localLu:** *regionLU* **- partnerLu:** *remoteSystem*

Example:

```
A  Unrecognized Error Occurred during Resynchronization for tid: 0x10000,
   luwid: 0x2081b97c {length: 16, luName: MYSNANET.TESA218,
   instance: 8b706bb88c78, sequence: 1}. localLu: OPENCICS
    - partnerLu: CICSESA
```

Explanation: This message means that a resynchronization request to *remoteSystem* has failed. Usually, the cause of this is that the **ppcadmin** command has been used to cancel all resynchronization requests to *remoteSystem*. However, this message might also appear as a result of other events occurring when the PPC Gateway server does not expect them.

The *luwId* is the SNA identifier for the LUW. It is displayed in the following format:

luwid: *address* {length: *luNameLen*, luName: *luName* , instance: *instance*

where
- *address* is the address of the LUW record in the PPC Gateway server's storage
- *luName* is the Logical Unit (LU) name of the system or terminal that started the LUW
- *luNameLen* is the length of *luName*
- *instance* and *seqno* uniquely identify the LUW

The *tId* is the local internal name for the LUW.

System action: The PPC Gateway server might retry the resynchronization request.

User response: Look for other error messages that were produced at the same time as was this message, and follow the instructions that are associated with those messages. No specific action is required for this message.

A Unrecognized Error Occurred during Resynchronization for localLu: *regionLU* - partnerLu: *remoteSystem*

Example:

```
A  Unrecognized Error Occurred during  Resynchronization for
   localLu: OPENCICS - partnerLu: CICSESA
```

Explanation: An exchange log names request to *remoteSystem* has failed. Usually, the cause of this that the **ppcadmin** command has been used to cancel all resynchronization requests to *remoteSystem*. However, this message might also appear as a result of other events occurring when the PPC Gateway server does not expect them.

System action: The PPC Gateway server might retry the exchange log names request.

User response: Look for other error messages that were produced at the same time as was this message, and follow the instructions that are associated with those messages. No specific action is required for this message.

Attention messages

W CreateLuEntry failed after add of local lu ENC-ppc-0025: Failure

Explanation: The PPC Gateway server cannot use the local LU name that CICS passes. This local LU name is configured in the **GatewayLUName** attribute of the Gateway Definitions (GD) entry fort the PPC Gateway server.

System action: The PPC Gateway server cannot receive incoming intersystem requests. Further messages will be logged.

User response: Look at additional messages in the PPC Gateway server message file, and follow the instructions that are given in these messages.

W failed to restore local lu *regionLU* from config: ENC-ppc-0025: Failure

Example:

```
W  failed to restore local lu OPENCICS from config: ENC-ppc-0025: Failure
```

Explanation: This message might occur if the PPC Gateway server is started while your SNA product is not running.

System action: The PPC Gateway server keeps polling to check whether your SNA product has been restarted.

User response: Start your SNA product and the links and connections to remote systems.

W failed to restore tpn entries [*regionLU*] from config: ENC-ppc-0055: SNA TPNs not configured

Example:

```
W failed to restore tpn entries [OPENCICS] from config:
  ENC-ppc-0055: SNA TPNs not configured
```

Explanation: This message might occur if the PPC Gateway server is started while your SNA product is not running.

System action: The PPC Gateway server keeps polling to check whether your SNA product has been restarted.

User response: Start your SNA product and the links and connections to remote systems.

W listener getStatus error

Explanation: Your SNA product failed while the PPC Gateway server was receiving a new intersystem request from a remote SNA system.

System action: The intersystem request is discarded and the PPC Gateway server starts polling to determine whether your SNA product has been restarted.

User response: Start your SNA product and the links and connections to remote systems.

W ppc_Extract failed (sna): *returnCode*

Example:

```
W  ppc_Extract failed (sna): ENC-ppc-0016: Connection failure, no retry
```

Explanation: The PPC Gateway server cannot receive an intersystem request from a remote system.

System action: The PPC Gateway server abnormally terminates the intersystem request.

User response: Look for other error messages that were produced at the same time as was this message, and follow the instructions that are associated with those messages. No specific action is required for this message.

W (SNA)EPERM: Primary group must be system

Explanation: The PPC Gateway server's user ID does not have its primary group set to a group name that is trusted by Communications Server for AIX. Until this is changed, the PPC Gateway server cannot receive:

- CICS user IDs
- Exchange lognames requests
- Resynchronization request

from remote systems.

System action: The PPC Gateway server continues processing but all synchronization level 2 intersystem requests and inbound security requests fail.

User response: List the attributes of the PPC Gateway server to determine the user ID. (See "Viewing the attributes of a PPC Gateway server" on page 188.) View the attributes for the user ID by using the **lsuser** ***userid*** command. The primary group is shown as the **pgrp** (it is usually set to **cics**). Add the primary group name to the SNA configuration, and activate this configuration. Finally, stop and restart SNA and the link stations to pick up the new group name.

W SNA_FAIL: Check sna system status (lssrc -s sna)

Explanation: Your SNA product is not running.

System action: The PPC Gateway server keeps polling to check whether SNA has been restarted.

User response: Start your SNA product and the links and connections to remote systems.

W SNA_NSLMT: Session resource limit encountered

Explanation: Your SNA product cannot bind any additional session for the connection because the session limit for the modename has been reached. The causes of this problem are:

- Your CICS region is sending more simultaneous intersystem requests that the modename is configured to allow.
- Your CICS region has issued a synchronization level 2 request but your SNA product is not configured to support synchronization level 2 requests. This means that all the sessions for the connection are bound at synchronization level 1. When the synchronization level 2 request is made, your SNA product tries to bind another session.

System action: The PPC Gateway server abnormally terminates the intersystem request.

User response: Check whether your SNA product is configured for synchronization level 2. If it is, ensure that the definition of your modenames in your SNA product allows for the number of simultaneous intersystem requests that your CICS region might issue.

W SNA_PARMS: Internal bad parameter

Explanation: Your CICS region has issued an intersystem request to a remote SNA system. This request has failed because one of your SNA configuration profiles contains an error. For example:

- A Communications Server for AIX LU 6.2 side information profile does not exist for your CICS region's LU name.
- The default mode name is not set up in the Communications Server for AIX LU 6.2 side information profile for your CICS region.
- A Communications Server for AIX LU 6.2 partner profile does not exist or is not set up correctly for the remote system. (For example, the **Partner LU alias** field is not set up.)

System action: The PPC Gateway server abnormally terminates the intersystem request.

User response: Correct your SNA configuration so that your SNA product has the information that it needs to process the intersystem request.

W SNA_PROTOCOL: System protocol error

Explanation: The PPC Gateway server has called Communications Server for AIX incorrectly. This mighty be because of incorrect SNA configuration, or it might be a programming error in the PPC Gateway server code.

System action: The PPC Gateway server abnormally terminates the current request. Further messages might be logged.

User response: Look for additional messages in the PPC Gateway server message file, and follow the instructions that are given in these messages. Check the SNA configuration that the PPC Gateway server uses, and make any necessary correction.

If you cannot solve the problem yourself, contact your support organization. "Getting further help" on page 215 describes the information that your support organization requires.

W sp support bits mismatch

Explanation: One of the remote SNA systems with which the PPC Gateway server is communicating has been upgraded so that it now supports a different set of SNA options. This message does not affect intercommunications with the remote system.

System action: The PPC Gateway server continues processing.

User response: This message is for information only. No action is required.

W System crash imminent, machine low on swap space.

Explanation: The PPC Gateway server has received the SIGDANGER signal from the operating system kernel. This means that your machine is running short of paging (swap) space. The PPC Gateway server ignores this signal, but Communications Server for AIX does not and terminates immediately.

System action: The PPC Gateway server continues processing.

User response: If required, restart your SNA product. If this error condition occurs regularly, you must increase the paging space for your machine. Refer to your AIX books for information about how to increase paging space.

W unable to listen for resync tp on local LU *regionLU***:** *returnCode*

Example:

```
W  unable to listen for resync tp on local LU OPENCICS:
   ENC-ppc-0055: SNA TPNs not configured
```

Explanation: The PPC Gateway server cannot start a listener for the SNA resynchronization transaction. The *returnCode* indicated the reason for the failure.

- ENC-ppc-0055: SNA TPNs not configured

  This means that either:

  – Your SNA product is not running.

  – The RESYNCTP TPN profile is not configured in Communications Server for AIX. Enter

    lssnaobj -t local_tp | grep RESYNCTP.

  – You have previously configured your CICS region to use a different gateway on this machine, and the other gateway is still running. Either stop the other gateway, or use ppcadmin to delete the luentries.

  – More than one copy of this PPC Gateway server is running with the same LU name. Refer to the description of the:

    A tpn not added: *transId*

    message for information about this problem.

- ENC-ppc-0088: Unknown LU Name

  The local CICS region configured the PPC Gateway server when your SNA product was unavailable. Ensure that your SNA product is running, then either:

  – Stop and restart your region or

  – Run the CGWY transaction

  to reconfigure the PPC Gateway server with the CICS region details.

System action: The PPC Gateway server retries the request to start the listener.

User response: Correct the cause of the problem.

Fatal messages

F Invalid trace specification -- *traceParameter*

Example:

```
 F  Invalid trace specification -- "/var/cics_servers/ ... /cicsgwy/msg"
```

Explanation: The PPC Gateway server has been started with an invalid **-t** or **-T** parameter. This is shown in *traceParameter*.

System action: The PPC Gateway server terminates immediately.

User response: Restart the PPC Gateway server with correct parameters.

Table 43. Authority required by PPC Gateway Server

CDS directory	Access required by the cics_ppcgwy group.
/.:	r--t---
/.:/cics	r--t---
/.:/cics/trpc	rwdtcia
/.:/cics/ppc	rwdtcia
/.:/cics/ppc/gateway	rwdtcia

Part 4. Writing application programs for intercommunication

This part describes how to write application programs that will be used in a distributed CICS network.

Table 44. Road map

If you want to...	Refer to...
Read about writing applications that use distributed program link	Chapter 10, “Distributed program link (DPL),” on page 247
Read about writing applications that use function shipping	Chapter 11, “Function shipping,” on page 257
Read about writing applications that use transaction routing	Chapter 12, “Transaction routing,” on page 263
Read about writing applications that use asynchronous processing	Chapter 13, “Asynchronous processing,” on page 273
Read about writing applications that use distributed transaction processing	Chapter 14, “Distributed transaction processing (DTP),” on page 279

Chapter 10. Distributed program link (DPL)

When a CICS program issues an EXEC CICS LINK command, control passes to a second program (referred to as the *linked-to program*) that is named in the EXEC CICS LINK command. The second program executes and, after completion, returns control to the first program (referred to as the *linking program*) at the instruction that follows the EXEC CICS LINK command. The linked-to program can return data to the linking program if the EXEC CICS LINK command has used the COMMAREA option to pass the address of a communication area.

Distributed program link (DPL) extends the use of the EXEC CICS LINK command so that the linked-to program can be on a remote CICS system. When the linked-to program is on a remote CICS system, it is referred to as a *back-end program*.

The following are some reasons why you might use DPL:

- To separate the end-user interface (for example, BMS screen handling) from the application business logic (for example, accessing and processing data). This makes it easier to port part of an application between systems, such as moving the end-user interface from a IBM mainframe-based CICS system to a TXSeries for Multiplatforms system.
- To obtain performance benefits from running programs closer to the resources that they access, therefore reducing the need for function shipping requests and reducing the number of flows of data between the connected CICS regions.
- Where applicable, to provide a simpler solution than distributed transaction processing (DTP).
- To access relational database management systems (RDBMSs) with a program using Structured Query Language (SQL).

DPL is used when either the linked-to program is defined as remote in the Program Definitions (PD) for that program, when the SYSID option is specified on the EXEC CICS LINK command, or when the program is linked with the dynamic distributed program link user exit. This is described in “Using the dynamic distributed program link user exit” on page 252.

Comparing DPL to function shipping

DPL is similar to function shipping in that it provides a way of executing an EXEC CICS call on a remote system. The difference is that the DPL function is used when the EXEC CICS LINK command is called, whereas function shipping is used for those EXEC CICS calls that access remote resources. In either case, CPMI and the mirror program are used on the remote system to execute the calls.

An example of a DPL request is given in Figure 88 on page 248. In this figure, the linking program issues a program-control EXEC CICS LINK command to a program named PGA. From the Program Definitions (PD), CICS discovers that PGA is owned by a remote CICS system named CICB. CICS changes the EXEC CICS LINK request into a suitable transmission format, then ships it to the remote system for execution.

In the remote system, a *mirror transaction* is attached. The *mirror program*, DFHMIRS, which is invoked by the mirror transaction, recreates the original request, issues it on the remote system, and, when the back-end program has run to completion, returns any communication-area data (COMMAREA) to the local region.

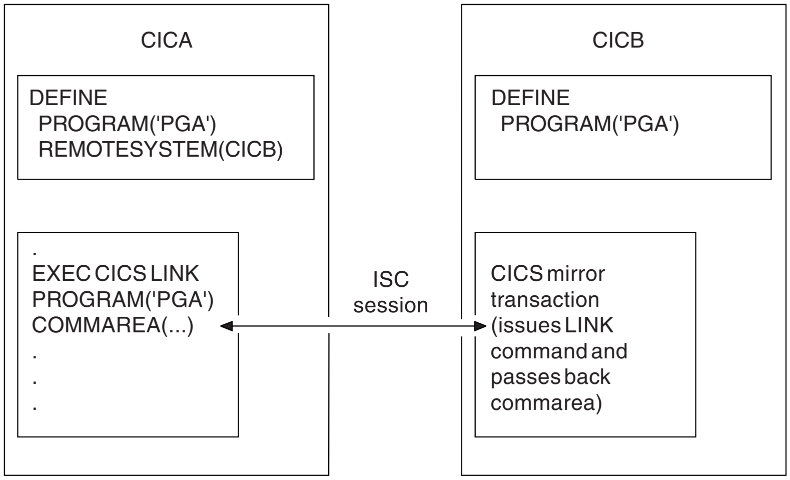


Figure 88. Distributed program link

It is possible for a back-end program to be defined on the remote system as remote. If this is the case, the link request is passed on. When this occurs, the systems are said to be *serially connected*.

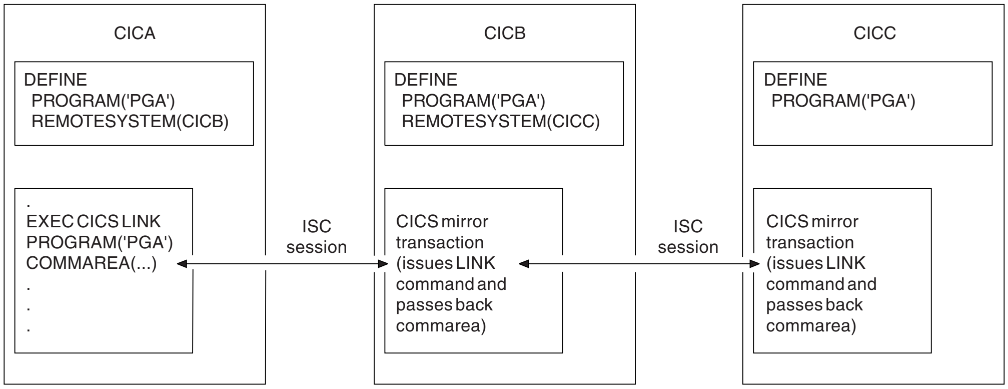


Figure 89. Serially connected systems

BDAM files, and IMS, DL/I, and SQL databases

DPL enables TXSeries for Multiplatforms application programs to access BDAM files and IMS™, DL/I, and SQL databases that are on a IBM mainframe-based CICS system. The TXSeries for Multiplatforms program links to a IBM mainframe-based CICS application program that reads and updates the databases or files.

Performance optimization for DPL

The performance of DPL can be affected by the amount of data transmitted, which includes the optional COMMAREA that is specified in an EXEC CICS LINK command. The length of the COMMAREA is 1 through 32767 bytes. CICS reduces the number of bytes transmitted by removing some trailing binary zeros from the COMMAREA before transmission and restoring them after transmission. This is transparent to the application programs, which always see the full-size COMMAREA.

When transmission time accounts for a significant part of the response time at a user terminal or workstation, application programs might be able to improve performance by using the DATALENGTH and LENGTH attributes in the EXEC CICS LINK command. LENGTH gives the number of bytes that are returned from the remote region in the COMMAREA, while DATALENGTH specifies a smaller size of COMMAREA that is sent from the local region to the remote region. If DATALENGTH is not specified, LENGTH is used to specify the number of bytes that are sent.

If the SYNCONRETURN option is not specified in the EXEC CICS LINK command, the mirror transaction remains active, and does not commit changes to resources, until the front-end program takes a sync point. This can produce unnecessary delays. Using SYNCONRETURN can prevent these delays, and also avoids communication between the front-end and back-end when the back-end resources are committed. The SYNCONRETURN option is not suitable for use, however, when the two regions share resources.

The dynamic distributed program link user exit can also be used to improve performance. This user exit is used to link a program dynamically, based on conditions that are specified in the user exit program. This is described further in "Using the dynamic distributed program link user exit" on page 252.

Restrictions on application programs that use DPL

Because the back-end program resides on a remote system, unpredictable results can occur that would not normally occur if the program were on the local system. Therefore, ensure that the back-end program is restricted to those functions that can perform reliably in a DPL condition. For example, do not attempt to link to back-end programs that share either a transient data queue or a temporary storage queue with the linking program. Use function shipping instead.

TXSeries for Multiplatforms enforces a subset of the full execution set of EXEC CICS calls. This subset is known as the *DPL subset*. If an attempt is made to issue a call that is not in the DPL subset, an INVREQ is returned to the linking program from the mirror program. To stay within the boundaries of the DPL subset, do not link to:

- Back-end programs that issue EXEC CICS XCTL commands.
- Back-end programs that issue EXEC CICS SYNCPOINT commands, unless SYNCONRETURN was specified in the EXEC CICS LINK command.
- Back-end programs that issue terminal control commands. Those commands are EXEC CICS CONVERSE, EXEC CICS HANDLE AID, EXEC CICS RECEIVE, EXEC CICS SEND, EXEC CICS WAIT TERMINAL, and EXEC CICS SEND TEXT.
- Back-end programs that issue Basic Mapping Support (BMS) commands. Those commands are EXEC CICS RECEIVE MAP, EXEC CICS SEND CONTROL, and EXEC CICS SEND MAP.
- Back-end programs that issue data interchange commands.
- Back-end programs that address the terminal user area (TCTUA), such as EXEC CICS ADDRESS TCTUA. Note that while you can use the transaction work area (TWA) in back-end programs, it is separate from the front-end TWA. You must use the COMMAREA to pass data between regions.
- Back-end programs that inquire on terminal attributes and that use the EXEC CICS ASSIGN command with the following options:
  - BTRANS
  - COLOR

- EXTDS
- FACILITY
- FCI
- GCHARS
- GCODES
- HILIGHT
- KATAKANA
- MAPCOLUMN
- MAPHEIGHT
- MAPLINE
- MAPWIDTH
- MSRCONTROL
- NETNAME
- NEXTTRANSID
- OPCLASS
- OPERKEYS
- OPSECURITY
- OUTLINE
- PS
- QNAME
- SCRNHT
- SCRNWD
- SIGDATA
- SOSI
- TCTUALENG
- TERMCODE
- UNATTEND
- VALIDATION

- Back-end programs that refer to the principal facility by using the following commands:
  - EXEC CICS ALLOCATE
  - EXEC CICS CONNECT PROCESS
  - EXEC CICS SEND
  - EXEC CICS RECEIVE
  - EXEC CICS CONVERSE
  - EXEC CICS FREE CONVID
  - EXEC CICS WAIT CONVID
  - EXEC CICS ISSUE ABEND
  - EXEC CICS ISSUE CONFIRMATION
  - EXEC CICS ISSUE ERROR
  - EXEC CICS ISSUE PREPARE
  - EXEC CICS ISSUE SIGNAL
  - EXEC CICS EXTRACT PROCESS

Security and DPL

If you specify the **RSLCheck** attribute as **internal** or **external** in the Transaction Definitions (TD) entry for a transaction, CICS raises the NOTAUTH condition on the local system if the transaction attempts to issue an EXEC CICS LINK command with the SYSID option specified. This prevents transactions from bypassing local security checking. Therefore, you must set the **RSLCheck** attribute to **none** for those transactions that invoke programs that use DPL.

In addition, you should be aware that when you are accessing DB2® data that is located on a IBM mainframe-based CICS system from a transaction in TXSeries for Multiplatforms, the mirror transaction (which is CPMI unless a different transaction is specified either with the EXEC CICS LINK command or in the Program Definitions (PD)) must have access to DB2 data. In all other respects (for example, security attributes and task priority) the mirror transaction on the IBM mainframe-based CICS system operates normally.

See “Security and function shipping” on page 141 for information about how to implement security checking when DPL is used.

Abends when using DPL

If the back-end program terminates abnormally, the mirror program returns an abend code. The code returned is that which would have been returned by an EXEC CICS ASSIGN ABCODE command. Note that the abend code that is returned to the linking CICS system represents the last abend to occur in the back-end program, which may have handled other abends before terminating.

Taking sync points when using DPL or function shipping

If the SYNCONRETURN option is not specified in an EXEC CICS LINK command, or for any type of function shipping, CICS initiates the commit procedure when the front-end program takes a sync point, by requesting the back-end CICS region to commit data changes. CICS then commits changes itself, after receiving confirmation of attached region commitment through the link.

You can, of course, have a number of connected CICS regions. In the case of multiple connected regions, the commit request is propagated through all the region connections. However, this is not safe with synchronization level 1.

You cannot take sync points in a back-end region unless the SYNCONRETURN option was specified on the EXEC CICS LINK command. If it was, sync points that are taken in the back-end program (either explicitly in the program, or implicitly taken by CICS when the linked-to program finishes) are not propagated to the front-end region. Because of this, no sharing of resources should occur between the front-end and back-end programs.

To determine whether the back-end program was invoked with the SYNCONRETURN option, the program can issue the EXEC CICS ASSIGN STARTCODE (or EXEC CICS INQUIRE TASK STARTCODE) command. These return a value of DS for a back-end DPL program that is started with the SYNCONRETURN option, and D for one that is started without it.

Multiple DPL links to the same region

When a front-end program issues a LINK command with the SYNCONRETURN option, the mirror transaction terminates as soon as control is returned to the front-end program. It is therefore possible for the front-end program to issue a subsequent LINK command to the same back-end region.

However, when a front-end program issues a LINK command without the SYNCONRETURN option, the mirror transaction is suspended pending a sync point request from the front-end region. The front-end program can issue subsequent LINK commands to the same back-end region provided that the SYNCONRETURN option is omitted and the TRANSID value is not changed. A subsequent LINK command with the SYNCONRETURN option or with a different TRANSID value is unsuccessful unless it is preceded by a SYNCPOINT command.

These errors are indicated by the INVREQ condition. An accompanying RESP2 value of 14 indicates that a sync point is necessary before the failed LINK command can be successfully attempted. A RESP2 value of 15 indicates that the TRANSID value is different from that of the linked mirror transaction.

Two ways to implement DPL in your application program

An application program can use DPL in two ways, either ignoring the location of the back-end program or explicitly specifying a remote system name. Dynamic distributed program linking, which is described in “Using the dynamic distributed program link user exit,” can be used both for implicit links and for explicit links.

Implicitly specifying the remote system

An application that uses DPL need not know the location of the back-end program; it can issue an EXEC CICS LINK command as if the program were owned by the local system. You specify that the back-end program is to run on a connected CICS region by providing the **RemoteSysId** and **RemoteName** attributes in the Program Definitions (PD) entry for the program. These attributes allow you to specify that the named program is owned by a remote system. The request is routed to the system that is named by the PD **RemoteSysId** attribute. Refer to the following example:

```
EXEC CICS LINK PROGRAM('LOCLPGM')
```

If the PD for LOCLPGM has a **RemoteSysId** defined, the link request is forwarded to that system.

Explicitly specifying the remote system

With the EXEC CICS LINK command, an application program can use the SYSID parameter to specify the connection to the remote system that owns the program. The advantage is that any system, including the local system, can be named in the SYSID attribute. Refer to the following example:

```
EXEC CICS LINK PROGRAM('LOCLPGM') SYSID('SYS1')
```

The EXEC CICS LINK command is routed to the region that is defined by the Communications Definitions (CD) SYS1. Any local Program Definitions (PD) entry for LOCLPGM is bypassed. A PD entry must be defined on SYS1 for LOCLPGM.

Using the dynamic distributed program link user exit

When a program link is requested from an application program, CICS links either to a program that is on the local system, or to a program that is on a remote system, depending on how the program has been defined in the PD, or on the parameters that are passed in the application program. The dynamic distributed program link user exit allows you to select dynamically the system to which the link request is routed.

CICS invokes the dynamic distributed program link user exit in the following conditions:

- When the user exit has been declared by using the **UserExitNumber** attribute in the PD for the user exit program. The user exit number is 50.
- When a program that has been specified in the PROGRAM field of the EXEC CICS LINK command is about to be linked to.
- When an External CICS Interface (ECI) program invokes a link request to a CICS application program.

- When a link request that is made by a CICS application program or by an ECI program specifies a program that does not have a PD for it.
- After a program that has been dynamically linked-to encounters a problem, and the initial invocation requests reinvocation at termination.
- After a program that has been dynamically linked-to successfully completes, and the initial invocation requests reinvocation at termination.

This user exit cannot be used:
- To link dynamically programs that are not started by using a link request
- When a short on storage problem occurs while the user exit is called
- To link dynamically a CICS-supplied transaction

Information passed to the user exit

CICS passes information to the dynamic distributed program link user exit by means of a parameter list. The parameter list contains both a standard header structure (cics_UE_Header_t) that is passed to all user exits, and a dynamic DPL specific structure (cics_UE015050_t). Both of these structures are included in the header file (cicsue.h).

References are made to the results of setting a parameter to a *null string*. A null string is a string that begins with a null character '\0'. For example, a null string could be placed into the SYSID parameter of the user exits specific structure by:

```
strcpy (UE_specificptr->UE_Dplsysid, "");
```

All other strings that are returned by the user exit must also be null terminated.

Initial invocation of the user exit

CICS invokes the dynamic distributed program link exit if a link request is issued for a program. This link request can take the form of either an EXEC CICS LINK command that is specified by an application program or a link request that is made by an ECI program.

If the program that is specified on the link request has a valid PD entry, the user exit is invoked with a reason of **UE_LINKSEL**. However, programs that are specified on link request do not require PD entries. If a link request for a program that does not have a database entry is received, the user exit is invoked with a reason of **UE_LINKUNKNOWN**. It is the responsibility of the user exit to decide whether to try rerouting the link request, or to decide that the program was invalid (by returning a return code of **UE_ProgramNotKnown**).

When CICS invokes the dynamic distributed program link user exit, it passes in information about where the program is to be run. This information consist of:
- The name of the remote system on which to run the program. This can be blank if the local system is to be used.
- The name of the mirror transaction that is to be used. This can be blank if the CICS-supplied mirror transaction CPMI is to be used.
- The name of the program that is to be run either on the local system, or on the remote system.
- The CICS user ID under which to run the program, on either local systems, or on remote systems. This is initially the user ID that is executing the EXEC CICS LINK command. This can be updated if the request is to the local system.

The above fields are set up by CICS to indicate the default action for a program link request. When the user exit invocation reason is **UE_LINKSEL**, these fields can remain as default, or be changed, depending on the action that y the user exit decides.

If the invocation is **UE_LINKUNKNOWN**, either the remote SYSID or the name of the program **must** be changed. Both of these fields (and the mirror transaction name) can be changed if needed.

The defaults for the above fields are decided by using the following, depending on whether an implicit or explicit request has been made, whether the program has a PD entry, and also whether this is a local or remote link request:

- If this is an explicit link request (regardless of whether the program has a PD entry):
  1. The remote program name is taken from the PROGRAM field of the EXEC CICS LINK command.
  2. The remote SYSID is taken from the SYSID field of the EXEC CICS LINK command.
  3. The mirror transaction ID is either taken from the TRANSID option of the EXEC CICS LINK, or, if no TRANSID option was specified, set to a null string.
- If this is an implicit request for a program that does not exist:
  1. The remote program name is taken from the PROGRAM field of the EXEC CICS LINK command.
  2. The remote SYSID is set to a null string.
  3. The mirror transaction ID is either taken from the TRANSID option of the EXEC CICS LINK, or, if no TRANSID option was specified, set to a null string.
- If this is an implicit local request for a program that does exist:
  1. The remote program name is taken from the PROGRAM field of the EXEC CICS LINK command.
  2. The remote SYSID is set to a null string.
  3. The mirror transaction ID is either taken from the TRANSID option of the EXEC CICS LINK, or, if no TRANSID option was specified, taken from the TransId field of the PD.
- If this is an implicit remote request for a program that does exist:
  1. The remote program name is taken from the RemoteName attribute of the PD, or, if this is a null string, the PROGRAM field of the EXEC CICS LINK command is used.
  2. The remote SYSID is taken from the RemoteSysId option of the PD entry for the program.
  3. The mirror transaction ID is either taken from the TRANSID option of the EXEC CICS LINK, or, if no TRANSID option was specified, taken from the TransId field of the PD.

Changing the target CICS system

The user exit is passed the default remote SYSID that CICS works out by using the rules that are shown in “Two ways to implement DPL in your application program” on page 252.

If the default SYSID is that of the local SYSID, a null string is passed to the user exit.

The information that is passed to the dynamic distributed program link exit in the user exit specific structure can be changed so that the distributed program link request can be rerouted.

If you want to reroute the program link request to the local system, leave the SYSID blank or set it to the local SYSID, which is passed in by using the parameter **UE_Dpllclsys**.

Changing the remote program name

The user exit is passed the default program name that CICS works out by using the rules that are shown in "Two ways to implement DPL in your application program" on page 252. If the request is local, this is the name of the program that is to be run on the local system. If the request is remote, this is the name of the program on the remote system.

Note: If the user exit returns a null string in the **UE_Dplprog** parameter, an abend and message is issued.

Changing the mirror transaction name

The user exit is passed the default mirror transaction ID that CICS work out by using the rules that are shown in "Two ways to implement DPL in your application program" on page 252. A blank mirror transaction ID indicates that the CICS-supplied mirror transaction CPMI will be used as the default.

Changing the user ID

The user exit is passed the user ID that is executing the EXEC CICS LINK command. If the request is to a remote system and the request is not already part of an existing logical unit of work in that system, the DPL request can be executed under the modified user ID at the remote system. The modified user ID does not need to be configured on the CICS system that is executing the DPL user exit. The user ID is sent the request as already verified; that is, any setting for the **OutboundUserids** attribute in the **Communication Definitions** for the remote system is ignored, and the user ID is sent without a password. If the exit changes the field UE_Dpluserid to a null string, the request is sent with no security attributes. If the user ID is unchanged, the security attributes are sent as configured by the **Communication Definitions** for the remote system.

If further requests are made within the same unit of work to the same remote system, the same user ID is used for all requests to that system for the particular logical unit of work.

Invoking the user exit at end of routed program

If you want your dynamic distributed program link user exit program to be invoked again when the routed program has completed, you must set the **UE_Dyropter** field in the parameter list to **UE_Yes** before returning control to CICS, on the initial invocation of the user exit.

The final exit is invoked either with a reason of **UE_LINKTERM** (the application program that is linked-to completed successfully) or **UE_LINKABEND** (the application program that is linked-to completed unsuccessfully). For both reasons, the parameter list contains the values that are used to route the link request.

Therefore, the final exit points can be used to keep statistics, such as which link requests have been successful, or which systems seem to be unavailable.

For more information about DPL, see the *TXSeries for Multiplatforms Administration Reference*.

Chapter 11. Function shipping

Function shipping enables CICS application programs to:

- Access CICS files that are owned by other CICS systems
- Transfer data to or from transient data and temporary storage queues in other CICS systems
- Initiate transactions in other CICS systems. This form of communication is described in “How to use asynchronous processing” on page 273

Note: Function shipped commands cannot access BDAM files, or IMS, DL/I, or DB2 databases in a IBM mainframe-based CICS. To access this data, use DPL as described in Chapter 10, “Distributed program link (DPL),” on page 247.

How to use function shipping

Use the guidelines in this section to write application programs that use function shipping. More information about function shipping is available in the *CICS Family: Interproduct Communication*

Two ways to use function shipping

An application program can use function shipping in two ways, either ignoring the location of resources, or explicitly specifying a remote system name.

- Implicitly specifying the remote system.

  An application that uses function shipping need not know the location of the requested resources; it can issue commands as if all resources are owned by the local system. CICS resource definitions allow the system programmer to specify that the named resource is owned by a remote system. The request is routed to the system that is named by the resource definition **RemoteSysId** attribute. Refer to the following example:

  ```
  EXEC CICS READ FILE(FILEA)
  ```

  The File Definitions (FD) for FILEA would have a **RemoteSysId** defined.
- Explicitly specifying the remote system.

  In a resource-accessing command, an application program can use the SYSID parameter to specify the connection to the remote system that owns the resource. The advantage is that any system, including the local system, can be named in the SYSID attribute. Refer to the following example:

  ```
  EXEC CICS READ FILE(FILEA)
            SYSID(SYS1)
  ```

  CICS routes the read request to the region that is defined by the Communications Definitions (CD) entry SYS1. Any local FD for FILEA is bypassed. A file should be defined on SYS1.

If the local SYSID is specified, the command is executed as if the SYSID option had not been given.

Serial connections

A definition of the resource that is being accessed is required in the remote CICS system to which the function shipping request is directed. This definition might itself be a remote definition, causing the request to be relayed to another CICS system. When this occurs, the linking system and the linked-to system are said to be serially connected.

CICS file control data sets

Function shipping allows read and update access to files located on a remote CICS system. Function ship of INQUIRE FILE and SET FILE are not supported.

Note: Take care when designing systems that use remote file requests that contain physical record identifier values (for example, VSAM RBA files, and files with keys not embedded in the record). Application programs that are in remote systems must have access to the correct values following the updating or reorganization of such files.

Transient data

When an application program accesses intrapartition or extrapartition transient data queues on a remote system, the queue definition that is in the remote system specifies whether the queue is protected, and whether it has a trigger level and associated terminal.

If a transient data destination has an associated transaction, the named transaction must be defined to be executed in the system that owns the queue; it cannot be defined as remote. If a terminal is associated with the transaction, it can be connected to another CICS system, and used through the transaction routing facility of CICS.

Local and remote names

Any type of remote resource can be defined with a local name that is different from its name in its owning system. This is useful when resources in different systems have the same name. For example, a program can send data to the CICS service destinations, such as CSMT, in both local and remote systems.

Synchronization

The CICS recovery and restart facilities ensure that when the requesting transaction reaches a sync point, any mirror transactions that are updating protected resources also take a sync point, so that changes to protected resources in remote and local systems are consistent. The CICS control region receives notification of any failures in this process, so that suitable corrective action can be taken. This action can be taken manually or by user-written code.

When a transaction issues a sync point request, or terminates successfully, the intercommunication component sends a message to the mirror transaction that causes it also to issue a sync point request and terminate. The successful sync point by the mirror transaction is indicated in a response sent back to the requesting system, which then completes its sync point processing, so committing changes to any protected resources.

Data security and integrity

Protection of data that is accessed by function shipping is the responsibility of the data-owning system.

A resource update that is caused by a function shipping request is committed when the request-issuing program issues a sync point request or terminates successfully. However, a risk occurs when shipping to more than one CICS region with synchronization level 1.

Application programming for function shipping

You write a program to access resources in a remote region, in much the same way as if the resources were on the local region.

The commands that you can use to access remote resources are:
- File control commands
- Temporary storage commands
- Transient data commands

Interval control commands are deliberately left out of this list. For information about this subject, see “Application programming for asynchronous processing” on page 277.

Your application can run in the CICS intercommunication environment, and use the intercommunication facilities, without being aware of the location of the resource that is being accessed. You define the resource location in the **Remote SysId** attribute of the appropriate CICS definition. Optionally, you can use the SYSID option on EXEC commands to select the region on which the command is to run. In this case, CICS does not reference the resource definitions on the local region unless the SYSID option names the local SYSID that is configured in the Region Definitions (RD) attribute **localSysId**.

When your application issues a command against a remote resource, CICS ships the request to the remote region, where a mirror transaction is initiated. The mirror transaction runs the request on your behalf, and returns any output to your application program. The mirror transaction is therefore, in effect, a remote extension of your application program.

Although the same commands are used to access local resources and remote resources, several restrictions apply when the resource is remote. For details of these restrictions, see “Exceptional conditions” on page 260.

Some errors that do not occur in single regions can occur when function shipping. For these reasons, you should always know whether resources that your program accesses can possibly be remote.

Long-running function shipping transactions that start multiple application servers can cause a degradation in performance when EXEC CICS SYNCPOINT is frequently used. Therefore, when coding EXEC CICS SYNCPOINT in your applications, decide the frequency with which you call it, by balancing your need for data integrity with the requirement for efficient use of machine performance.

File control

Function shipping allows you to access Structured File Server (SFS) or Virtual Sequential Access Method (VSAM) files that are on a remote region.

If you use the SYSID option to access a remote region directly, you must observe the following rules.

- For a file that is referencing a keyed file, you must specify KEYLENGTH if you specify RIDFLD, unless you are using Relative Byte Addresses (RBA) or Relative Record Numbers (RRN).
- If the file has fixed length records, you must specify the record length (LENGTH).

These rules also apply if the File Definitions (FD) entry for the file does not define the appropriate values.

Temporary storage

Function shipping allows you to send data to, or receive data from, temporary storage queues that are on remote regions. You can define remote temporary storage queues in the Temporary Storage Definitions (TSD). You can, however, use the SYSID option on the EXEC CICS WRITEQ TS, EXEC CICS READQ TS, and EXEC CICS DELETEQ TS commands, to specify the region on which the request is to run.

Transient data

Function shipping allows you to access intrapartition or extrapartition transient data queues that are on remote regions. You can define remote transient data queues in the Transient Data Definitions (TDD). You can, however, use the SYSID option on the EXEC CICS WRITEQ TD, EXEC CICS READQ TD, and EXEC CICS DELETEQ TD commands, to specify the region on which the request is to run.

If the remote transient data queue has fixed length records, you must supply the record length in the LENGTH option:

- If you do not specify the record length in the TDD entry for the transient data queue
- If you use the SYSID option

Exceptional conditions

Requests that are shipped to a remote region can raise any of the exceptional conditions for the command that can occur for a local resource. In addition, some conditions are possible that apply only when the resource is remote.

Remote region not available

At the time that CICS issues a function shipping request, a link to the remote region might not be available. If this is the case, CICS raises the SYSIDERR condition in the application program.

CICS also raises this condition if the named region is undefined, but this error should not occur in a production system unless the application obtains the name of the remote region from a terminal user.

The default action for the SYSIDERR condition is to abnormally terminate the task.

Invalid request

The ISCINVREQ condition occurs when the remote region indicates an error that does not correspond to a known condition. The default action is to terminate the task abnormally.

Mirror transaction abnormal termination

An application request against a remote resource can cause an abnormal termination in the mirror transaction (for example, the requested TDD might have been disabled).

In these conditions, CICS also abnormally terminates the application program, but with an abnormal termination code of ATNI.

Note: The ATNI abnormal termination, which is caused by a mirror transaction abnormal termination, is not related to a terminal control command, and, therefore, CICS does not raise the TERMERR condition.

Long-running mirror transactions

Mirror transactions normally terminate when they expect no more work from the requesting system. This is usually when the requesting transaction executes a sync point or terminates. The longevity of a mirror task is a compromise between the overhead of using resources (such as an application server and, possibly, an SNA session), while waiting for work, and the overhead of starting a mirror transaction in an application server and possibly allocating an SNA session.

For this reason, a mirror transaction that is started on a TCP/IP connected system at synchronization level 2 always waits for the requesting application to either terminate, or perform a sync point before terminating itself.

A mirror transaction that is started on an SNA-connected system at synchronization level 1 always terminates after each shipped request is completed, or when the requesting application terminates. The exception to this is when the mirror transaction has done recoverable work, or has a browse active, or when it has executed a DPL command.

When designing applications that use function shipping, you should be aware of the effects that long or short running mirror transactions have on system resource use in the resource owning region, and on the performance of your application. In this case, the mirror transaction terminates when the requesting transaction executes a sync point, or terminates.

Timeout on function shipped requests

When CICS receives a function shipped request, the started transaction is the mirror transaction. The CICS supplied definitions of the mirror transaction (CPMI, CVMI, and CSM*) all specify **DeadLockTimeout=0**. This means that the function shipped request does not timeout if the resource that it is attempting to access is locked. If you require function shipped requests to timeout in such a condition, change the **DeadLockTimeout** value for the mirror transactions accordingly.

Chapter 12. Transaction routing

Transaction routing allows terminals that are connected to one CICS region, to run with transactions that are in another connected CICS region. This means that you can distribute terminals and transactions among your CICS regions and still have the ability to run any transaction with any terminal.

In transaction routing, the two regions that are involved are referred to as:

The terminal-owning region
The CICS system on which the terminal is locally defined.

The application-owning region
The CICS system on which the application is locally defined.

For more information about transaction routing, see the *CICS Family: Interproduct Communication*

Initiating a transaction from a terminal

Generally, the ability to perform transaction routing is implemented by the way that the involved resources are defined to CICS. Those resources are:
- The initiating terminal
- The initiated transaction

The initiating terminal

How you define the terminal depends on whether or not the terminal definitions are shipped from the terminal-owning region to the application-owning region. If a terminal definition is shipped, enough data is passed with a transaction routing request to enable the remote system to install dynamically (autoinstall) the necessary remote terminal definitions. The benefit of shipping terminal definitions is that you do not need to define the terminal on the application-owning region.

If the terminal definitions are not shipped, the terminal definitions that are on the application-owning region require the following information to implement transaction routing. (See also “Defining terminals to the application-owning region” on page 118 for details).
- The local name of the terminal
- The name of the connection to the terminal-owning region
- The name of the terminal in the terminal-owning region
- The network name of the terminal

When terminals are defined on the application-owning region, take care to ensure that each terminal name is unique for that terminal wherever it is defined across the network. If the terminal names cannot be unique, you can use the Terminal Definitions (WD) **RemoteName** attribute to specify an alias, as described in the *TXSeries for Multiplatforms Administration Reference*.

The initiated transaction

The terminal-owning region requires a Transaction Definitions (TD) entry for the remote transaction. This definition entry need only specify the information that is needed to implement transaction routing, as follows:

- The local name of the transaction
- The name of the connection to the application-owning region
- The name of the transaction in the application-owning region
- The transaction is dynamically routed with the dynamic transaction routing user exit. Refer to “Dynamic transaction routing” on page 267.

All other transaction characteristics are defined in the local definition of the transaction in the application-owning region. See “Defining remote transactions for transaction routing” on page 119 for details.

Application programming for transaction routing

If you are writing a transaction that might be used in a transaction routing environment, you can design and code that transaction as you would for a single region. You must, however, be aware of several restrictions, which are described in this topic. The same considerations apply if you are migrating an existing transaction to the transaction routing environment.

Any Basic Mapping Support (BMS) maps that your program uses must reside in the application owning region. If you run a transaction from an EBCDIC system, you must not use square bracket characters in maps. For more information, see the *TXSeries for Multiplatforms Application Programming Guide*.

Pseudo-conversational transactions

A routed transaction requires the use of an intersystem (LU 6.2) conversation for as long as the transaction is running. For this reason, you should duplicate long-running conversational transactions in the two regions, or you should design the transactions as pseudo-conversational transactions.

Pseudo-conversational transactions are used in CICS application programs that consist, internally, of multiple tasks that are designed to appear to the operator as a continuous conversation. The program issues an EXEC CICS RETURN request with the TRANSID option. The next input from the terminal causes the specified transaction to be initiated unless the IMMEDIATE option is specified. If the IMMEDIATE option is specified, the transaction is initiated without waiting for any input from the terminal.

Take care when naming and defining the individual transactions that make up a pseudo-conversational transaction, because CICS returns a TRANSID, which is specified in an EXEC CICS RETURN command, to the terminal-owning region, where the TRANSID might be a local transaction.

You can design a pseudo-conversational transaction that is made up of local and remote transactions. However, if these transactions share a COMMAREA or TCTUA, you might need to code a DFHTRUC program to convert the data between the code pages that the two systems use. Refer to “Data conversion for transaction routing” on page 164 for more details.

The terminal in the application-owning region

The terminal with which your transaction runs, is represented by a Terminal Definitions (WD) entry, which is in many ways a copy of the real terminal definition entry that is in the terminal-owning region. This copy is known as the **surrogate** terminal definition entry. See “Shipping terminal definitions” on page 117.

Using the assign command in the application-owning region

The EXEC CICS ASSIGN command might perform differently in a transaction routing environment, from how it does in a single region. Therefore, you might need to include different processing to reflect this.

You might find that three of the options to the EXEC CICS ASSIGN command cause an unexpected reaction, or return unexpected values. A closer look at these helps you to understand why:

PRINSYSID

This option returns the SYSID of the principal facility to the transaction. This option requires that this facility be an LU 6.2 conversation. The principal facility for a routed transaction is represented by the surrogate terminal definition entry, which does not meet the requirement. Therefore, CICS raises the INVREQ condition.

Note: You cannot use an EXEC CICS ASSIGN PRINSYSID command to find the name of the terminal-owning region.

USERID

This option returns the user ID that is associated with the task. For a routed transaction, the user ID that is returned is based on:

- Whether security for inbound requests is “local” or “trusted”. In CICS, this would be specified with the Communications Definitions (CD) **RemoteSysSecurity** attribute in the application-owning region.
- Whether or not a user ID is sent from the terminal-owning region to the application-owning region. If the terminal-owning region is CICS, this is specified in the terminal-owning region with the CD **OutboundUserIds** attribute.
- Whether or not a link user ID is locally defined for the connection between the terminal-owning region and the application-owning region. In CICS, a link user ID is specified with the **LinkUserId** attribute.
- The value of the local default user ID (in those conditions when a user ID is not available; for example, when a user ID is not flowed and a link user is not defined.) In CICS, the default user ID is specified with the Region Definitions (RD) **DefaultUserId** attribute.

See “Link security and user security compared” on page 136 for information about how a user ID is determined for inbound requests. See also “Using CRTE and CESN to sign on from a remote system” on page 142.

OPERKEYS

This option returns a 64-bit mask that represents the TSL keys assigned to the remote user. The TSL keys assigned are based on the local definitions of the link keys for the connection in addition to the keys that are locally defined for the user ID that the user is logged on as.

If the remote user is signed on locally (for example, if the user uses CRTE to route to the remote system, then uses CESN to sign on to a local user ID), the returned mask represents the keys that are defined for the user in the definition entry and are also defined for the link.

In some conditions, the user might be given public access only.

See “Link security and user security compared” on page 136 for information about how security keys are assigned. See also “Using CRTE and CESN to sign on from a remote system” on page 142.

Automatic transaction initiation (ATI)

Automatic Transaction Initiation (ATI) is a process in which a transaction can request that another transaction be started automatically on a named terminal, when the named terminal becomes available.

Transaction routing allows ATI requests to be made that name terminals that are attached to a remote region. When the ATI transaction is ready to be run, it is shipped from the application-owning region to the terminal-owning region, naming the transaction that is to be started and the terminal on which it is to be started.

Terminal definitions for ATI

A definition of the remote terminal that names the remote terminal-owning region must be available so that the application-owning region can ship the ATI request. If the ATI started transaction becomes ready to run after the terminal definition has been deleted, CICS cannot find the terminal-owning region, and therefore cannot ship the ATI request to it. To solve this problem, either create a local Terminal Definitions (WD) entry of the remote terminal, or ensure that the ATI started transaction becomes ready to run before any autoinstalled WD is deleted. Refer to "Shipping terminal definitions" on page 117.

TXSeries for Multiplatforms does not support *terminal not found* user exit programs.

Note: Shipped terminal definitions exist from the time that the definition is received until a request is sent from the terminal-owning region to delete it. See "Shipping terminal definitions" on page 117 for more information.

Transaction definitions for ATI

In the terminal-owning region, a definition for the remote transaction must exist and, with the exception of dynamically routed transactions, the transaction definition must name the region on which the ATI request is to be initiated. (In TXSeries for Multiplatforms, this region is named with the Transaction Definitions (TD) **RemoteSysId** attribute.) In particular, the same ATI transaction name cannot be specified to run on two or more different systems. If this is required, you can do one of the following:

1. Create a TD entry for each ATI transaction, giving each a different name. If you create separate TD entries for each ATI transaction, the **RemoteName** and **RemoteSysId** attributes should then be used to distinguish between the two transactions.
2. Use the dynamic transaction routing User Exit. For information about using dynamic transaction routing, see "Dynamic transaction routing" on page 267.

Indirect links for transaction routing

Because CICS does not support indirect links for transaction routing, each region in a chain of three or more regions must contain a WD entry and TD entry for the ATI started transaction and the terminal against which it is to be started. These definitions must name the neighboring system, such that the terminal and transaction can be found. For example, consider the following three regions: REGIONA, REGIONB, and REGIONC. REGIONA is the terminal-owning region and REGIONC is the application-owning region. A terminal that is attached to REGIONA is starting a transaction called *XXXA*, which results in REGIONC trying to start a transaction on REGIONC called *XXXC*, as shown in Table 45 on page 267:

Table 45. Indirect links for transaction routing

REGIONA	REGIONB	REGIONC
RD: LocalSysId="REGA" CD: REGB WD: RemoteSysId="" NetName="TERM0001" TD: Entry for transaction\ "XXXA" RemoteName="XXXB" RemoteSysId="REGB"	RD: LocalSysId="REGB" CD: REGA CD: REGC WD: RemoteSysId= \ "REGIONA" NetName="TERM0001" TD: Entry for transaction\ "XXXB" RemoteName="XXXC" RemoteSysId="REGC"	RD: LocalSysId="REGC" CD: REGB WD: RemoteSysId="REGB" NetName="TERM0001" TD: Entry for transaction\ "XXXC" RemoteName="" RemoteSysId=""
In this table: • REGA, REGB, and REGC are the entries for REGIONA, REGIONB, and REGIONC respectively. • The RemoteName defaults to transaction entry name if not specified.		

Dynamic transaction routing

Transaction routing can be either static or dynamic:

- With *static transaction routing*, the transaction is routed to the system that is named with the **RemoteSysId** attribute in the Transaction Definitions (TD) for that transaction. In this case, the value of the TD **Dynamic** attribute is **no** in the local definition for that transaction.
- With *Dynamic transaction routing*, the transaction can be dynamically routed to any available system, either remote or local, when the transaction is started. In this case, the value of the TD **Dynamic** attribute is **yes** in the local definition for that transaction.

Dynamic transaction routing enables you to define how a transaction is to be routed depending on such factors as:

- Input to the transaction
- Available CICS systems
- Relative loading of the available systems

Also, a routing program can perform other functions besides redirecting transaction requests, such as:

- Balancing of workload. For example, in a multiple-CICS environment, your program could make intelligent choices between equivalent transactions on parallel systems.
- Handling transactions that cannot be routed, such as when no remote CICS regions are available.
- Handling abends in the routed-to transaction.
- Monitoring the number of requests that are routed to particular systems.

Dynamic transaction routing cannot be used to reroute remote ATI requests.

CICS manages dynamic transaction routing through the use of the CICS-supplied *dynamic transaction routing user exit*. This user exit is invoked:

- Before routing a transaction that is defined as **Dynamic=yes**
- If an error occurs in route selection

- At the end of a routed transaction if the initial invocation requests reinvocation at termination
- If a routed transaction abends and the initial invocation requests reinvocation at termination

Parameters are passed in a structure between CICS and the dynamic routing program. The program might change some of these parameters to influence subsequent CICS action. The parameters include:

- The reason for the current invocation.
- Error information.
- The name of the target system. Initially, this is the system that is specified with the TD **RemoteSysId** attribute. If is system is not specified, the name that is passed is that of the local system.
- The name of the target transaction. Initially, this is the name that is specified with the **RemoteName** attribute. If a target transaction is not specified, the name that is passed is the name of the local transaction.
- A pointer to the CWA.
- A pointer to the TCTUA.
- A user area.

A dynamic transaction routing program must follow standard user exit rules.

CICS supplies a sample program that can be invoked by the user exit, but you can replace this with one of your own. To do this, define your program in the PD of the region and set the **UserExitNumber** attribute to 25.

Although the **RemoteSysId** and **Remotename** attributes are used by the dynamic transaction routing user exit for routing the transactions, they are ignored when the transactions are run locally. If you set up the dynamic transaction routing user exit to allow a transaction to run locally, you need to define the local program.

This user exit is not invoked:

- When the transaction that is defined as dynamic is started in an intermediate application-owning region. CICS attempts to abend such transactions but cannot detect all attempts to “daisy chain” dynamic transactions. You must ensure that your dynamic transaction routing program does not “daisy chain” dynamic transactions.
- When a problem occurs because of lack of storage while the parameter list or standard header for the user exit is being built. This causes the transaction to abend.
- When the dynamic transaction is started in a Terminal-Owning Region by a command of the following type that is issued in an Application-Owning Region:

  ```
  EXEC CICS START TERM(yyyy) TRAN(xxxx)
  ```

  In this case, the user exit is not invoked and the transaction is routed back to the system where the request was issued.

Writing a dynamic transaction routing user exit

Because of the intercommunication aspects of dynamic transaction routing and the building of a parameter list for the user exit, a performance overhead is involved when a dynamic transaction routing user exit is used. However, the benefits of a well-written dynamic transaction routing user exit to the overall performance of several connected regions can far outweigh the negative effect.

The information below outlines some of the tasks that you might want your dynamic transaction routing user exit to perform, and describes how to implement them in your program in the most efficient way.

Note: Programs that are to be used for a CICS user exit are subject to some rules and conditions. These are described in the *CICS Administration Guide*.

How information is passed between CICS and the user exit

CICS passes information to the dynamic transaction routing exit by means of a parameter list. The parameter list contains both a standard header structure (cics_UE_Header_t), which is passed to all user exits, and a structure that is specific to dynamic transaction routing called (cics_UE014025_t). Both these structures are included in the header file (cicsue.h). Some of the data that is passed to the dynamic transaction routing program in the parameter list is:

- The system ID of the remote CICS region that is specified in the Transaction Definitions (TD)
- The name of the remote transaction
- A task-local user data area

You can write a dynamic transaction routing program that accepts these values, or changes them, or instructs CICS not to continue routing the transaction. The values that are used depend on the function that is to be performed; that is, some values might be ignored.

Throughout this section, references are made to the results of setting a parameter to a **null string**. A null string is a string that begins with a null character ’\0’. For example, a null string can be placed into the system ID parameter of the parameter list of user exit UE014025 as follows:

```
strcpy (UE_specificptr->UE_Dyrsysid, "");
```

All other strings that are returned by the user exit must also be terminated with a null character.

For a complete description of the parameters that are passed between CICS and the dynamic transaction routing program, see the *TXSeries for Multiplatforms Administration Reference*.

Changing the target CICS system

The parameter list that is passed to the dynamic transaction routing user exit initially contains the system ID of the default CICS region to which the transaction is to be routed. This is derived from the value of the **Remote system ID** attribute of the installed transaction definition. If the transaction definition does not specify a **Remote system ID** value, the system ID that is passed is that of the local CICS region.

The information that is passed to the dynamic transaction routing program in the user exit specific structure can be changed to have the transaction rerouted.

Changing the program name

When the dynamic transaction routing user exit is invoked, the **UE_Dyrprog** parameter contains the program name that is taken from the **Progname** attribute of the transactions TD. If no program name is defined in this attribute, a null string is passed to the user exit in the program name parameter. If you decide to route the transaction locally, you can use this field to specify an alternative program to be run. For example, if all remote CICS systems are unavailable and the transaction cannot be routed, you might want to run a program in the local CICS system to send an appropriate message to the user.

Note: If the dynamic transaction routing user exit returns a null string in the **UE_Dyrprog** parameter, CICS issues an abend and message, even if it the dynamic transaction routing user exit has chosen to route to a remote system.

Telling CICS whether to route or terminate a transaction

If you want a transaction to be routed, whether you have changed any values or not, you return **UE_Normal** to CICS with the return code. If you want to terminate the transaction with a message and an abend, you supply a return code of **UE_Term_Abend.** A further option (return code **UE_Terminate)** tells CICS to terminate the transaction with a message, but not an abend.

When you return control to CICS with return code **UE_Normal,** CICS first compares the returned system ID with the local system ID:

- If the system IDs are the same (or the returned system ID is a null string), CICS uses the program name that is specified in the parameter **UE_Dyrprog,** and executes the transaction locally.
- If the two system IDs are not the same, CICS uses the remote transaction name and remote system name that are specified by the user exit, and routes the transaction to the remote CICS system.

The dynamic transaction routing program is invoked again if:

- The routed transaction abends.
- The remote system is unavailable or not known.
- The routed transaction terminates successfully.

If the system is unavailable or not known

The dynamic transaction routing program is invoked again if the remote system name that you specify on the route selection call is not known or is unavailable, and you have specified that you want to retry the route request, by setting **UE_Dyrretry** to **UE_Yes.** When this happens, you have a choice of actions:

- You can instruct CICS not to continue trying to route the transaction, by issuing a return code of **UE_Term_Abend.** If the reason for the error is that the system is unavailable, CICS issues message ’ERZ1433E’ and abend ’A149’.
- You can tell CICS to terminate the transaction with only a message by returning a return code of **UE_Terminate.**
- You can change the system ID, and issue a return code of **UE_Normal** to try to route the transaction again. If you change the system ID, you might also need to supply a different remote transaction ID. You need to do this if, for example, the transaction has a different remote transaction name on each system.
- You can choose to run the transaction locally, by supplying the local system ID (or setting the system ID to a null string) and by supplying a program name. Note that the program name can be allowed to default to the program name that has been specified in the Transaction Definitions (TD) for the transaction.
- You can attempt to route to the same system again.

A count of the times that the routing program has been invoked for routing purposes for this transaction is passed in field **UE_Dyrcount.** In your program, you can set a limit on the value of this parameter to enable it to decide when to stop trying to route a particular transaction instance.

Invoking the user exit at the end of routed transactions

If you want your dynamic transaction routing program to be invoked again when the routed transaction has completed, you must set the **UE_Dyropter** field in the parameter list to **UE_Yes** before returning control to CICS, on the initial invocation of the user exit. You might want to do this, for example, if you are keeping a count of the number of transactions that are executing on a particular CICS system. However, during this reinvocation, the dynamic transaction routing program should update only its own resources.

Invoking the user exit on abend

If the routed transaction abends, the CICS system that started the transaction reinvoked the dynamic transaction routing program (if it had been requested by setting **UE_Dyropter** to **UE_Yes**). If a dynamic transaction abends, this exit point is called, whether it has been routed to a remote region or to a local region.

This exit point cannot retry the route request and should only update its own resources. This exit point can be used to keep information about the transaction that abended, which the program can use to influence where future transactions are routed.

Chapter 13. Asynchronous processing

Asynchronous processing is a special case of function shipping in which the shipped command starts a remote transaction. Unlike distributed transaction processing (DTP), the initiating and initiated transactions do not engage in synchronous communication. Instead, they are executed and terminated independently.

The interval control commands that can be used for asynchronous processing are:
- EXEC CICS START
- EXEC CICS CANCEL
- EXEC CICS RETRIEVE

For information about how to define remote transactions, see “Defining remote transactions for asynchronous processing” on page 120.

Security considerations

If you specify the **RSLCheck** attribute as **internal** or **external** in the Transaction Definitions (TD) entry for a transaction, CICS raises the NOTAUTH condition if the transaction attempts to issue an EXEC CICS command with the SYSID option specified. This prevents transactions from bypassing the local security check. Refer to “Security and function shipping” on page 141 for more information.

How to use asynchronous processing

This section describes how to initiate asynchronous processing and how to start and cancel remote transactions.

Two ways to initiate asynchronous processing

Asynchronous processing is initiated by the issuing of an EXEC CICS START command. Like other function shipping commands, the application program can ignore the location of the started transaction or can explicitly specify the system name.

- Implicitly specifying the location of the transaction

  A program can issue an EXEC CICS START command for a remote transaction as if the transaction is local; it does not need to specify the location of the requested resources. CICS resource definitions allow the system programmer to specify that the transaction is owned by a remote system. The request is routed to the system that is named by the Transaction Definitions (TD) **RemoteSysId** attribute. Refer to the following example:

  ```
          EXEC CICS START TRANID(TRN1)
  ```

  The TD entry for TRN1 would have a **RemoteSysId** defined if the transaction is to be run on the remote system. Otherwise, it is run on the local system.

- Explicitly specifying the location of the transaction

  In a resource-accessing command, an application program can use the SYSID parameter to specify the connection to the remote system that owns the transaction. The advantage is that any system, including the local system, can be named in the SYSID attribute. The decision whether to access a local or remote transaction can be taken at execution time, based on initialization parameters that are passed to the application program. Refer to the following example:

```
EXEC CICS START TRANID(TRN1)
                SYSID(SYS1)
```

CICS routes the start request to the region that is defined by the Communications Definitions (CD) SYS1. Any local TD for TRN1 is bypassed. It is assumed that a TD for TRN1 exists on SYS1.

Note: Asynchronous processing can also be initiated by using distributed transaction processing (DTP), as described in Chapter 14, "Distributed transaction processing (DTP)," on page 279.

Starting and canceling remote transactions

The EXEC CICS START command is used to queue a transaction initiation request in a remote CICS system, to which the command is function shipped. In the remote system, the mirror transaction is invoked to issue the EXEC CICS START command.

You can include time control information on the shipped EXEC CICS START command, by using the INTERVAL or TIME parameter. Before a command is shipped, CICS converts a TIME specification to a time interval that is relative to the local clock. The interval is the delay from receipt of the command on the remote system, ***not*** from the time of submitting the request.

The time interval, which is specified in the INTERVAL or TIME parameter of an EXEC CICS START command, is the time at which the remote transaction is to be initiated, ***not*** the time at which the request is to be shipped to the remote system.

An EXEC CICS START command that is shipped to a remote CICS system can be canceled, before the expiry of the time interval, by shipping an EXEC CICS CANCEL command to the same system. The EXEC CICS START command that is to be canceled is uniquely identified by the REQID value that is specified on the EXEC CICS START command and on the associated EXEC CICS CANCEL command. Any task can issue the EXEC CICS CANCEL command. It is not possible to cancel locally queued requests on CICS.

Passing information with the START command

The EXEC CICS START command has several parameters that enable information to be made available to the remote transaction when it is started. If the remote transaction is in a CICS system, the information is obtained by using the EXEC CICS RETRIEVE command. The information that can be specified is summarized in the following list:

- User data that is specified in the FROM parameter. This is the principal way in which data can be passed to the remote transaction.
- Temporary storage queue-named in the QUEUE parameter. This is an additional way of passing data. The queue can be in any CICS system that is accessible to the system on which the remote transaction is executed.
- A terminal name-specified in the TERMID parameter. This is the name of a terminal that is to be associated with the remote transaction when it is initiated. If a terminal is defined in the system that owns the remote transaction but is not owned by that system, an automatic transaction initiation (ATI) request is sent to the terminal-owning region (TOR), when the transaction is ready to run.
- A transaction name and an associated terminal name-specified in the RTRANSID and RTERMID parameters. These parameters enable the local transaction to specify transaction and terminal names for the remote transaction to use in an EXEC CICS START command to initiate a transaction in the local system.

Passing an APPLID with the EXEC CICS START command

If you have a transaction that can be started from several different systems, it is worthwhile to know where the transaction was initiated.

You can arrange for each invoking transaction to send its local APPLID as part of the user data in the EXEC CICS START command. The APPLID is accessed for the local system by using the EXEC CICS ASSIGN APPLID command. This APPLID is equivalent to the SNA LU name for the region and should be unique within the network. This is then known as the **RemoteLUName** in the remote region.

The SYSID is the local name for a connection. Using the **RemoteLUName**, the SYSID can be derived from EXEC CICS INQUIRE CONNECTION. To obtain a SYSID from a **RemoteLUName**:

```
EXEC CICS INQUIRE CONNECTION START;
while not end
    EXEC CICS INQUIRE CONNECTION(SYSID) NETNAME(RemoteLUName) NEXT;
    If NETNAME is the RemoteLUName we are looking for
  Then the CONNECTION value is the applicable SYSID;
     break;
EXEC CICS INQUIRE CONNECTION END;
```

Improving performance of intersystem START requests

In some inquiry-only applications, sophisticated error checking and recovery procedures might not be justified. When transactions make inquiries only, the terminal operator can retry an operation if no reply is received within a specific time. In such a condition, the number of data flows to and from the remote system can be substantially reduced by using the NOCHECK option on the EXEC CICS START command.

Deferred sending of START requests with the NOCHECK parameter

For EXEC CICS START commands with the NOCHECK parameter, CICS defers transmission of the request until one of the following events occurs:

- The transaction issues another function shipping request for the same system, or executes a sync point
- The transaction terminates with an implicit sync point
- An EXEC CICS START NOCHECK with PROTECT was specified for the same system
- When enough EXEC CICS START NOCHECK requests have accumulated on the local system to make sending them efficient

The first, or only, start request that is transmitted from a transaction to a remote system carries the begin-bracket indicator; the last, or only, request carries the end-bracket indicator. Also, if any of the start requests that are issued by the transaction specifies PROTECT, sync point coordination occurs after the last request. The sequence of requests is transmitted within a single SNA bracket and all the requests are handled by the same mirror task.

The NOCHECK parameter is always required when shipping of the EXEC CICS START command is queued pending the establishment of links with the remote system.

Local queuing of EXEC CICS START commands for remote transactions

When a local transaction is ready to ship an EXEC CICS START command, the intersystem facilities might be unavailable, either because the remote system is not active or because a connection cannot be established. The normal CICS action in these conditions is to raise the SYSIDERR condition.

This can be avoided by using the NOCHECK parameter, and arranging for CICS to queue the request locally and forward it when the required link is in service. Local queuing can be attempted for an EXEC CICS START NOCHECK command if the system name is valid but the system is not available. A system is defined as not available if the system is *out of service* when the request is initiated, or an attempt to initiate a session to the remote system fails. If either of the above conditions occurs, CICS queues an EXEC CICS START command for a remote transaction only if the following two conditions are met:
1. The SYSID parameter is not coded in the EXEC CICS START command, and
2. The local TD entry of the transaction specifies **LocalQ=yes**.

Local queuing should be used only for EXEC CICS START NOCHECK commands that represent time-independent requests. The delay that is implied by local queuing affects the time at which the request is actually started.

Including EXEC CICS START request delivery in a logical unit of work

The delivery of a start request to a remote system can be made part of a logical unit of work by specifying the PROTECT parameter on the EXEC CICS START command. The PROTECT parameter indicates that the remote transaction must not be scheduled until the initiating transaction has successfully sync pointed.

A successful sync point of the transaction guarantees that the start request has been delivered to the remote system or successfully locally queued. It does not guarantee that the remote transaction has completed, or even that it has been, or will be, initiated.

The started transaction

A CICS transaction that is initiated by an EXEC CICS START command can get the user data and other information that is associated with the request by using the EXEC CICS RETRIEVE command.

Started transaction satisfying multiple EXEC CICS START requests

In accordance with the normal rules for interval control, CICS queues a start request for a transaction that carries both user data and a terminal identifier if the transaction is already active and associated with the same terminal. During the waiting period, the active transaction can issue a further EXEC CICS RETRIEVE command to access the data that is associated with the queued request. Such an access automatically cancels the queued start request.

Thus, it is possible to design a transaction that can handle the data that is associated with multiple start requests. A long-running transaction can accept multiple inquiries from a terminal and ship start requests to a remote system. In the remote system, the first request causes a transaction to start. From time to time, the started transaction can issue EXEC CICS RETRIEVE commands to receive the data that is associated with further requests, the absence of further requests being indicated by the ENDDATA condition.

Overall application design should ensure that a transaction cannot get into a permanent wait state because of the absence of further start requests; for example, the transaction can be defined with a time-out interval.

Terminal acquisition by a remotely initiated CICS transaction

When a CICS transaction is started by a start request that names a terminal (TERMID), CICS makes the terminal available to the transaction as its principal facility. It makes no difference whether the start request was issued by a user transaction in the local CICS system or was received from a remote system and issued by the mirror transaction.

Application programming for asynchronous processing

This section describes application programming for asynchronous processing between CICS systems. The general information that is given for CICS transactions that use the EXEC CICS START or EXEC CICS RETRIEVE commands is applicable to communications between CICS and non-CICS LU 6.2 systems that support mapped conversations.

Starting a transaction on a remote region

You can start a transaction on a remote region by issuing an EXEC CICS START command just as though the transaction were a local one.

Generally, the transaction is defined as being remote. You can, however, name a remote region explicitly in the SYSID option. This use of the EXEC CICS START command is therefore essentially a special case of CICS function shipping.

Exceptional conditions for the EXEC CICS START command

The exceptional conditions that can occur as a result of issuing an EXEC CICS START request for a remote transaction, depend on whether or not you specify the NOCHECK option on the EXEC CICS START command.

If you specify NOCHECK, no conditions are raised as a result of the remote running of the EXEC CICS START command. SYSIDERR, however, still occurs if no link to the remote region is available, unless you have arranged for local queuing of start requests. Also, CICS abnormally terminates the local transaction if the remote mirror transaction that is associated with the EXEC CICS START command abnormally terminates.

If you do not specify NOCHECK, the raising of conditions follows the normal rules for EXEC CICS START.

Retrieving data associated with a remotely issued start request

You use the EXEC CICS RETRIEVE command to retrieve data that has been stored for a task as a result of a remotely-issued EXEC CICS START request. This is the only available method for accessing such data.

For your transaction, no distinction exists between data that is stored by a remote EXEC CICS START request and data that is stored by a local EXEC CICS START request, and the normal considerations for use of the EXEC CICS RETRIEVE command apply.

The *CICS Family: Interproduct Communication* provides much information about programming for asynchronous processing

Chapter 14. Distributed transaction processing (DTP)

DTP is one of the five ways that CICS allows processing to be split between intercommunicating systems. Only DTP allows two or more communicating application programs to run simultaneously in different systems and to pass data backward and forward between themselves; that is, to perform a synchronous conversation.

Of the five intercommunication facilities that CICS offers, DTP is the most flexible and powerful, but also the most complex.

The CICS family TCP/IP network protocol does not support DTP.

Concepts of distributed transaction processing (DTP)

DTP allows two or more partner programs that are in different systems to interact with each other. DTP enables a CICS transaction to communicate with one or more transactions that are running in different systems. A group of such connected transactions is called a *distributed process*.

The process can best be shown by discussing the operation of DTP between two CICS systems, CICSA and CICSB, where:

1. A transaction (TRAA) is initiated on CICSA; for example, by a terminal operator who is keying in a transaction ID and initial data.
2. To fulfill the request, the processing program X begins to execute on CICSA, probably reading initial data from files, perhaps updating other files and writing to print queues.
3. Without ending, program X asks CICSA to establish a conversation with another CICS system, CICSB. CICSA responds to the request.
4. Also without ending, program X sends a message across the conversation and asks CICSB to start a new transaction, TRBB. CICSB initiates transaction TRBB by invoking program Y.
5. Program X now sends and receives messages, including data, to and from program Y. Between sending and receiving messages, both program X and program Y continue normal processing completely independently. When the two programs communicate, their messages can consist of:
    a. Agreements about how to proceed with conversation or how to end it. For example, program X can tell program Y when it can transmit messages across the session. At any time, both programs must know the state of their conversation, and therfore, what actions are allowed. At any time, either system might have actual control of the conversation.
    b. Agreements to make permanent all changes that have made up to that point. This allows the two programs to *synchronize* changes. For example, a dispatch billing program on CICSA might want to commit delivery and charging for a stock item, but only when a warehouse program in CICSB confirms that it has successfully allocated the stock item and adjusted the inventory file accordingly.
    c. Agreements between CICSA and CICSB to cancel, rather than to make permanent, changes to data that have been made since a given point. Such a cancelation (or rollback) might occur when customers change their minds,

for example. Alternatively, it might occur because of uncertainty that has been caused by failure of the application, the system, the communication path, or the data source.

Although the two programs X and Y exist as independent units, it is clear that they are designed to work as one. Of course, DTP is not limited to pairs of programs. You can chain many programs together to distribute processing more widely. This is discussed later in the book.

In the overview of the process that is given above, the location of program Y has not been specified. Program X is a CICS program, but program Y need not be, because CICS can establish conversations with non-CICS partners. This is discussed in "Designing distributed processes" on page 283.

Conversations

Although several programs can be involved in a single distributed process, information transfer within the process is always between self-contained *communication pairs*. The exchange of information between a pair of programs is called a *conversation*. During a conversation, both programs are active; they send data to, and receive data from, each other. The conversation is two-sided but at any moment, each partner in the conversation has more or less control than the other. According to its level of control (known as its *conversation state*), a program has more or less choice in the commands that it can issue.

CICS supports conversations over SNA and CICS PPC TCP/IP. Slight differences exist between the two protocols. However, DTP programs can be written in such a way that they can work with both protocols. This is described in Appendix C, "Migrating DTP applications," on page 343.

Conversation states

Thirteen conversation states have been defined for CICS DTP. The set of states that is possible for a particular conversation depends on the synchronization level that is used. (The concepts of synchronization level are explained in "Maintaining data integrity" on page 282. The following table shows which conversation states are defined for each synchronization level. The conversation states YES and NO indicate whether the state is defined.

Table 46. Conversation states available for each synchronization level

State number	State name	Sync level 0	Sync level 1	Sync level 2
1	Allocated	Yes	Yes	Yes
2	Send	Yes	Yes	Yes
3	Pendreceive	Yes	Yes	Yes
4	Pendfree	Yes	Yes	Yes
5	Receive	Yes	Yes	Yes
6	Confreceive	No	Yes	Yes
7	Confsend	No	Yes	Yes
8	Conffree	No	Yes	Yes
9	Syncreceive	No	No	Yes
10	Syncsend	No	No	Yes
11	Syncfree	No	No	Yes
12	Free	Yes	Yes	Yes

Table 46. Conversation states available for each synchronization level (continued)

State number	State name	Sync level 0	Sync level 1	Sync level 2
13	Rollback	No	No	Yes

By using a special CICS command (EXTRACT ATTRIBUTES), or the STATE option on a DTP command, a program can obtain a value that indicates its own conversation state. CICS places such a value in a variable that is named by the program; the variable is sometimes referred to as a *state variable*. Knowing the current conversation state, the program then knows which commands are allowed. If, for example, a conversation is in **send**, the transaction can send data to the partner. (The transaction can take other actions instead, as indicated in the relevant state table.)

When a transaction issues a DTP command, that action can cause the conversation state to change. For example, a transaction can deliberately switch the conversation from **send** to **receive** by issuing a command that invites the partner to send data. When a conversation changes from one state to another, it is said to undergo a *state transition*. The state tables that are given in later chapters show how these transitions take place.

Not only does the conversation state determine which commands are allowed, but the state on one side of the conversation reflects the state that is on the other side. For example, if one side is in **send**, the other side is in either **receive**, **confreceive**, or **syncreceive**.

Distributed processes

A transaction can initiate other transactions, and therefore, conversations. In a complex process, a distinct hierarchy emerges, usually with the terminal-initiated transaction at the top. Consider the following scenario:

1. Transaction TRAA, in system CICSA, is initiated from a terminal.
2. Transaction TRAA requests a conversation with transaction TRBB to run in system CICSB.
3. Transaction TRBB in turn requests a conversation with transaction TRCC in system CICSC and transaction TRDD in system CICSD. Both transactions TRCC and TRDD request a conversation with the same transaction SUBR in system CICSE, therefore giving rise to two copies of SUBR.

Notice that each transaction can be invoked only by one partner transaction. However any transaction can invoke several remote transactions. The conversation that activates a transaction is called its *principal facility*. A conversation that is allocated by a transaction to activate another transaction is called its *alternate facility*. Therefore, a transaction can have only one principal facility, but several alternative facilities.

When a transaction initiates a conversation, it is the front-end transaction on that conversation. Its conversation partner is the back-end transaction on the same conversation. It is normally the front-end transaction that dominates, and determines the way the conversation goes. This style of processing is sometimes referred to as the *client/server* model (sometimes referred to as *master/slave*).

Alternatively, the front-end transaction and back-end transaction might switch control between themselves. This style of processing is called *peer-to-peer*. As the name implies, this model describes communication between equals. You are free to select whichever model you need when designing your application; CICS supports both.

Maintaining data integrity

DTP applications must be designed to manage the many error conditions that can arise when applications run in different systems. For example, one system might encounter a problem, or the communication link between the system might fail. CICS provides DTP commands and responses that help you recover from errors, and ensures that the two systems remain in step with each other. This use of the conversation is called *synchronization*.

Synchronization allows you to protect recoverable resources such as transient data queues and files, whether they are local or remote. Whatever goes wrong during the running of a transaction should not leave the associated resources in an inconsistent state.

An application program can cancel all changes that have made to recoverable resources since the last known consistent state. This process is called *rollback*. The physical process of recovering resources is called *backout*. The condition that exists when no loss of consistency occurs between distributed resources is called *data integrity*.

Sometimes you might need to backout changes to resources, although no error conditions have arisen. Consider an order entry system. While entering an order for a customer, an operator is told by the system that the customer’s credit limit would be exceeded if the order went through. Because it is of no use to continue until the customer is consulted, the operator presses a PF key to abandon the order. The transaction is programmed to respond by returning the data resources to the state that they were in at the start of the order transaction.

The point in a process where resources are declared to be in a known consistent state is called a *synchronization point*, often shortened to *sync point*. Sync points are implied at the beginning and end of a transaction. A transaction can define other sync points by program command. All processing between two sync points belongs to a *logical unit of work* (LUW). In a distributed process, this is also known as a *distributed unit of work*.

When a transaction issues a sync point command, CICS attempts to *commit* all changes to recoverable resources that are associated with that transaction. If this is successful, the transaction can no longer back out changes that have been made since the previous sync point. They have become irreversible. However, if the syncpoint command fails, the changes are backed out.

Although CICS can commit and backout changes to local and remote resources for you, this service must be paid for in performance. If the recovery of resources throughout a distributed process is not a problem (for example, in an inquiry-only application), you can use simpler methods of synchronization.

CICS defines three levels of synchronization for DTP conversations:
- Level 0: None
- Level 1: Confirm
- Level 2: Sync point

At synchronization level 0, no CICS support exists for synchronization of remote resources on connected systems. But it is still possible, under the control of the application to achieve some degree of synchronization by interchanging data, by using the SEND and RECEIVE commands.

At synchronization level 1, you can use special commands for communication between the two conversation partners. One transaction can *confirm* the continued presence and readiness of the other. Both transactions are responsible for preserving the data integrity of recoverable resources by issuing sync point requests at the appropriate times.

At synchronization level 2, all sync point requests are automatically propagated across multiple systems. CICS implies a sync point when it starts a transaction; that is, it initiates logging of changes to recoverable resources, but no control flows take place. CICS takes a sync point when one of the transactions terminates normally. One abending transaction causes all to rollback. The transactions themselves can initiate sync point or rollback requests. However, a sync point or rollback request is propagated to another transaction only when the originating transaction is in conversation with the other transaction, and synchronization level 2 has been selected.

Remember that sync point and rollback are not limited to any one conversation within a transaction. They are propagated on every conversation that is currently active at synchronization level 2.

For more information, see "Safeguarding data integrity" on page 296.

Designing distributed processes

This section discusses the issues that you must consider when designing distributed processes to run under APPC. These issues include structuring distributed processes and designing conversations.

It is assumed that you are already familiar with the issues that are involved in designing applications in single CICS systems. For guidance information, see the *TXSeries for Multiplatforms Application Programming Guide*.

Structuring distributed transactions

As with many design problems, designing a DTP application involves that handling of several conflicting objectives that must be carefully balanced against each other. These include performance, ease of maintenance, reliability, security, connectivity to existing functions, and recovery.

Avoiding performance problems

If performance is the highest priority, design your application so that data is processed as close to its source as possible. This avoids unnecessary transmission of data across the network. Alternatively, if processing can be deferred, you might want to consider batching data locally before transmitting.

To maintain performance across the intersystem connection, the conversation should be freed as soon as possible so that the session can be used by other transactions. In particular, avoid holding a conversation across a terminal wait.

Facilitating maintenance

To correct errors or to adapt to the evolving needs of an organization, distributed processes always need to be modified. Whether these changes are made by the original developers or by others, this task is likely to be easier if the distributed processes are relatively simple. So consider minimizing the number of transactions that are involved in a distributed process.

Going for reliability

If you are particularly concerned with reliability, consider minimizing the number of transactions in the distributed process.

Protecting sensitive data

If the distributed process is to handle security-sensitive data, you could place this data onto a single system. This means that only one of the transactions needs knowledge of how or where the sensitive data is stored.

Data conversion

For communication with non-TXSeries for Multiplatforms systems, data conversion might be required. When using DTP, it is the responsibility of the application to perform these data conversions.

Safeguarding data integrity

If it is important for you to be able to recover your data when things go wrong, design conversations for synchronization level 2, and keep the LUWs as small as possible. However, this is not always possible, because the size of an LUW is determined largely by the function that is being performed. Remember that CICS sync point processing has no information about the structure and purpose of your application. As an application designer, you must ensure that sync points are taken at the correct time and place, and to good purpose. If you do, error conditions are unlikely to lead to inconsistencies in recoverable data resources.

Figure 90 shows a temporary storage queue being transferred from system A to system B through a conversation at synchronization level 2. The numbers mark points at which you might consider taking a sync point.

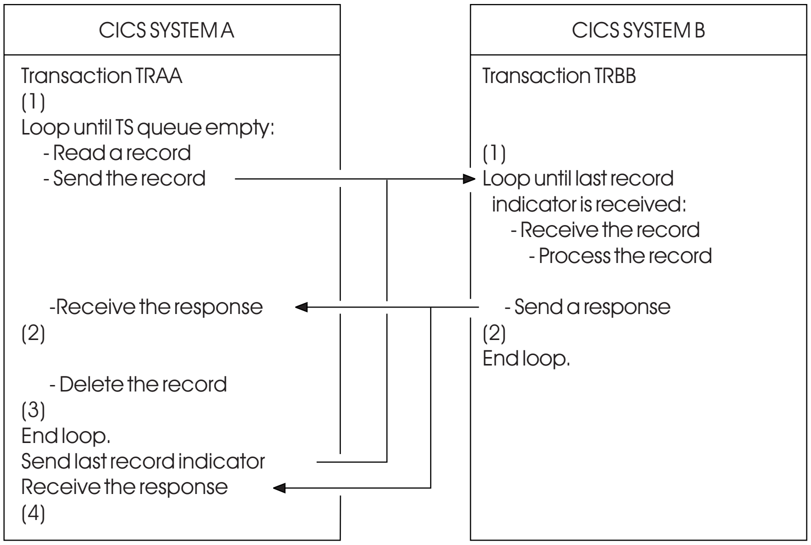


Figure 90. Transferring a temporary storage queue

Here are the relative advantages of taking a sync point at each of these points:

1. Because an LUW starts at point **(1)**, a sync point has no effect. If TRBB tries to take a sync point without having first issued a command to receive data, it will be abended because it is not valid to perform sync point with a synchronization level 2 conversation in **receive** state.
2. A sync point at point **(2)** causes CICS to commit a record in system B before it has been deleted from system A. If either system (or the connection between them) fails before the distributed process is completed, data might be duplicated.
3. Because minimum processing is needed before resources are committed, point **(3)** might be a safe place to take a sync point if the queue is long or the records are large. However, performance might be poor because many sync points are likely to be taken.
4. If you take a sync point only at point **(4)**, a failure before this point means that all data that is sent will have to be retransmitted. A distributed process that sync points only at this stage completes more quickly than one that sync points at point (3), provided that no failure occurs. However, it takes longer to recover. If more than two systems are involved in the process, this problem is made worse.

Remember that having too many conversations within one distributed transaction complicates error recovery. A complex structure might sometimes be unavoidable, but usually it means that the design could be improved by simplifying the structure of the distributed transaction.

An LUW must be recoverable for the whole process of which it forms a part. All changes that are made by both partners in every conversation must be backed out if the LUW does not complete successfully. Sync points are not arbitrary divisions, but must reflect the functions of the application. LUWs must be designed to preserve consistent resources so that when a transaction fails, **all** resources are restored to their correct state.

Designing conversations

When the overall structure of the distributed process has been decided, you can then start to design individual conversations. Designing a conversation involves deciding which functions to put into the front-end transaction and into the back-end transaction, and deciding what should be in a distributed unit of work. So you have to make decisions about how to subdivide the work that is to be done for your application.

Because a conversation involves transferring data between two transactions, to function correctly, each transaction must know what the other intends. For example, little advantage is gained by the front-end transaction’s sending data if all the back-end transaction is designed to do is print the weekly sales report. You must therefore consider each front-end and back-end transaction pair as one software unit.

The sequences of commands that you can issue on a conversation are governed by a protocol that is designed to ensure that commands are not issued in inappropriate conditions. The protocol is based on the concept of several conversation states. A conversation state applies only to one side of a single conversation and not to a transaction as a whole. In each state, are several commands that might reasonably be issued. The command itself, together with its outcome, might cause the conversation to change from one state to another.

To determine the conversation state, you can use either the STATE option on a command, or the EXTRACT ATTRIBUTES command. For the state values that are returned by different commands, see the *TXSeries for Multiplatforms Application Programming Guide*.

Note: You can also determine the state of a conversation by examining the EIB values after each DTP command. However, it is more efficient to use explicit STATE values.

When a conversation changes state, it is said to have undergone a **state transition**, which generally makes a different set of commands available. The available commands and state transitions are shown in a series of state tables. Which state table you use depends on the synchronization level chosen.

For more information, see "How to use the state tables" on page 331.

Writing programs for CICS DTP

CICS enables DTP by using the LU 6.2 APPC mapped conversations commands. The following sections describe how to code a simple DTP program:
- "Conversation initiation and the front-end transaction"
- "Back-end transaction initiation" on page 289
- "Transferring data on the conversation" on page 291
- "Communicating errors across a conversation" on page 295
- "Safeguarding data integrity" on page 296
- "Ending the conversation" on page 298
- "Checking the outcome of a DTP command" on page 300

You need to manage the changing conversation states of your program. A description of how you test the state, and a complete list of all the conversation states, is given in:
- "Testing the conversation state" on page 303
- Appendix B, "The conversation state tables," on page 331

The APPC commands are summarized in:
- "Summary of CICS commands for APPC mapped conversations" on page 304

Conversation initiation and the front-end transaction

The front-end transaction is responsible for acquiring a conversation, specifying the conversation characteristics, and requesting the startup of the back-end transaction in the remote system.

This section describes the following topics:
- Allocating a conversation
- Using ATI to allocate a conversation identifier (CONVID)
- Connecting the partner transaction
- Initial data for the back-end transaction

Allocating a conversation

Initially, no conversation occurs, and therefore no conversation state. By issuing an EXEC CICS ALLOCATE command, the front-end transaction acquires a *conversation identifier* (CONVID) for a new conversation.

The RESP value that is returned from ALLOCATE should be checked to ensure that a CONVID has been allocated. If the CONVID is successfully allocated, (DFHRESP(NORMAL)), the conversation is in *allocated state* (state 1) and the CONVID in EIBRSRCE must be saved immediately. The SYSID option contains the name of the Communications Definitions (CD) entry for the remote system.

The CONVID must be used in subsequent commands for this conversation.

A full description of the EXEC CICS ALLOCATE command can be found in the *TXSeries for Multiplatforms Application Programming Reference*. Figure 91 shows an example of an EXEC CICS ALLOCATE command:

```
*    ...
DATA DIVISION.
WORKING-STORAGE SECTION.
*    ...
 01  FILLER.
     02  WS-CONV         PIC X(4).
     02  WS-RESP         PIC S9(8) COMP.
     02  WS-STATE        PIC S9(8) COMP.
     02  WS-SYSID        PIC X(4) VALUE 'SYSB'.
     02  WS-PROC         PIC X(32) VALUE 'DTP2'
     02  WS-LEN-PROCN    PIC S9(5) COMP VALUE +4.
     02  WS-SYNC-LVL     PIC S9(5) COMP VALUE +2.
*    ...
 PROCEDURE DIVISION.
*    ...
     EXEC CICS ALLOCATE SYSID(WS-SYSID) RESP(WS-RESP) END-EXEC.
     IF WS-RESP = DFHRESP(NORMAL)
     THEN MOVE EIBRSRCE TO WS-CONVID
     ELSE
*       ...  No session allocated.  Examine RESP code.
     END-IF.
*    ...
     EXEC CICS CONNECT PROCESS CONVID(WS-CONV) STATE(WS-STATE)
                               RESP(WS-RESP) PROCNAME(WS-PROC)
                               PROCLENGTH(WS-LEN-PROCN)
                               SYNCLEVEL(WS-SYNC-LVL)
     END-EXEC.
     IF WS-RESP = DFHRESP(NORMAL)
     THEN
*       ...  No errors.  Check EIB flags.
     ELSE
*       ...  Conversation not started.  Examine RESP code.
     END-IF.
```

Figure 91. Starting a conversation at synchronization level 2

Using ATI to allocate a conversation identifier (CONVID)

Front-end transactions are often initiated from terminals, but they can be started by automatic transaction initiation (ATI). ATI can start a transaction with a conversation as its principal facility.

A transaction can be started automatically either by an EXEC CICS START command, or by a transient-data trigger. For an EXEC CICS START request, the TERMID option is used. If a SYSID is specified in the TERMID option, the transaction is started with a conversation as its principal facility.

For a transient-data trigger, the following Transient Data Definition (TDD) attributes are used:

- A **FacilityType** of **system** is specified
- The **FacilityId** specifies the SYSID of the remote system

A transaction that is started in either of these two ways already has an conversation allocated. Although this conversation is the principal facility of the started transaction, it is handled in the same way as a terminal-attached transaction handles a secondary facility. The started transaction completes the initiation of a conversation by issuing the EXEC CICS CONNECT PROCESS command, as described below.

Connecting the partner transaction

When the front-end transaction has acquired a CONVID, the next step is to initiate the partner transaction. The state tables show that, in the *allocated state* (state 1), one of the commands that is available is EXEC CICS CONNECT PROCESS. This command allows the conversation characteristics to be specified and attaches the required back-end transaction. It should be noted that the results of the EXEC CICS CONNECT PROCESS are placed into the send buffer and are not sent immediately to the partner system. Transmission occurs later, when the conversation is used. If the remote system failed to start, the TERMERR condition is returned on a later verb.

A successful EXEC CICS CONNECT PROCESS causes the conversation to switch to *send state* (state 2). The following program fragment shows an example of an EXEC CICS CONNECT PROCESS command. The PROCNAME option contains what is referred to in SNA terminology as the *transaction program name (TPN)*. If the remote system is CICS, PROCNAME should be the four-character transaction identifier that is configured in the remote system for the back-end program.

```
 *     ...
       EXEC CICS CONNECT PROCESS CONVID(WS-CONV) STATE(WS-STATE)
                                 RESP(WS-RESP) PROCNAME(WS-PROC)
                                 PROCLENGTH(WS-LEN-PROCN)
                                 SYNCLEVEL(WS-SYNC-LVL)
       END-EXEC.
```

For a full description of the EXEC CICS CONNECT PROCESS command, refer to the *TXSeries for Multiplatforms Application Programming Reference*.

Initial data for the back-end transaction

While connecting the back-end transaction, the front-end transaction can send initial data to it. This type of data, called *program initialization parameters* (PIPs), is placed into specially formatted structures and is specified on the EXEC CICS CONNECT PROCESS command. The PIPLIST (along with PIPLENGTH) option of the EXEC CICS CONNECT PROCESS command is used to send PIPs to the back-end transaction.

When using Communications Server for AIX

Note: Current AIX SNA limitations specify that only 16 fields of 64 characters each of PIP data can be transmitted. That data must not contain space characters. If the data that is transmitted does not conform to this, the CONNECT PROCESS command apparently succeeds, but the next DTP command returns TERMERR.

To examine any PIPs that are received, the back-end transaction uses the EXEC CICS EXTRACT PROCESS command.

PIP data is used only by the two connected transactions and not by the CICS systems. The support of PIP data is optional for APPC systems. If PIP data is not supported by the partner system, the TERMERR is returned on a later verb with EIBERRCD set to 10086032.

The PIP data must be formatted into one or more subfields in accordance with the CICS-architected rules. The content of each subfield is defined by the application developer. You should format PIP data as follows:

```
L1   rr   PIP1  L2   rr  PIP2 ......  Ln   rr  PIPn
```

where Ln is a halfword binary integer that specifies the length of the subfield, and rr represents a reserved halfword. The length includes the length field itself and the length of the reserved field; that is, Ln = (length of PIPn + 4).

The PIPLENGTH option must specify the total length of the PIP list and must be in the range 4 through 32763.

Back-end transaction initiation

The back-end transaction is initiated as a result of the front end transaction's EXEC CICS CONNECT PROCESS command. Initially, the back-end transaction should determine the CONVID. This is not strictly necessary because the conversation is the back-end transaction's principal facility. Therefore, the CONVID parameter is optional for DTP commands on this conversation. However, the CONVID is useful for audit trails. Also, if the back-end transaction is involved in more than one conversation, by always specifying the CONVID option, you improve program readability and problem determination.

Figure 92 on page 290shows a fragment of a back-end transaction that does obtain the conversation identifier. Although the example uses the EXEC CICS ASSIGN command for this purpose, a simpler way would be to access the information in EIBTRMID.

```
      *    ...
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      *    ...
       01  FILLER.
           02  WS-CONVID       PIC X(4).
           02  WS-STATE        PIC S9(7) COMP.
           02  WS-SYSID        PIC X(4) VALUE 'SYSB'.
           02  WS-PROC         PIC X(32) VALUE 'BBBB'.
           02  WS-LEN-PROCN    PIC S9(5) COMP VALUE +4.
           02  WS-SYNC-LVL     PIC S9(5) COMP VALUE +2.
      *    ...
       01  FILLER.
           02  WS-RECORD       PIC X(100).
           02  WS-MAX-LEN      PIC S9(5) COMP VALUE +100.
           02  WS-RCVD-LEN     PIC S9(5) COMP VALUE +0.
      *    ...
       PROCEDURE DIVISION.
      *    ...
           EXEC CICS ASSIGN FACILITY(WS-CONVID) END-EXEC.
      *    ...
      *    Extract the conversation characteristics.
      *
           EXEC CICS EXTRACT PROCESS PROCNAME(WS-PROC)
                                     PROCLENGTH(WS-LEN-PROCN)
                                     SYNCLEVEL(WS-SYNC-LVL)
           END-EXEC.
      *    ...
      *    Receive data from the front-end transaction.
      *
           EXEC CICS RECEIVE CONVID(WS-CONVID) STATE(WS-STATE)
                             INTO(WS-RECORD) MAXLENGTH(WS-MAX-LEN)
                             NOTRUNCATE LENGTH(WS-RCVD-LEN)
           END-EXEC.
      *
      *    ... Check outcome of EXEC CICS RECEIVE.
      *    ...
```

Figure 92. Startup of a back-end APPC mapped transaction at synchronization level 2

The back-end transaction can also retrieve its transaction name by issuing the EXEC CICS EXTRACT PROCESS command. In the example that is shown in Figure 92, CICS places the transaction name in WS-PROC and the length of the name in WS-LEN-PROCN.

Note: The APPC architecture, which is described in *Transaction Programmer's Reference for LUTYPE6.2*, GC30-3084, states that the maximum length for the PROCNAME field is 64 bytes and that implementation might limit this field to less than 64 bytes. TXSeries for Multiplatforms supports up to 32 bytes of data for PROCNAMEs. The PROCNAME field is space padded to 32 bytes.

With the EXEC CICS EXTRACT PROCESS, the back-end transaction can also retrieve the synchronization level at which the conversation was started. In the example, CICS places the synchronization level in WS-SYNC-LVL.

The EXEC CICS ASSIGN and the EXEC CICS EXTRACT PROCESS commands are discussed here to give you some idea of what you can do in the back-end transaction. They are not essential.

The back-end transaction starts in *receive state* (state 5), and must issue an EXEC CICS RECEIVE command. By doing this, the back-end transaction receives whatever data the front-end transaction has sent and allows CICS to raise EIB flags and change the conversation state to reflect any request that the front-end transaction has issued.

The back-end transaction fails to start

It is possible that the back-end transaction fails to start. However, APPC contains a transmission delay mechanism that informs the front-end transaction of the failure when the conversation has been active long enough for responses from the back-end system to have been received. The front-end transaction is informed of this by way of a TERMERR condition in response to a DTP command. EIBERR, EIBFREE, and EIBERRCD are set (see “Checking the outcome of a DTP command” on page 300 for the possible values of EIBERRCD).

Before sending data, the front-end transaction should determine whether the back-end transaction has started successfully. One way of doing this is to issue an EXEC CICS SEND CONFIRM command directly after the EXEC CICS CONNECT PROCESS command. This causes the front-end transaction to suspend until the back-end transaction has responded or the back-end transaction has sent the failure notification that is described above. EXEC CICS SEND CONFIRM is discussed in “Safeguarding data integrity” on page 296.

Transferring data on the conversation

This section discusses how to pass data between the front- and back-end transactions. In this section:

- “Sending data to the partner transaction” explains how to send data
- “Switching from sending to receiving data” on page 292 describes how to switch from sending to receiving data
- “Receiving data from the partner transaction” on page 293 explains how to receive data

Also contained in this section is a program fragment that shows the commands that are described and the suggested response code checking.

Sending data to the partner transaction

The EXEC CICS SEND command is valid only in *send state* (state 2). Because a successful simple EXEC CICS SEND leaves the conversation in *send state* (state 2), it is possible to issue several successive sends. The data from the simple EXEC CICS SEND command is initially stored in a local CICS buffer, which is “flushed” either when this buffer is full or when the transaction requests transmission. The transaction can request transmission either by using an EXEC CICS WAIT CONVID command, or by using the WAIT option on the EXEC CICS SEND command. The reason why data transmission is deferred is to reduce the number of calls to the network. Data can be buffered by the network layers between CICS. Therefore, use of the WAIT command does not guarantee that the partner transaction immediately receives the data.

An example of a simple EXEC CICS SEND command is shown in Figure 93 on page 292. For a full description of this command, see the *TXSeries for Multiplatforms Application Programming Reference*.

```
*    ...
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*    ...
 01  FILLER.
     02  WS-CONVID       PIC X(4).
     02  WS-STATE        PIC S9(7) COMP.
*    ...
 01  FILLER.
     02  WS-SEND-AREA    PIC X(70).
     02  WS-SEND-LEN     PIC S9(5) COMP VALUE +70.
*    ...
 01  FILLER.
     02  WS-RCVD-AREA    PIC X(100).
     02  WS-MAX-LEN      PIC S9(5) COMP VALUE +100.
     02  WS-RCVD-LEN     PIC S9(5) COMP VALUE +0.
*    ...
 PROCEDURE DIVISION.
*    ...
     EXEC CICS SEND CONVID(WS-CONVID) STATE(WS-STATE)
                    FROM(WS-SEND-AREA) LENGTH(WS-SEND-LEN)
     END-EXEC.
*    ... Check outcome of SEND.
*    ...
*
     EXEC CICS SEND CONVID(WS-CONVID) STATE(WS-STATE)
                    INVITE WAIT
     END-EXEC.
*    ...
*    Receive data from the partner transaction.
*
     EXEC CICS RECEIVE CONVID(WS-CONVID) STATE(WS-STATE)
                       INTO(WS-RCVD-AREA) MAXLENGTH(WS-MAX-LEN)
                       NOTRUNCATE LENGTH(WS-RCVD-LEN)
     END-EXEC.
*
*    ... Check outcome of EXEC CICS RECEIVE.
*    ...
```

Figure 93. Transferring data on a conversation at synchronization level 2

Switching from sending to receiving data

The column for *send state* (state 2) in the state tables (see Appendix B, “The conversation state tables,” on page 331) shows that several ways of switching from *send state* (state 2) to *receive state* (state 5) are possible.

One possibility is to use an EXEC CICS RECEIVE command. The state tables show that CICS supplies the INVITE and WAIT when an EXEC CICS SEND is followed immediately by an EXEC CICS RECEIVE.

Another possibility is to use an EXEC CICS SEND INVITE command. The state tables show that after EXEC CICS SEND INVITE, the conversation switches to **pendreceive** (state 3). The column for state 3 shows that an EXEC CICS WAIT CONVID command switches the conversation to *receive state* (state 5).

Still another possibility is to specify the INVITE and WAIT options on the EXEC CICS SEND command. The state tables show that after EXEC CICS SEND INVITE WAIT, the conversation switches to *receive state* (state 5).

Figure 94 shows the response-testing sequence after an EXEC CICS SEND INVITE WAIT with the STATE option.

```
*    ...
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*    ...
 01  FILLER.
     02  WS-RESP         PIC S9(7) COMP.
     02  WS-STATE        PIC S9(7) COMP.
*    ...
 PROCEDURE DIVISION.
*    ...
* Check return code from SEND INVITE WAIT
     IF WS-RESP = DFHRESP(NORMAL)
     THEN
*       ...  Request successful
        IF EIBERR = LOW-VALUES
        THEN
*          ...  No errors, check state
           IF WS-STATE = DFHVALUE(RECEIVE)
           THEN
*             ...  SEND OK, continue processing
           ELSE
*             ...  Logic error, should never happen
           END-IF
        ELSE
*          ...  Error indicated
           EVALUATE WS-STATE
             WHEN DFHVALUE(ROLLBACK)
*                 ...  ROLLBACK received
             WHEN DFHVALUE(RECEIVE)
*                 ...  ISSUE ERROR received, reason in EIBERRCD
             WHEN OTHER
*                 ...  Logic error, should never happen
           END-EVALUATE
        END-IF
     ELSE
*       ...  Examine RESP code for source of error.
     END-IF.
```

Figure 94. Checking the outcome of an EXEC CICS SEND INVITE WAIT command

For more information about response testing, see “Checking the outcome of a DTP command” on page 300.

Receiving data from the partner transaction

The EXEC CICS RECEIVE command is used to receive data from the connected partner. The rows in the state tables for the EXEC CICS RECEIVE command show the EIB fields that should be tested after an EXEC CICS RECEIVE command is issued. In addition to showing which field should be tested, the state tables also show the sequence in which the tests should be made.

As an alternative to testing the EIB fields, it is possible to test the resulting conversation state; this is shown in the following figure. The conversation state can be meaningfully tested only after you have issued a command with the STATE option, or when you use the EXTRACT ATTRIBUTES command. For more information about response testing, see “Checking the outcome of a DTP command” on page 300.

For information about testing the conversation state, see “Testing the conversation state” on page 303.

Figure 95 shows the response-testing and state-testing sequence.

Note: In the same way as it is possible to send the INVITE, LAST, and CONFIRM commands with data, it is also possible to receive them with data. It is also possible to receive a sync point request with data. However, EXEC CICS ISSUE ERROR, EXEC CICS ISSUE ABEND, and conversation failure are never received with data.

```
 WORKING-STORAGE SECTION.
*    ...
 01  FILLER.
     02  WS-RESP         PIC S9(8) COMP.
     02  WS-STATE        PIC S9(8) COMP.
*    ...
 PROCEDURE DIVISION.
*    ...
* Check return code from RECEIVE
     IF WS-RESP = DFHRESP(EOC)
     OR WS-RESP = DFHRESP(NORMAL)
     THEN
*       ...  Request successful
        IF EIBERR = LOW-VALUES
        THEN
*          ...  No errors, check state
           EVALUATE WS-STATE
             WHEN DFHVALUE(SYNCFREE)
 *                 ...  Partner issued SYNCPOINT and LAST
             WHEN DFHVALUE(SYNCRECEIVE)
 *                 ...  Partner issued SYNCPOINT
              WHEN DFHVALUE(SYNCSEND)
 *                 ...  Partner issued SYNCPOINT and INVITE
              WHEN DFHVALUE(CONFFREE)
 *                 ...  Partner issued CONFIRM and LAST
              WHEN DFHVALUE(CONFRECEIVE)
 *                 ...  Partner issued CONFIRM
              WHEN DFHVALUE(CONFSEND)
 *                 ...  Partner issued CONFIRM and INVITE
              WHEN DFHVALUE(FREE)
 *                 ...  Partner issued LAST or FREE
              WHEN DFHVALUE(SEND)
 *                 ...  Partner issued INVITE
              WHEN DFHVALUE(RECEIVE)
 *                 ...  No state change.  Check EIBCOMPL.
              WHEN OTHER
 *             ...  Logic error, should never happen
            END-EVALUATE.
         ELSE
 *          ...  Error indicated
            EVALUATE WS-STATE
              WHEN DFHVALUE(ROLLBACK)
 *                 ...  ROLLBACK received
              WHEN DFHVALUE(RECEIVE)
 *                 ...  ISSUE ERROR received, reason in EIBERRCD
              WHEN OTHER
 *                 ...  Logic error, should never happen
            END-EVALUATE
         END-IF
      ELSE
 *       ...  Examine RESP code for source of error
      END-IF.
```

Figure 95. Checking the outcome of an EXEC CICS RECEIVE command

For a full description of the EXEC CICS RECEIVE command, refer to the *TXSeries for Multiplatforms Application Programming Reference*.

The EXEC CICS CONVERSE command

The EXEC CICS CONVERSE command combines the functions EXEC CICS SEND INVITE WAIT and EXEC CICS RECEIVE. This command is useful when one transaction needs a response from the partner transaction in order to continue processing. The use of EXEC CICS CONVERSE instead of EXEC CICS SEND INVITE WAIT allows CICS to improve network performance.

Refer to the *TXSeries for Multiplatforms Application Programming Reference* for a full description of the EXEC CICS CONVERSE command.

Communicating errors across a conversation

The APPC mapped API provides commands to enable transactions to pass error notification across a conversation. Three coomands are possible, depending on the severity of the error. The most severe, EXEC CICS ISSUE ABEND, causes the conversation to terminate abnormally, and is described in “Emergency termination of a conversation” on page 299. The other two commands are described below.

Requesting invite from the partner transaction

If a transaction is receiving data on a conversation and wants to send, it can use the EXEC CICS ISSUE SIGNAL command to request that the partner transaction does an EXEC CICS SEND INVITE. When the EXEC CICS ISSUE SIGNAL request is received, EIBSIG=X’FF’ and the SIGNAL condition are raised. It should be noted that on receipt of SIGNAL, a transaction is *not* obliged to issue EXEC CICS SEND INVITE. For a full description of the EXEC CICS SEND INVITE command, see the *TXSeries for Multiplatforms Application Programming Reference*.

Demanding invite from the partner transaction

If a transaction needs to send an immediate error notification to its partner program but the conversation is in receive (state 5), it can use the ISSUE ERROR command to try to switch the conversation state to send (state 2). When the partner application receives ISSUE ERROR, the EIBERR flag is X’FF’, the EIBERRCD field begins X’0889’, and the conversation state is set to receive (state 5). This error condition cannot be processed by HANDLE CONDITION (or RESP).

ISSUE ERROR can be called in send (state 2) and can also be used to give an error response to a SEND CONFIRM request that the partner program has made. However it should not be used in response to an ISSUE PREPARE, SYNCPOINT or SYNCPOINT ROLLBACK.

If an ISSUE ERROR command is sent when the conversation is in receive (state 5), all incoming data from the partner program is purged until either the remote system acknowledges receipt of the ISSUE ERROR and reports it to the partner application by setting EIBERR=X’FF’ and EIBERRCD=X’0889’, or the partner application stops sending data and issues a command such as SYNCPOINT, SYNCPOINT ROLLBACK, SEND LAST WAIT, FREE, ISSUE ERROR, or ISSUE ABEND.

Always ensure that the EIB values are checked after ISSUE ERROR is called, because it is possible to receive a response from the partner program that prevents the conversation from switching to send (state 2). The response from ISSUE ERROR can be checked as follows:

```
*    ...
     DATA DIVISION.
     WORKING-STORAGE SECTION.
    *   ...
     01  FILLER.
```

```
      02  WS-RESP      PIC S9(7) COMP.
      02  WS-STATE     PIC S9(7) COMP.
 *   ...
 * Check the response from ISSUE ERROR.
      IF WS-RESP = DFHRESP(NORMAL)
      THEN
 *    ... Request successful
          IF EIBERR = LOW-VALUES
          THEN
 *        ... No errors, check state
              EVALUATE WS-STATE
                  WHEN DFHVALUE(SEND)
 *                ... ISSUE ERROR worked.  Use CONVERSE to
 *                ... send an appropriate error message and
 *                ... receive a reply.
                  WHEN DFHVALUE(FREE)
 *                ... Partner sent SEND LAST (or ISSUE ABEND
 *                ... while in send (state 2) and this has been
 *                ... partially purged by the local program calling
 *                ... ISSUE ERROR in receive (state 5)).
                  WHEN OTHER
 *                ... Logic error, should never happen.
              END-EVALUATE
          ELSE
 *        ... Errors received
              EVALUATE WS-STATE
                  WHEN DFHVALUE(RECEIVE)
*                ... ISSUE ERROR received from partner.
                  WHEN DFHVALUE(ROLLBACK)
 *                ... Partner sent SYNCPOINT ROLLBACK.
                  WHEN OTHER
 *                ... Logic error, should never happen.
              END-EVALUATE
      ELSE
 *    ... RESP indicates a failure.  This could be a TERMERR
 *    ... caused by the partner abending or calling ISSUE ABEND.
```

For a full description of the EXEC CICS ISSUE ERROR command, see the *TXSeries for Multiplatforms Application Programming Reference*.

Safeguarding data integrity

Use the following CICS synchronization commands to safeguard data integrity across connected transactions:

Table 47. CICS synchronization commands for use across transactions

Synchronization level	**Commands**
0 - none	None
1 - confirm	EXEC CICS SEND CONFIRM EXEC CICS ISSUE CONFIRMATION
2 - sync point	EXEC CICS SEND CONFIRM EXEC CICS ISSUE CONFIRMATION EXEC CICS SYNCPOINT EXEC CICS ISSUE PREPARE EXEC CICS SYNCPOINT ROLLBACK

How to synchronize a conversation using CONFIRM commands

A confirmation exchange affects a single specified conversation and invokes only two commands:

1. EXEC CICS SEND CONFIRM: The conversation that is in **send** (state 2) issues an EXEC CICS SEND CONFIRM command, causing a request for confirmation to be sent to the partner transaction. The transaction suspends awaiting a response.
2. EXEC CICS ISSUE CONFIRMATION: The partner transaction receives a request for confirmation. It can then respond positively by issuing an EXEC CICS ISSUE CONFIRMATION command. Alternatively, it can respond negatively by using the EXEC CICS ISSUE ERROR or EXEC CICS ISSUE ABEND commands.

Requesting confirmation

The CONFIRM option of the EXEC CICS SEND command flushes the conversation send buffer. This causes a transmission to occur. When the conversation is in **send** (state 2), you can send data with the EXEC CICS SEND CONFIRM command. You can also specify either the INVITE or the LAST option.

The **send** (state 2) column of the synchronization level 1 state table (see "Synchronization level 1 conversation state table" on page 335) shows what happens for the possible combinations of the CONFIRM, INVITE, and LAST options. After an EXEC CICS SEND CONFIRM command, without the INVITE or LAST options, the conversation remains in a **send** state. If the INVITE option is used, the conversation switches to **receive** (state 5). If the LAST option is used, the conversation switches to **free** (state 12).

A similar effect to EXEC CICS SEND LAST CONFIRM can be achieved by using the following command sequence:

```
EXEC CICS SEND LAST EXEC CICS SEND CONFIRM
```

Note from the state tables that the EXEC CICS SEND LAST command puts the conversation in **pendfree** (state 4), so data cannot be sent when an EXEC CICS SEND CONFIRM command is used as shown in the above example.

The form of command that is used depends on how the conversation is to continue if the required confirmation is received. However, the response from EXEC CICS SEND CONFIRM must always be checked.

Receiving and replying to a confirmation request

On receipt of a confirmation request, the EIB and conversation state is set depending on the request that the partner transaction issues. These, together with the contents of the EIBCONF, EIBRECV, and EIBFREE fields, are shown in the following table:

Table 48. EIB and conversation state request responses

Command issued in reply by partner transaction	On receipt of response - Conversation State	On receipt of response - EIBCONF	On receipt of response - EIBRECV	On receipt of response - EIBFREE
EXEC CICS SEND CONFIRM	confreceive (state 6)	X'FF'	X'FF'	X'00'
EXEC CICS SEND INVITE CONFIRM	confsend (state 7)	X'FF'	X'00'	X'00'
EXEC CICS SEND LAST CONFIRM	conffree (state 8)	X'FF'	X'00'	X'FF'

You can reply in three ways:

1. Reply positively with an EXEC CICS ISSUE CONFIRMATION command.
2. Reply negatively with an EXEC CICS ISSUE ERROR command. This reply puts the conversation into **send** state, regardless of the partner transaction request.
3. Abnormally end the conversation with an EXEC CICS ISSUE ABEND command. This makes the conversation unusable and an EXEC CICS FREE command must be issued immediately.

Checking the response to EXEC CICS SEND CONFIRM

After issuing EXEC CICS SEND INVITE CONFIRM or EXEC CICS SEND LAST CONFIRM, it is important to test EIBERR to determine the partner's response. The following table shows how the partner's response is indicated by EIB flags and the conversation states:

Table 49. Indications of partner response

Command issued in reply by partner transaction	On receipt of response - Conversation State	On receipt of response - EIBERR	On receipt of response - EIBFREE
EXEC CICS ISSUE CONFIRMATION	Dependent on the original EXEC CICS SEND INVITE CONFIRM or EXEC CICS SEND LAST CONFIRM request	X'00	X'00
EXEC CICS ISSUE ERROR	Receive (state 5)	X'FF'	X'00
EXEC CICS ISSUE ABEND	Free (state 12)	X'FF'	X'FF'

If EIBERR=00, the partner has replied EXEC CICS ISSUE CONFIRMATION.

If EIBERR=X'FF' and the first two bytes of EIBERRCD=X'0889', the partner replied EXEC CICS ISSUE ERROR. When the partner replies EXEC CICS ISSUE ERROR in response to EXEC CICS SEND LAST CONFIRM, the LAST option is ignored and the conversation is not terminated. The conversation state is switched to **receive**.

If the partner replies EXEC CICS ISSUE ABEND, the TERMERR condition is raised. In addition, EIBERR and EIBFREE are set and the first two bytes of EIBERRDC=0864. The conversation is switched to **free** state.

How to synchronize conversations using EXEC CICS SYNCPOINT commands

Data synchronization (the EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK commands) affects all connected conversations at synchronization level 2. The use of these commands is described in Chapter 15, "Sync pointing a distributed process," on page 307.

Ending the conversation

The following information describes the different ways a conversation can end, either unexpectedly or under transaction control. To end a transaction, one transaction issues a request for termination and the other receives this request. When this has happened, the conversation is unusable and **both** transactions must issue an EXEC CICS FREE command to release the session. Conversations are implicitly freed when a transaction ends.

Normal termination of a conversation

The EXEC CICS SEND LAST command is used to terminate a conversation. It should be used in conjunction with either the WAIT or CONFIRM options, the EXEC CICS SYNCPOINT command, or the EXEC CICS WAIT CONVID command (depending on the conversation synchronization level). This is described in the following table:

Table 50. Command sequence for synchronization levels

Synchronization level	Command sequence
0	EXEC CICS SEND LAST WAIT EXEC CICS FREE
1	EXEC CICS SEND LAST CONFIRM EXEC CICS FREE
2	EXEC CICS SEND LAST (see note) EXEC CICS SYNCPOINT EXEC CICS FREE

Note: It is important that the EXEC CICS SEND LAST command for synchronization level 2 is not accompanied by WAIT. This sequence of commands (with or without the implicit EXEC CICS FREE) is the only way in which CICS synchronization level 2 conversations can normally be terminated.

From the state tables it can be seen that it is possible to end a synchronization level 0 or 1 conversation by issuing the EXEC CICS FREE command, provided the conversation is in **send** (state 2). This generates an implicit EXEC CICS SEND LAST WAIT command before the EXEC CICS FREE is executed.

Emergency termination of a conversation

The EXEC CICS ISSUE ABEND command provides a way to abnormally end the conversation. It is valid for all levels of synchronization.

EXEC CICS ISSUE ABEND can be issued by either transaction, irrespective of whether it is in **send** or **receive** state, at any time after the conversation has started. For a conversation in **send** state (state 2), any deferred data that is waiting for transmission is flushed before the EXEC CICS ISSUE ABEND command is transmitted.

The transaction that issues the EXEC CICS ISSUE ABEND command is not itself abended. It must, however, issue an EXEC CICS FREE command for the conversation unless it is designed to terminate immediately.

If an EXEC CICS ISSUE ABEND command is issued in **receive** (state 5), CICS purges all incoming data until an INVITE, sync point request, or LAST indicator is received. If LAST is received, no abend indication is sent to the partner transaction.

If an EXEC CICS ISSUE ABEND is received, CICS raises the TERMERR condition, sets on EIBERR (=X’FF’), EIBFREE (=X’FF’), and places X’0864’ in the first two bytes of EIBERRCD. The only command that can be subsequently issued for the conversation is EXEC CICS FREE.

When a synchronization level 2 conversation receives EXEC CICS ISSUE ABEND or when you call EXEC CICS ISSUE ABEND of a synchronization level 2 conversation, all other conversations go into backout-required state.

For a complete description of the EXEC CICS ISSUE ABEND command, see the *TXSeries for Multiplatforms Application Programming Reference*.

Unexpected termination of a conversation

If a partner system fails, or a session goes out of service in the middle of a DTP conversation, the conversation is terminated abnormally, and the TERMERR condition is raised on the next command that accesses the conversation. In addition EIBERR and EIBFREE are set on (X'FF'), and EIBERRCD contains a value that represents the reason for the error. Refer to the table in "Checking the outcome of a DTP command."

More information can be found in the following tables:

- "Synchronization level 0 conversation state table" on page 333
- "Synchronization level 1 conversation state table" on page 335
- "Synchronization level 2 conversation state table" on page 338

Checking the outcome of a DTP command

Checking the response from a DTP command can be separated into three stages:

1. Testing for a request failure
2. Testing for indicators that are received on the conversation
3. Testing the conversation state

Testing for request failure is the same as for other EXEC CICS commands in that conditions are raised and can be handled by use of EXEC CICS HANDLE CONDITION or RESP. EIBRCODE will also contain an error code.

If the request has not failed, it is then possible to test for indicators that are received on the conversation. These are returned to the application in the EIB. The following EIB fields are relevant to all DTP commands:

EIBERR
When set to X'FF', indicates that an error has occurred on the conversation. The reason is in EIBERRCD. This could be as a result of an EXEC CICS ISSUE ERROR, EXEC CICS ISSUE ABEND, or EXEC CICS SYNCPOINT ROLLBACK command that the partner transaction issued. EIBERR can be set as a result of any command that can be issued while the conversation is in **receive** (state 5), or following any command that causes a transmission to the partner system. It is safest to test EIBERR in conjunction with EIBFREE and EIBSYNRB after every DTP command.

EIBERRCD
Contains the error code that is associated with EIBERR. If EIBERR is not set, this field is not used.

EIBFREE
When set to X'FF', indicates that the partner transaction had ended the conversation. It should be tested along with EIBERR and EIBSYNC to find out exactly how to end the conversation.

EIBSYNRB
When set to X'FF', indicates that the partner transaction or system has issued an EXEC CICS SYNCPOINT ROLLBACK command. (This is relevant only for conversations at synchronization level 2.)

EIBSIG
When set to X'FF', indicates that the partner transaction or system has issued an EXEC CICS ISSUE SIGNAL command.

The following table shows how these EIB fields interact.

Table 51. Interaction between some EIB fields

EIBERR	EIBFREE	EIBSYNRB	EIBERRCD	Description
X'FF'	X'FF'	X'00'	08640000	The remote application or system has sent ISSUE ABEND.
X'FF'	X'FF'	X'00'	08640001	The remote system has sent ISSUE ABEND.
X'FF'	X'FF'	X'00'	08640002	A remote resource has timed out.
X'FF'	X'00'	X'00'	08890000	The partner transaction has sent EXEC CICS ISSUE ERROR
X'FF'	X'FF'	X'00'	10086032	The PIP data that was sent with the EXEC CICS CONNECT PROCESS was incorrectly specified.
X'FF'	X'FF'	X'00'	10086034	The partner system does not support mapped conversations.
X'FF'	X'FF'	X'00'	080f6051	The partner transaction failed security check.
X'FF'	X'FF'	X'00'	10086041	The partner transaction does not support the synchronization level that is requested on the EXEC CICS CONNECT PROCESS.
X'FF'	X'FF'	X'00'	10086021	The partner transactions name is not recognized by the partner system.
X'FF'	X'FF'	X'00'	084c0000	The partner system cannot start the partner transaction.
X'FF'	X'FF'	X'00'	084b6031	The partner system temporarily cannot start the partner transaction.
X'FF'	X'00'	X'FF'	08240000	The partner transaction or system has issued EXEC CICS SYNCPOINT ROLLBACK.
X'FF'	X'FF'	X'00'	a0000100	The session to the remote system has failed.
X'00'	X'00'	--	--	The command completed successfully.

Refer to the EIB flags that are described below.

In addition, the following EIB fields are relevant only to the EXEC CICS RECEIVE and EXEC CICS CONVERSE commands:

EIBCOMPL
When set to X'FF', indicates that all the data that was sent at one time has been received. This field is used in conjunction with the EXEC CICS RECEIVE NOTRUNCATE command.

EIBEOC
When set to X'FF', indicates that an end-of-chain indicator has been received. This field is normally associated with a successful EXEC CICS RECEIVE command. Because, CICS supports only mapped conversations, this field is provided only for compatibility with old CICS programs.

EIBNODAT
When set to X'FF', indicates that no application data has been received.

EIBRECV
Is used only when EIBERR is not set. When EIBRECV is on (X'FF'), another EXEC CICS RECEIVE is required.

EIBSYNC
When set to X'FF', indicates that the partner transaction or system has requested a sync point. (This is relevant only for conversations at synchronization level 2.)

EIBCONF
When set to X'FF', indicates that the partner transaction has issued an EXEC CICS SEND CONFIRM command and requires a response.

Table 52. RECEIVE and CONFIRM flags

EIBERR	EIBFREE	EIBRECV	EIBSYNC	EIBCONF	Description
X'00'	X'00'	X'00'	X'00'	X'00'	The partner transaction or system issued EXEC CICS SEND INVITE WAIT. The local program is now in send state.
X'00'	X'00'	X'00'	X'FF'	X'00'	The partner transaction or system issued EXEC CICS SEND INVITE followed by an EXEC CICS SYNCPOINT. The local program is now in syncsend state.
X'00'	X'00'	X'00'	X'00'	X'FF'	The partner transaction or system issued EXEC CICS SEND INVITE CONFIRM. The local program is now in confsend state.
X'00'	X'00'	X'FF'	X'00'	X'00'	The partner transaction or system issued EXEC CICS SEND or EXEC CICS SEND WAIT. The local program is in receive state.
X'00'	X'00'	X'FF'	X'FF'	X'00'	The partner transaction or system issued an EXEC CICS SYNCPOINT. The local program is now in syncreceive state.
X'00'	X'00'	X'FF'	X'00'	X'FF'	The partner transaction or system issued EXEC CICS SEND CONFIRM. The local program is now in confreceive state.
X'00'	X'FF'	X'00'	X'00'	X'00'	The partner transaction or system issued EXEC CICS SEND LAST WAIT. The local program is now in free state.
X'00'	X'FF'	X'00'	X'FF'	X'00'	The partner transaction or system issued EXEC CICS SEND LAST followed by an EXEC CICS SYNCPOINT. The local program is now in syncfree state.
X'00'	X'FF'	X'00'	X'00'	X'FF'	The partner transaction or system issued EXEC CICS SEND LAST CONFIRM. The local program is now in conffree state.

After analyzing the EIB fields, you can test the conversation state to determine which DTP commands you can issue next. See “Testing the conversation state” on page 303.

Checking EIB fields and the conversation state

Most of the information that is supplied by EIB indicator fields can also be obtained from the conversation state. However, although the conversation state is easier to test, you cannot ignore EIBERR (and EIBERRCD).

For example, if after an EXEC CICS SEND INVITE WAIT or an EXEC CICS RECEIVE command has been issued, the conversation is in **receive** (state 5), only

EIBERR indicates that the partner transaction has sent an ISSUE ERROR. This is shown in “Switching from sending to receiving data” on page 292 and “Receiving data from the partner transaction” on page 293.

It should be noted that the state tables provided contain not only states and commands that are issued, but also relevant EIB field settings. The sequence in which these EIB fields are shown provides a sensible sequence of checks for an application.

Testing the conversation state

A transaction can inquire about the current state of one of its conversations in two ways:

- The first is to use the EXEC CICS EXTRACT ATTRIBUTES command.
- The second is to use the STATE parameter on the DTP commands.

In both cases, the current state is returned to the application in a CICS value data area (cvda). The following table shows how the cvda codes relate to the conversation state. The table also shows the symbolic names that are defined for these cvda values.

Table 53. Relationship of cvda codes to conversation states

State name of conversation states	State number	Symbolic name of states that are used in DTP programs	cvda code
Allocated	1	DFHVALUE (ALLOCATED)	81
Send	2	DFHVALUE (SEND)	90
Pendreceive	3	DFHVALUE (PENDRECEIVE)	87
Pendfree	4	DFHVALUE (PENDFREE)	86
Receive	5	DFHVALUE (RECEIVE)	88
Confreceive	6	DFHVALUE (CONFRECEIVE)	83
Confsend	7	DFHVALUE (CONFSEND)	84
Conffree	8	DFHVALUE (CONFFREE)	82
Syncreceive	9	DFHVALUE (SYNCRECEIVE)	92
Syncsend	10	DFHVALUE (SYNCSEND)	93
Syncfree	11	DFHVALUE (SYNCFREE)	91
Free	12	DFHVALUE (FREE)	85
Rollback	13	DFHVALUE (ROLLBACK)	89

Initial states

A front-end transaction in a conversation must issue an ALLOCATE command to acquire a conversation. If the conversation is successfully allocated, the front-end transaction’s side of the conversation goes into **allocated** (state 1).

A back-end transaction is initially in **receive** (state 5).

Appendix B, “The conversation state tables,” on page 331 tabulates the conversation states, and shows which commands you can issue from any state, and what the effect of those commands will be.

Summary of CICS commands for APPC mapped conversations

This table shows the CICS commands that are used in APPC mapped conversations:

Table 54. CICS commands used in APPC mapped conversations

CICS API command	Use to ...	Sync-levels
EXEC CICS ALLOCATE	Acquire a session.	0, 1, 2
EXEC CICS CONNECT PROCESS	Initiate a conversation.	0, 1, 2
EXEC CICS EXTRACT PROCESS	Access session-related information.	0, 1, 2
EXEC CICS SEND	Send data and control information to the conversation partner.	0, 1, 2
EXEC CICS RECEIVE	Receive data from the conversation partner.	0, 1, 2
EXEC CICS CONVERSE	Send and receive data on the conversation.	0, 1, 2
EXEC CICS WAIT CONVID	Transmit any deferred data or control indicators.	0, 1, 2
EXEC CICS ISSUE CONFIRMATION	Reply positively to EXEC CICS SEND CONFIRM.	1, 2
EXEC CICS ISSUE PREPARE	Prepare a conversation partner for sync pointing.	2
EXEC CICS ISSUE ERROR	Inform the conversation partner of a program-detected error.	0, 1, 2
EXEC CICS ISSUE SIGNAL	Signal an unusual condition to the conversation partner, usually against the flow of data.	0, 1, 2
EXEC CICS ISSUE ABEND	Inform the conversation partner that the conversation should be abandoned.	0, 1, 2
EXEC CICS FREE	Free the session.	0, 1, 2
EXEC CICS SYNCPOINT	Inform all conversation partners of readiness to commit changes to recoverable resources.	2
EXEC CICS SYNCPOINT ROLLBACK	Inform conversation partners of the need to back out changes to recoverable resources.	2

Using DTP to check the availability of the remote system

TXSeries for Multiplatforms does not establish long term sessions with the systems with which it is communicating, in the same way that IBM mainframe-based CICS does. This makes it difficult to determine from TXSeries for Multiplatforms whether a remote system is available. Similarly, it is not possible to determine from a IBM mainframe-based CICS system whether a TXSeries for Multiplatforms region is available. CEMT INQUIRE CONNECTION shows whether any sessions are bound between the local IBM mainframe-based CICS system and SNA. However, this does not mean that the PPC Gateway server, or the TXSeries for Multiplatforms region that is behind i, is available.

You can use a pair of distributed transaction processing (DTP) programs to check whether a remote system is available. All that these programs do is establish whether it is possible to schedule a transaction on the remote system. This is the conversation in which the two programs take part:

```
Front-end program                      Back-end Program
running on local system                running on remote system
```

```
ALLOCATE SYSID(conn-name)
CONNECT PROCESS SYNCLEVEL(1)
SEND LAST CONFIRM            --------->  RECEIVE
  RESP=NORMAL                <---------  ISSUE CONFIRMATION
FREE                                     FREE
```

The front-end program requests that the back-end program is started on the remote system, then waits for a reply from the back-end program by using the EXEC CICS SEND LAST CONFIRM command. This command will return a NORMAL response only if the back-end program successfully starts and replies.

The front-end program can be enhanced to check the availability of all its remote systems by using the EXEC CICS INQUIRE CONNECTION (Browse) capability. Therefore, the front-end program can extract the name of each connection and attempt the DTP conversation with the corresponding remote system. Here is an outline of the possible front-end program logic:

```
EXEC CICS INQUIRE CONNECTION START

while RET-CODE = DFHRESP(NORMAL)
:      EXEC CICS INQUIRE CONNECTION(CONN-NAME)
:                        NETNAME(NET-NAME)
:                        RESP(RET-CODE)
:      if RET-CODE = DFHRESP(NORMAL)
:      :   EXEC CICS ALLOCATE SYSID(CONN-NAME)
:      :                      RESP(DTP-RET-CODE)
:      :   if DTP-RET-CODE = DFHRESP(NORMAL)
:      :   :   CONV-ID = EIBRSRCE
:      :   :   EXEC CICS CONNECT PROCESS CONVID(CONV-ID)
:      :   :                             PROCNAME(tranid)
:      :   :                             PROCLENGTH(4)
:      :   :                             SYNCLEVEL(1)
:      :   :                             RESP(DTP-RET-CODE)
:      :   :   if DTP-RET-CODE = DFHRESP(NORMAL)
:      :   :   :   EXEC CICS SEND CONVID(CONV-ID)
:      :   :   :                  LAST
:      :   :   :                  WAIT
:      :   :   :                  RESP(DTP-RET-CODE)
:      :   :   :   if DTP-RET-CODE = DFHRESP(NORMAL)
:      :   :   :   :   EXEC CICS FREE CONVID(CONV-ID)
:      :   :   :   :                  WAIT
:      :   :   :   :                  RESP(DTP-RET-CODE)
:      :   Write out results

EXEC CICS INQUIRE CONNECTION END

EXEC CICS RETURN
```

Note: These programs should work on any type of CICS system.

Chapter 15. Sync pointing a distributed process

This chapter discusses how to include sync pointing in a distributed process. The information concentrates on the programming aspects of using the EXEC CICS SYNCPOINT [ROLLBACK] command across conversations.

The following are described:
- "The EXEC CICS SYNCPOINT command"
- "The EXEC CICS ISSUE PREPARE command" on page 308
- "The EXEC CICS SYNCPOINT ROLLBACK command" on page 308
- "When a backout is required" on page 309
- "Synchronizing two CICS systems" on page 309
- "Synchronizing three or more CICS systems" on page 314

The EXEC CICS SYNCPOINT command

The EXEC CICS SYNCPOINT command is used to commit recoverable resources. In a DTP environment, the effect of the EXEC CICS SYNCPOINT command is propagated across all conversations that are using synchronization level 2. So, no matter how many DTP transactions are connected by conversations at synchronization level 2, the distributed process should be designed such that only one of the transactions initiates sync point activity for the distributed unit of work. When issuing the SYNCPOINT command, this transaction, known as the *sync point initiator* must be in send state (state 2), pendreceive state (state 3), or pendfree state (state 4) on all its conversations at synchronization level 2. Any transaction that receives the sync point request becomes a *sync point agent*.

A sync point agent is in **receive** state on its conversation with the sync point initiator and becomes aware of the sync point request by testing EIBSYNC after issuing an EXEC CICS RECEIVE command. If it decides to respond positively by issuing EXEC CICS SYNCPOINT, it must be in an appropriate state on all the conversations with its own agents, for which it has become sync point initiator. If an agent transaction responds negatively to a sync point request by issuing EXEC CICS SYNCPOINT ROLLBACK, the initiator sees EIBRLDBK set (FF), which must be tested on return from the EXEC CICS SYNCPOINT command.

Your transaction design should ensure that all transactions are in the correct conversation state before an EXEC CICS SYNCPOINT command is issued.

When a sync point agent receives the sync point request, it is given the opportunity to respond positively (to commit recoverable resources) with an EXEC CICS SYNCPOINT command or negatively (to back out recoverable resources) with an EXEC CICS SYNCPOINT ROLLBACK command.

For information about backing out recoverable resources, see "The EXEC CICS SYNCPOINT ROLLBACK command" on page 308

Examples of these commands are given in "Synchronizing two CICS systems" on page 309 and "Synchronizing three or more CICS systems" on page 314

The EXEC CICS ISSUE PREPARE command

The EXEC CICS ISSUE PREPARE command is used to send the initial sync point flow to a selected partner on an APPC conversation at synchronization level 2. Depending on the partner′s response, this command can then be followed by an EXEC CICS SYNCPOINT or EXEC CICS SYNCPOINT ROLLBACK command.

The reasons for using EXEC CICS ISSUE PREPARE are as follows:

1. In complex DTP that involves several conversing transactions, an ISSUE ERROR command from one of the transactions might not reach the sync point initiator in time to prevent it from issuing an EXEC CICS SYNCPOINT command. This can lead to complex backout procedures for the distributed unit of work.

   Use EXEC CICS ISSUE PREPARE as a way of flushing any error responses from the network.
2. If one or more sync point agents are not completely ″reliable″, use EXEC CICS ISSUE PREPARE to check the status of these agents before proceeding with a general distributed sync point.

Receiving EXEC CICS ISSUE PREPARE is exactly the same as receiving EXEC CICS SYNCPOINT. The sync point agent program cannot detect any difference.

The EXEC CICS SYNCPOINT ROLLBACK command

The EXEC CICS SYNCPOINT ROLLBACK command is used to back out changes to recoverable resources. In a DTP environment, the effect of the EXEC CICS SYNCPOINT ROLLBACK command is propagated across all conversations that are using synchronization level 2. An EXEC CICS SYNCPOINT ROLLBACK command can be issued in any conversation state. If the command is issued when a conversation is in **receive** (state 5), incoming data on that conversation is purged as described for the EXEC CICS ISSUE ERROR and EXEC CICS ISSUE ABEND commands.

When a transaction receives an EXEC CICS SYNCPOINT ROLLBACK in response to a sync point request, the EIBRLDBK indicator is set. If EXEC CICS SYNCPOINT ROLLBACK is received in response to any other request, the EIBERR and EIBSYNRB indicators are set.

The rules for determining the state after EXEC CICS SYNCPOINT ROLLBACK depend on the CICS release of the remote partner system. CICS follows APPC architecture. Therefore, following an EXEC CICS SYNCPOINT ROLLBACK command, conversations revert to the state that they were in at the start of the LUW.

If a session failure or notification of a deallocate abend occurs during EXEC CICS SYNCPOINT ROLLBACK processing, the command still completes successfully. If the same thing happens during EXEC CICS SYNCPOINT processing, the command can complete successfully with EIBRLDBK set. In such conditions, the conversation on which the failure or abend occurred is in **free** state (state 12).

To avoid potential state checks, it is therefore advisable to check the conversation state, by using the EXEC CICS EXTRACT ATTRIBUTES command before issuing further DTP commands.

When a backout is required

A backout is required in the following conditions:
- When EXEC CICS SYNCPOINT ROLLBACK is received
- After EXEC CICS ISSUE ABEND is sent
- After EIBERR and EIBFREE are returned together

The conversation state does not always reflect the requirement to back out. However, CICS is aware of this requirement and converts the next EXEC CICS SYNCPOINT request to an EXEC CICS SYNCPOINT ROLLBACK request. If no EXEC CICS SYNCPOINT or EXEC CICS SYNCPOINT ROLLBACK request is issued before the end of the task, the task is abended, and all recoverable resources are backed out.

Synchronizing two CICS systems

This information gives examples of how to commit and back out changes to recoverable resources that are made by two DTP transactions connected on a conversation that is using synchronization level 2.

The examples show the following scenarios:
- “SYNCPOINT in response to SYNCPOINT”
- “SYNCPOINT in response to ISSUE PREPARE” on page 310
- “SYNCPOINT ROLLBACK in response to SYNCPOINT ROLLBACK” on page 311
- “SYNCPOINT ROLLBACK in response to SYNCPOINT” on page 311
- “SYNCPOINT ROLLBACK in response to ISSUE PREPARE” on page 312
- “ISSUE ERROR in response to SYNCPOINT” on page 312
- “ISSUE ERROR in response to ISSUE PREPARE” on page 313
- “ISSUE ABEND in response to SYNCPOINT” on page 313
- “ISSUE ABEND in response to ISSUE PREPARE” on page 314

SYNCPOINT in response to SYNCPOINT

Figure 96, Figure 97 on page 310, and Figure 98 on page 310 show the effect of EXEC CICS SEND, EXEC CICS SEND INVITE, or EXEC CICS SEND LAST preceding EXEC CICS SYNCPOINT on an APPC mapped conversation. These figures also show the conversation state before each command and the state and EIB fields that are set after each command.

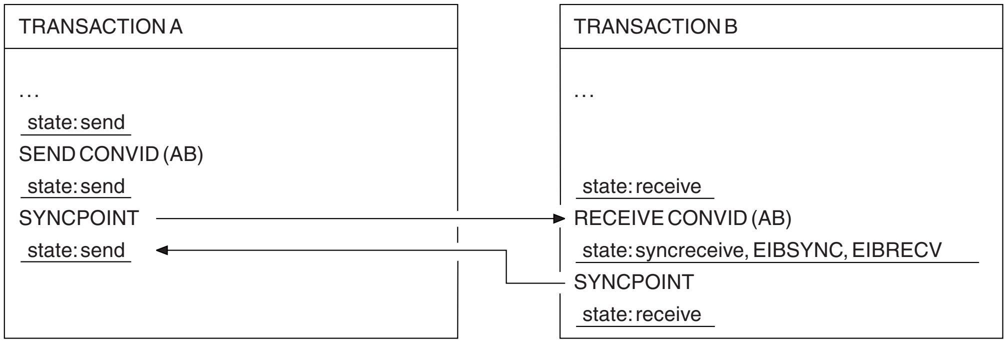


Figure 96. EXEC CICS SYNCPOINT in response to EXEC CICS SEND followed by EXEC CICS SYNCPOINT on a conversation

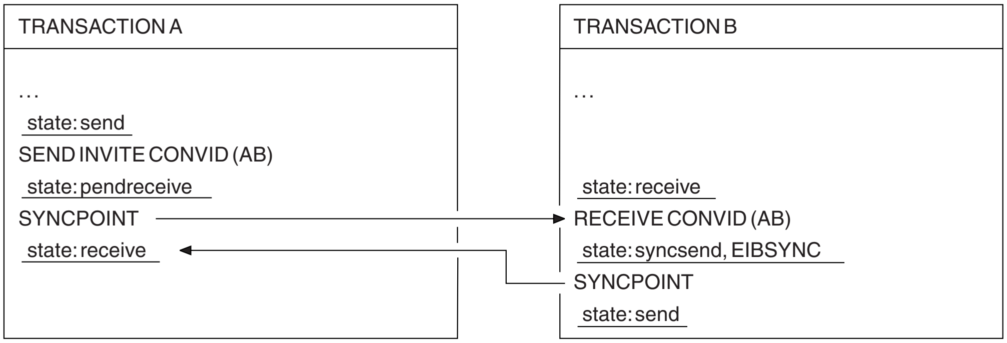


Figure 97. EXEC CICS SYNCPOINT in response to EXEC CICS SEND INVITE followed by EXEC CICS SYNCPOINT on a conversation

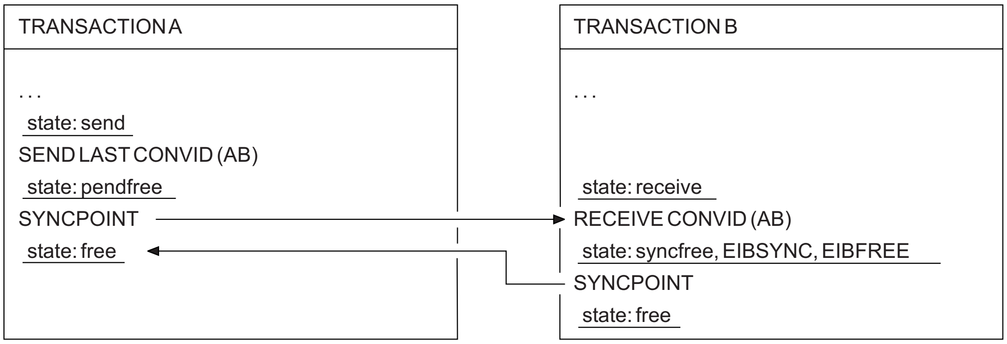


Figure 98. EXEC CICS SYNCPOINT in response to EXEC CICS SEND LAST followed by EXEC CICS SYNCPOINT on a conversation

SYNCPOINT in response to ISSUE PREPARE

Figure 99 on page 311 shows an EXEC CICS SYNCPOINT command being used in response to EXEC CICS ISSUE PREPARE on a conversation. This figure also shows the conversation state before each command and the state and EIB fields that are set after each command.

Note that you can use also an EXEC CICS ISSUE PREPARE command in **pendreceive** state (state 3) and **pendfree** state (state 4).

Note also that, although the EXEC CICS ISSUE PREPARE command in the figure returns with the conversation in **syncsend** state (state 10), the only commands that are available for use on that conversation are EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK. All other commands abend ATCV.

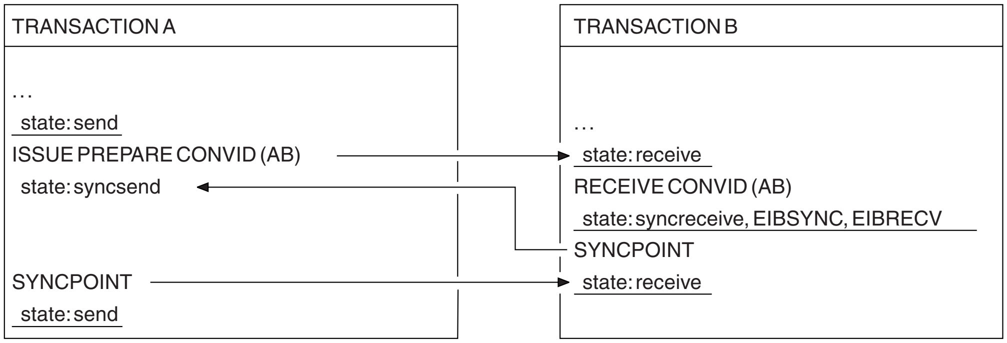


Figure 99. EXEC CICS SYNCPOINT in response to EXEC CICS ISSUE PREPARE on a conversation

SYNCPOINT ROLLBACK in response to SYNCPOINT ROLLBACK

Figure 100 shows an EXEC CICS SYNCPOINT ROLLBACK command being used in response to EXEC CICS SYNCPOINT ROLLBACK on a conversation. This figure also shows the conversation state before each command and the state and EIB fields that are set after each command.

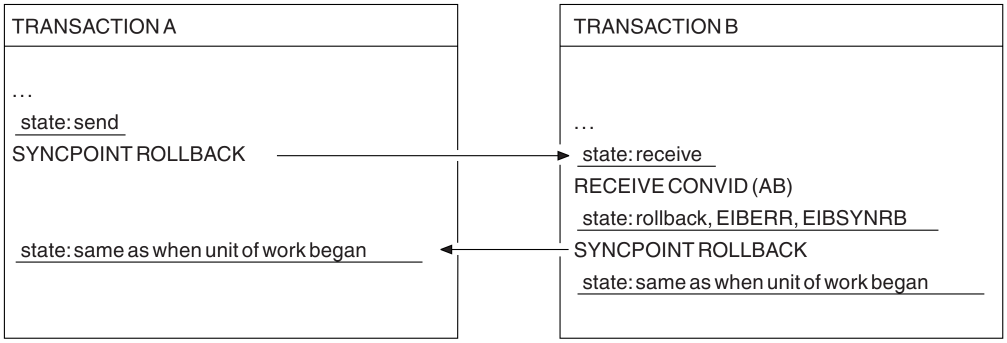


Figure 100. EXEC CICS SYNCPOINT ROLLBACK in response to EXEC CICS SYNCPOINT ROLLBACK on a conversation

SYNCPOINT ROLLBACK in response to SYNCPOINT

Figure 101 on page 312 shows an EXEC CICS SYNCPOINT ROLLBACK command being used in response to EXEC CICS SYNCPOINT on a conversation. This figure also shows the conversation state before each command and the state and EIB fields that are set after each command.

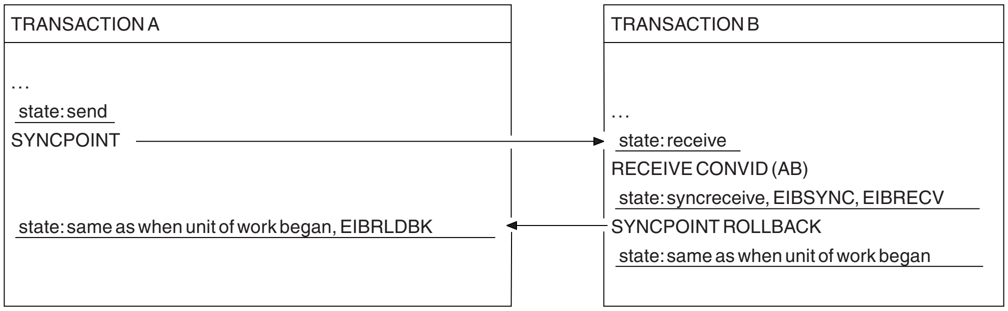

TRANSACTION A
...
state: send
SYNCPOINT
state: same as when unit of work began, EIBRLDBK
TRANSACTION B
...
state: receive
RECEIVE CONVID (AB)
state: syncreceive, EIBSYNC, EIBRECV
SYNCPOINT ROLLBACK
state: same as when unit of work began

Figure 101. EXEC CICS SYNCPOINT ROLLBACK in response to EXEC CICS SYNCPOINT on a conversation

SYNCPOINT ROLLBACK in response to ISSUE PREPARE

Figure 102 shows an EXEC CICS SYNCPOINT ROLLBACK command being used in response to EXEC CICS ISSUE PREPARE on a conversation. This figure also shows the conversation state before each command and the state and EIB fields that are set after each command.

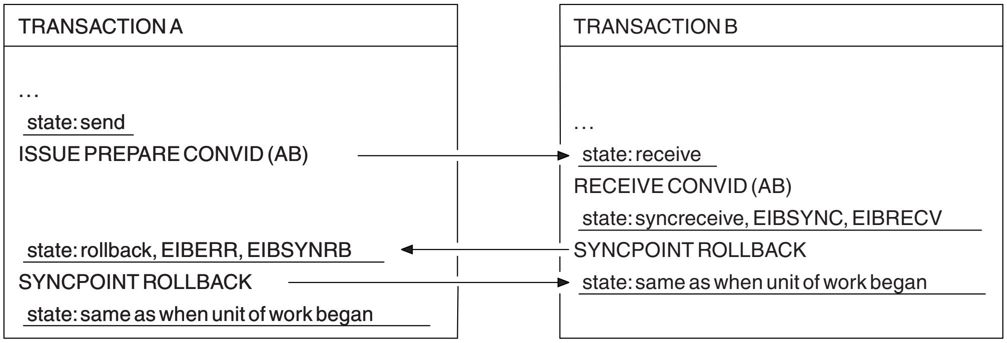

TRANSACTION A
...
state: send
ISSUE PREPARE CONVID (AB)
state: rollback, EIBERR, EIBSYNRB
SYNCPOINT ROLLBACK
state: same as when unit of work began
TRANSACTION B
...
state: receive
RECEIVE CONVID (AB)
state: syncreceive, EIBSYNC, EIBRECV
SYNCPOINT ROLLBACK
state: same as when unit of work began

Figure 102. EXEC CICS SYNCPOINT ROLLBACK in response to EXEC CICS ISSUE PREPARE on a conversation

ISSUE ERROR in response to SYNCPOINT

Figure 103 on page 313 shows an EXEC CICS ISSUE ERROR command being used in response to EXEC CICS SYNCPOINT on a conversation. The figure also shows the conversation state before each command and the state and EIB fields that are set after each command. You can also send EXEC CICS ISSUE ERROR before receiving EXEC CICS SYNCPOINT, but this is not shown because the results are the same.

It is pointless to use EXEC CICS ISSUE ERROR as a response to EXEC CICS SYNCPOINT, because this causes the sync point initiator to discard all data that was transmitted with the EXEC CICS ISSUE ERROR by the sync point agent. To safeguard integrity, the sync point agent has to issue a EXEC CICS SYNCPOINT ROLLBACK command.

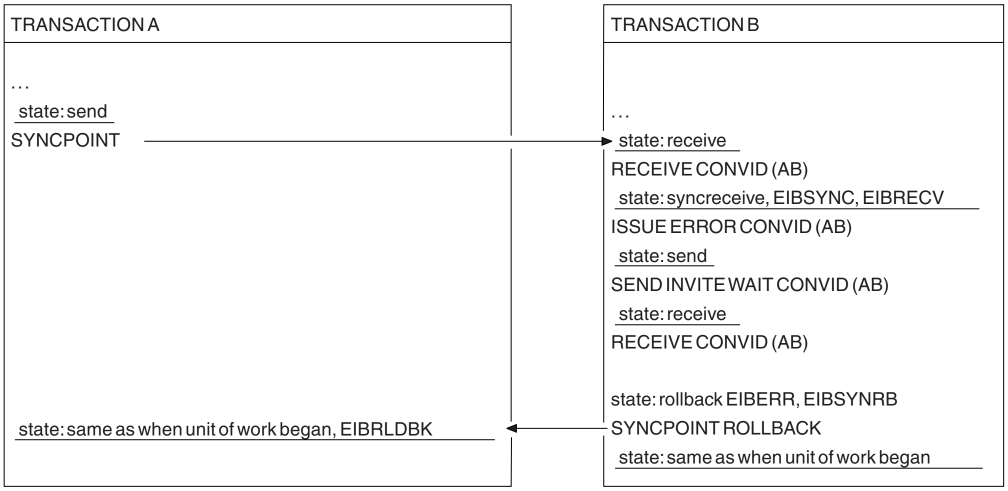

TRANSACTION A
TRANSACTION B
…
state: send
SYNCPOINT
…
state: receive
RECEIVE CONVID (AB)
state: syncreceive, EIBSYNC, EIBRECV
ISSUE ERROR CONVID (AB)
state: send
SEND INVITE WAIT CONVID (AB)
state: receive
RECEIVE CONVID (AB)
state: rollback EIBERR, EIBSYNRB
state: same as when unit of work began, EIBRLDBK
SYNCPOINT ROLLBACK
state: same as when unit of work began

Figure 103. EXEC CICS ISSUE ERROR in response to EXEC CICS SYNCPOINT on a conversation

ISSUE ERROR in response to ISSUE PREPARE

Figure 104 shows an EXEC CICS ISSUE ERROR command being used in response to EXEC CICS ISSUE PREPARE on an APPC mapped conversation. This figure also shows the conversation state before each command and the state and EIB fields that are set after each command. You can also send EXEC CICS ISSUE ERROR before receiving EXEC CICS ISSUE PREPARE, but this is not shown, because the results are the same.

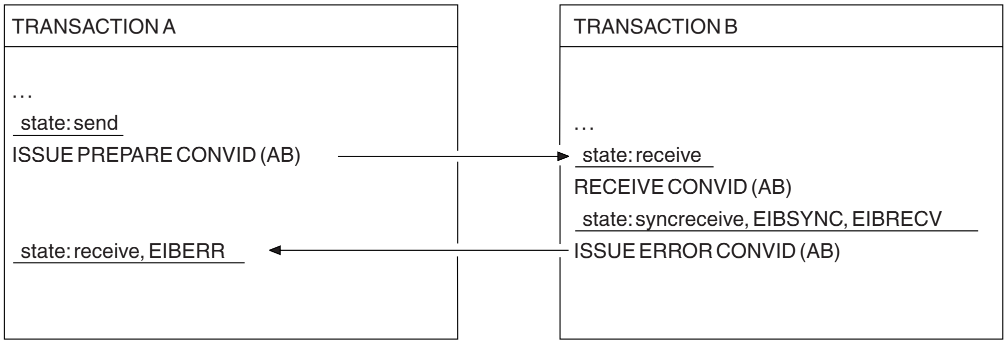


Figure 104. EXEC CICS ISSUE ERROR in response to EXEC CICS ISSUE PREPARE on a conversation

ISSUE ABEND in response to SYNCPOINT

Figure 105 on page 314 shows an EXEC CICS ISSUE ABEND command being used in response to EXEC CICS SYNCPOINT on a conversation. The figure also shows the conversation state before each command and the state and EIB fields that are set after each command. You can also send EXEC CICS ISSUE ABEND before receiving EXEC CICS SYNCPOINT, but this is not shown because the results are the same.

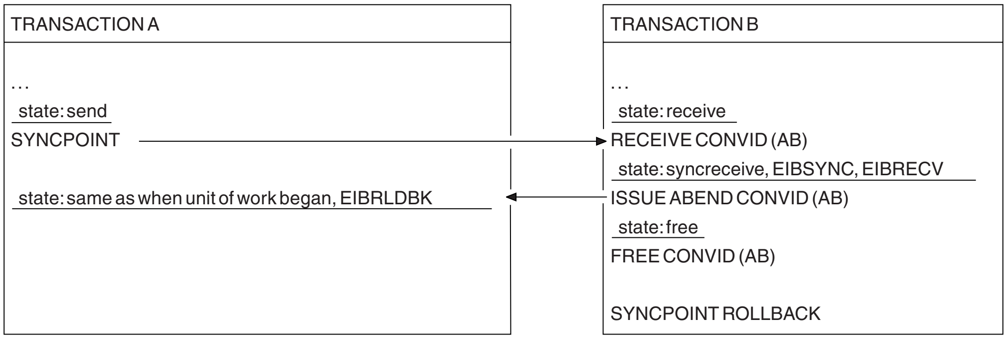


Figure 105. EXEC CICS ISSUE ABEND in response to EXEC CICS SYNCPOINT on a conversation

ISSUE ABEND in response to ISSUE PREPARE

The following figure shows an EXEC CICS ISSUE ABEND command being used in response to EXEC CICS ISSUE PREPARE on a conversation. The figure also shows the conversation state before each command and the state and EIB fields that are set after each command. You can also send EXEC CICS ISSUE ABEND before receiving EXEC CICS ISSUE PREPARE, but this is not shown because the results are the same.

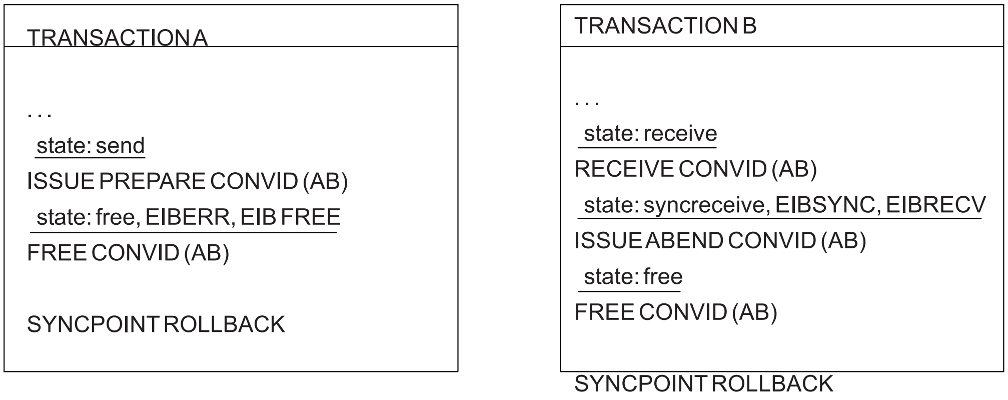


Figure 106. EXEC CICS ISSUE ABEND in response to EXEC CICS ISSUE PREPARE on a conversation

Synchronizing three or more CICS systems

This section gives examples of how to commit and back out recoverable resources that are affected by three or more DTP transactions that are connected on conversations at synchronization level 2.

Sync point in response to EXEC CICS SYNCPOINT

Figure 107 on page 315 shows the sequence of events for a successful sync point involving six conversing transactions. It illustrates the states and actions that occur when transactions issue EXEC CICS SYNCPOINT requests. To write successful distributed applications, you do not need to understand all the data flows that occur during a distributed sync point. In this example, the programmer is concerned only with issuing EXEC CICS SYNCPOINT in response to finding a conversation in **syncreceive** (state 9).

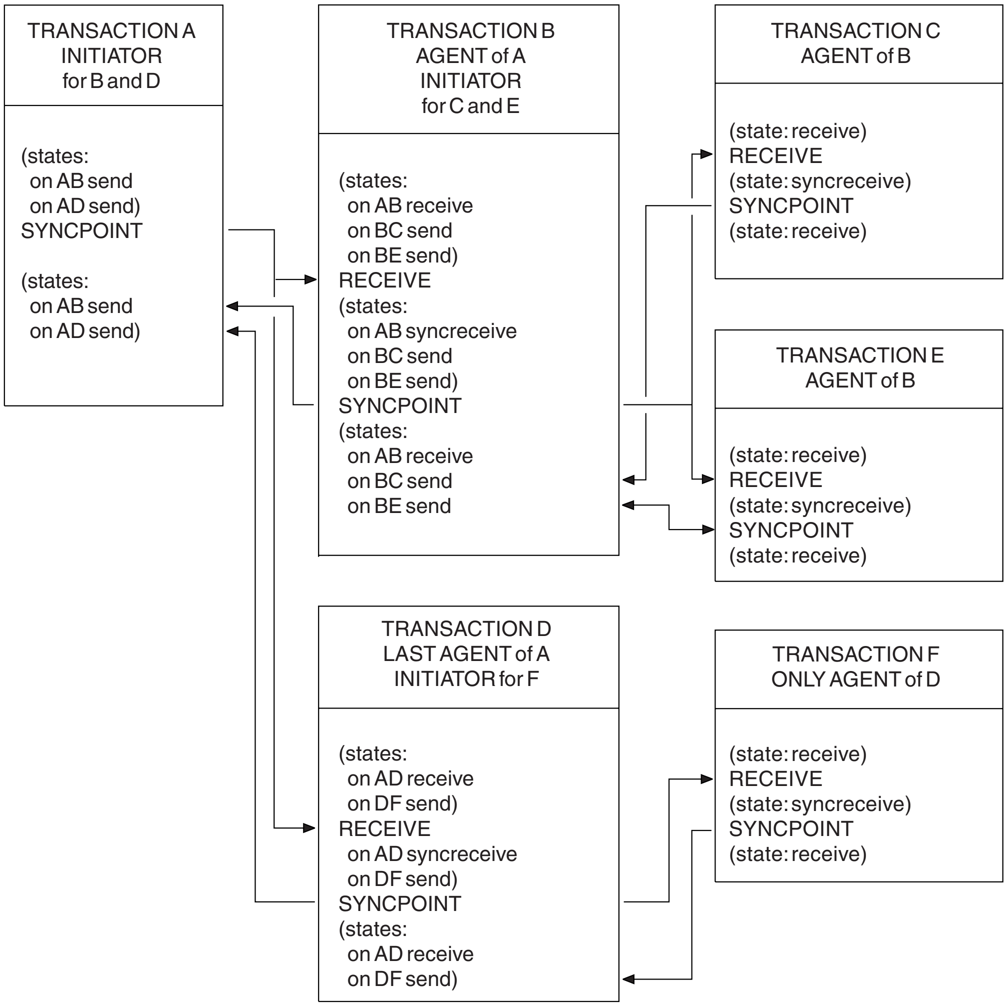


Figure 107. A distributed sync point with all partners running on CICS

1. Transaction A, which is in **send** state (state 2) on its conversations with transactions B and D, decides to end the distributed unit of work, and therefore issues an EXEC CICS SYNCPOINT command.
2. Transaction B sees that its half of its conversation with transaction A is in **syncreceive** state (state 9), so it issues an EXEC CICS SYNCPOINT command. Transaction B is responding to a request from transaction A, but it also becomes the sync point initiator for transactions C and E, and must ensure that its conversations with these transactions are in a valid state for issuing an EXEC CICS SYNCPOINT command. In this example, they are both in **send** state (state 2).
3. Transaction C sees that its half of its conversation with transaction B is in **syncreceive** state (state 9), so it issues an EXEC CICS SYNCPOINT command.
4. Transaction E sees that its half of its conversation with transaction B is in **syncreceive** state (state 9), so it issues an EXEC CICS SYNCPOINT command.
5. Transaction D sees that its half of its conversation with transaction A is in **syncreceive** state (state 9), so it issues an EXEC CICS SYNCPOINT command. Transaction D is responding to a request from transaction A, but it also becomes the sync point initiator for transaction F, and must ensure that its conversation with this transaction is in a valid state for issuing an EXEC CICS SYNCPOINT command. In this example, it is in **send** state (state 2).

6. Transaction F sees that its half of its conversation with transaction D is in **syncreceive** state (state 9), so it issues an EXEC CICS SYNCPOINT command.
7. All the transactions have now indicated, by issuing EXEC CICS SYNCPOINT commands, that they are ready to commit their changes. This process begins with transaction F, which has no agents and has responded to request commit by issuing an EXEC CICS SYNCPOINT command.
8. The distributed sync point is complete and control returns to transaction A following the EXEC CICS SYNCPOINT command.

The previous discussion of the EXEC CICS SYNCPOINT command assumed that all the agent transactions were ready to take a sync point by issuing EXEC CICS SYNCPOINT when their conversation entered **syncreceive** state (state 9).

If, however, an agent has detected an error, it can reject the sync point request with one of the following commands:

- EXEC CICS SYNCPOINT ROLLBACK (preferred response)
- EXEC CICS ISSUE ERROR
- EXEC CICS ISSUE ABEND

The EXEC CICS SYNCPOINT ROLLBACK command enables a transaction to initiate a backout operation across the whole distributed unit of work. When it is issued in response to a sync point request, it has the following effects:

1. Any changes that are made to recoverable resources by the transaction that issues the rollback request are backed out.
2. The sync point initiator is also backed out (EIBRLDBK set).

This causes the sync point initiator to initiate a backout operation across the distributed unit of work.

Sync point rollback in response to EXEC CICS SYNCPOINT

Figure 108 on page 317 shows the same distributed processes as the those that are shown in Figure 107 on page 315. Six transactions are engaged in related conversations. Transaction A (the first initiator) has two conversations: one with transaction B, and the other with transaction D. Transaction B has three conversations: one on its principal facility (with transaction A), another with transaction C, and another with transaction E. Transactions C and E each have one conversation: on their principal facility (with transaction B). Transaction D has two conversations: one on its principal facility (with transaction A), and the other with transaction F. Transaction F has one conversation: on its principal facility (with transaction D).

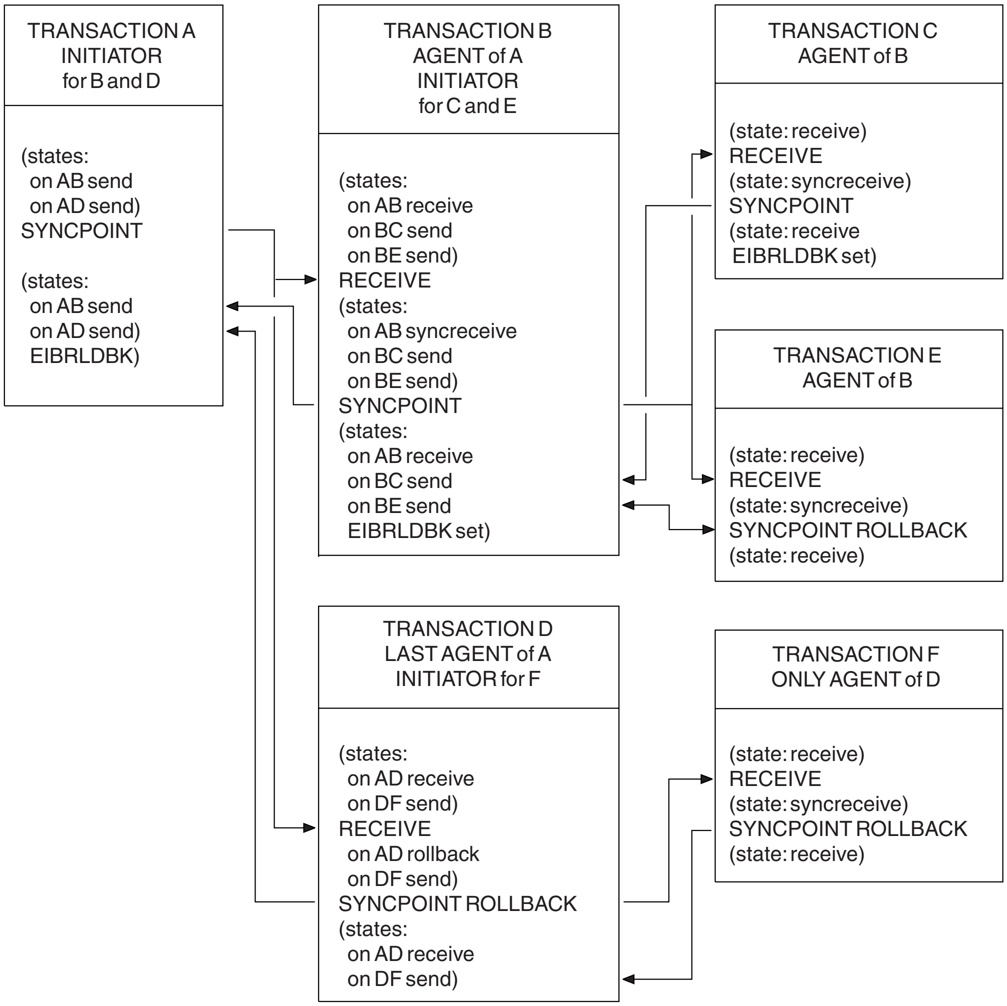


Figure 108. Rollback during distributed sync pointing

As is shown in Figure 107 on page 315, transaction A, while in **send** state (state 2), issues the EXEC CICS SYNCPOINT command, and CICS initiates a chain of events. Here, however, transaction E has detected an error that makes it unable to commit, and it issues EXEC CICS SYNCPOINT ROLLBACK when it detects that the conversation on its principal facility is in **syncreceive** state (state 9, EIBSYNC is also set). This causes any changes that transaction E has made to be backed out, and initiates a distributed rollback.

Transaction D senses that the conversation on its principal facility is in **rollback** state (state 13), and that transactions B, C and A are rolled back (EIBRLDBK is also set). Transaction D therefore issues an EXEC CICS SYNCPOINT ROLLBACK command. Transaction F too senses that the conversation on its principal facility is in **rollback** state, and issues an EXEC CICS SYNCPOINT ROLLBACK command. The distributed rollback is now complete.

Conversation failure and the indoubt period

During the period between the sending of the sync point request to the partner system and the receipt of the reply, the local system does not know whether the partner system has committed the change. This is known as the *indoubt period*. If the intersystem session fails during this period, the local CICS system cannot tell whether the partner system has committed or backed out its resource changes.

Part 5. Appendixes

Appendix A. DFHCNV - The data conversion macros

This reference section describes the macros that are used for data conversion.

Using the DFHCNV macros

Note: DFHCNV macros are generally portable between CICS systems. Some minor changes might be required. See "When TXSeries for Multiplatforms does not convert the data" on page 157 for information about the differences between CICS systems.

The following descriptions provide overview information for using the macros:

DFHCNV TYPE=INITIAL

Establishes the beginning of the macro source conversion table. INITIAL is used to set up the control section for the table. It is also used to specify the default code page that represents how data is stored for all resources on your system. (This can be overridden at the resource level.) A shortcode can be used.

Refer to Table 32 on page 153 for a list of the shortcode and code pages that are supported by the TXSeries for Multiplatforms systems.

This macro can also be used to specify a code page for the incoming request. The TYPE=ENTRY macro overrides this.

Note: TXSeries for Multiplatforms flows a code page in the PIP data. When a code page is flowed to TXSeries for Multiplatforms, the flowed code page is used instead of the code page that is specified with the TYPE=INITIAL macro. See "When TXSeries for Multiplatforms does not convert the data" on page 157 for information about which CICS systems flow code pages.

DFHCNV TYPE=ENTRY

Specifies how data conversion is to occur for a specific resource. Code page information that is specified with the TYPE=ENTRY macro overrides code page information that is specified with the TYPE=INITIAL macro.

The TYPE=ENTRY macro is also used to indicate whether or not a standard or nonstandard conversion is required. If a standard conversion can be used, the TYPE=SELECT and TYPE=FIELD macros are used to specify the field lengths and the type of data that is contained in the fields, such as character, binary, MBCS (graphic), or packed decimal. If a nonstandard conversion is required, the TYPE=ENTRY macro is used to specify that a user exit is required (described in "Non-standard data conversion (DFHUCNV) for function shipping, DPL and asynchronous processing" on page 160). Standard conversions can be used when:

1. The field contains data that can be converted with a type that is specified with DFHCNV TYPE=FIELD DATATYP.
2. The fields are fixed length.

DFHCNV TYPE=KEY

Indicates the start of conversions to be applied to a key. This is applicable only when the resource is a file with key (KSDS file).

DFHCNV TYPE=SELECT
Declares the selection criteria for a particular conversion. SELECT cannot be used if a nonstandard conversion is specified.

DFHCNV TYPE=FIELD
Specifies field offsets and the type of conversions that are required. FIELD cannot be used if a nonstandard conversion is specified.

DFHCNV TYPE=FINAL
Concludes the macro source conversion table definition. This must occur only once, as the last definition.

Examples of DFHCNV macros

An example macro source conversion table is shown in Figure 109:

```
         DFHCNV    TYPE=INITIAL,CLINTCP=037,SRVERCP=850
***********************************************************************
* CONVERSION MACROS FOR TEMPORARY DATA QUEUE "TDQ1" -                 *
***********************************************************************
         DFHCNV    TYPE=ENTRY,RTYPE=TD,RNAME=TDQ1
         DFHCNV    TYPE=SELECT,OPTION=DEFAULT
         DFHCNV    TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,              X
               DATALEN=98,LAST=YES
***********************************************************************
* CONVERSION MACROS FOR TEMPORARY STORAGE QUEUE "TSQ1" -              *
***********************************************************************
         DFHCNV    TYPE=ENTRY,RTYPE=TS,RNAME=TSQ1
         DFHCNV    TYPE=SELECT,OPTION=DEFAULT
         DFHCNV    TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,              x
               DATALEN=98,LAST=YES
***********************************************************************
* CONVERSION MACROS FOR DPL COMMAREA - VSAMSFS APPLICATION            *
***********************************************************************
         DFHCNV    TYPE=ENTRY,RTYPE=PC,RNAME=VSAMSFS
         DFHCNV    TYPE=SELECT,OPTION=DEFAULT
         DFHCNV    TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,              x
               DATALEN=2
         DFHCNV    TYPE=FIELD,OFFSET=2,DATATYP=BINARY,                 x
               DATALEN=2
         DFHCNV    TYPE=FIELD,OFFSET=4,DATATYP=CHARACTER,              x
               DATALEN=129,LAST=YES
***********************************************************************
  DFHCNV TYPE=FINAL
```

Figure 109. Sample macro source for data conversion

Rules for character placement in the source are:
- The statement area extends between positions 2 through 71.
- DFHCNV must begin in column 10.
- TYPE= must begin in column 20.
- The continuation character field occupies position 72 and can be any nonblank character.
- The continued source statement must begin on the next line in position 16.
- Positions 73 and beyond are ignored and can be used for identification sequence numbers.

In this example, the TYPE=INITIAL macro specifies that code page IBM-037 be assumed as the default code page when a function shipping request either does not specify a code page, or the code page specified is invalid. This macro also states that resources that are defined in this table are encoded with IBM-850.

TDQ1, in this example, contains one field. That field contains 98 bytes of character data.

TSQ1 also contains one field of character data.

VSAMSFS defines three fields. The first field is two bytes long and consists of character data. The second field is two bytes long and consists of binary data. The last field is 129 bytes long and consists of character data. Only the character data is converted.

When your conversion table is coded, you can use **cicscvt** to build the conversion templates

Figure 110 shows a record layout for a file named VSAM99. Figure 111 shows a set of conversion macros for the record layout in Figure 110.

```
02  FILEREC.
  03  STAT         PIC X.
  03  NUMB         PIC X(6).
  03  NAME         PIC X(20).
  03  ADDRX        PIC X(20).
  03  PHONE        PIC X(8).
  03  DATEX        PIC X(8).
  03  AMOUNT       PIC X(8).
  03  COMMENT      PIC X(9).
  03  VARINF1.
  03  COUNTER1     PIC 9999 USAGE COMP-4.
  03  COUNTER2     PIC 9999 USAGE COMP-4.
  03  ADDLCMT      PIC X(30).
  03  VARINF2 REDEFINES VARINF1.
  03  COUNTER1     PIC 9999 USAGE COMP-4.
  03  COUNTER2     PIC 9999 USAGE COMP-4.
  03  COUNTER3     PIC 9999 USAGE COMP-4.
  03  COUNTER4     PIC 9999 USAGE COMP-4.
  03  ADDLCMT2     PIC X(26).
```

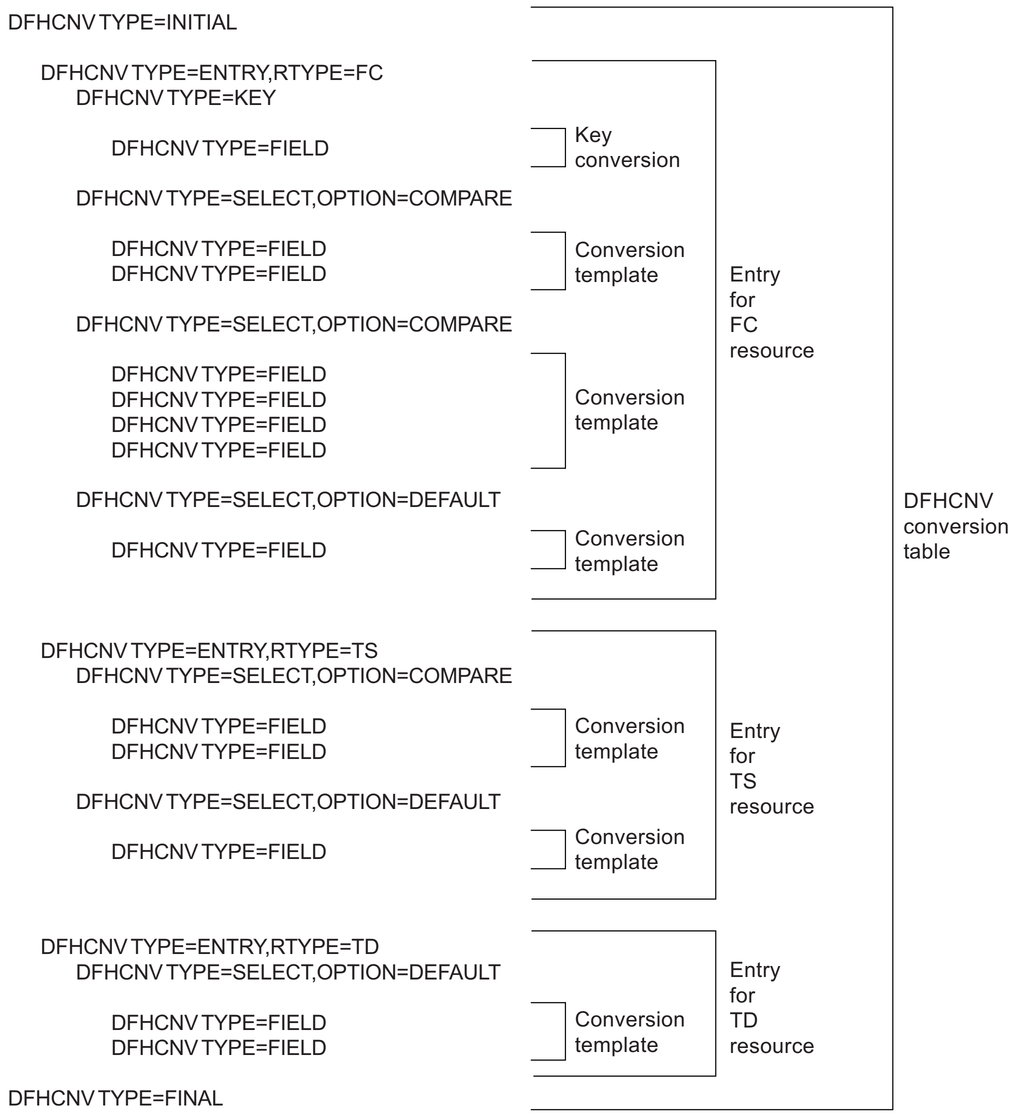
Figure 110. Record layout for VSAM99

```
   DFHCNV TYPE=INITIAL
   DFHCNV TYPE=ENTRY,RTYPE=FC,RNAME=VSAM99
   DFHCNV TYPE=KEY
   DFHCNV TYPE=FIELD,OFFSET=0,DATATYP=CHARACTER,DATALEN=6,LAST=YES
*
*  If offset 0 is a character 'X' use the following
*  conversion definitions:
*

   DFHCNV TYPE=SELECT,OPTION=COMPARE,OFFSET=00,DATA='X'
   DFHCNV TYPE=FIELD,OFFSET=00,DATATYP=CHARACTER,DATALEN=80
   DFHCNV TYPE=FIELD,OFFSET=80,DATATYP=BINARY,DATALEN=4
   DFHCNV TYPE=FIELD,OFFSET=84,DATATYP=CHARACTER,DATALEN=30,LAST=YES
*
*  Otherwise use the following (default)
*  conversion definitions
*
   DFHCNV TYPE=SELECT,OPTION=DEFAULT
   DFHCNV TYPE=FIELD,OFFSET=00,DATATYP=CHARACTER,DATALEN=80
   DFHCNV TYPE=FIELD,OFFSET=80,DATATYP=BINARY,DATALEN=8
   DFHCNV TYPE=FIELD,OFFSET=88,DATATYP=CHARACTER,DATALEN=26,LAST=YES
   DFHCNV TYPE=FINAL
```

Figure 111. Description for record layout for VSAM99

Flow of DFHCNV macro sequence

Refer to Figure 112 that shows an example of the DFHCNV macro sequence.



```
DFHCNV TYPE=INITIAL

   DFHCNV TYPE=ENTRY,RTYPE=FC
      DFHCNV TYPE=KEY

         DFHCNV TYPE=FIELD

      DFHCNV TYPE=SELECT,OPTION=COMPARE

         DFHCNV TYPE=FIELD
         DFHCNV TYPE=FIELD

      DFHCNV TYPE=SELECT,OPTION=COMPARE

         DFHCNV TYPE=FIELD
         DFHCNV TYPE=FIELD
         DFHCNV TYPE=FIELD
         DFHCNV TYPE=FIELD

      DFHCNV TYPE=SELECT,OPTION=DEFAULT

         DFHCNV TYPE=FIELD


   DFHCNV TYPE=ENTRY,RTYPE=TS
      DFHCNV TYPE=SELECT,OPTION=COMPARE

         DFHCNV TYPE=FIELD
         DFHCNV TYPE=FIELD

      DFHCNV TYPE=SELECT,OPTION=DEFAULT

         DFHCNV TYPE=FIELD


   DFHCNV TYPE=ENTRY,RTYPE=TD
      DFHCNV TYPE=SELECT,OPTION=DEFAULT

         DFHCNV TYPE=FIELD
         DFHCNV TYPE=FIELD

DFHCNV TYPE=FINAL
```

Figure 112. Example of DFHCNV macro sequence

The DFHCNV TYPE=INITIAL macro

The DFHCNV TYPE=INITIAL macro establishes the beginning of the conversion table. It sets up the control section for the table. The macro must occur only once as the first definition.

Syntax

DFHCNV TYPE=INITIAL
,CLINTCP={**037**|codepage},SRVERCP={**850**|codepage} [,XA={YES|NO}]
[,CDEPAGE={codepage}]

The options are defined as follows:

CLINTCP

Use CLINTCP to specify the code page of the remote system that is sending the incoming function shipping request. (This code page can be overridden if the requesting region sends a code page with the request that is supported by the local operating system). The TYPE=ENTRY CLINTCP operand overrides the TYPE=INITIAL CLINTCP operand.

The code page can be expressed as a shortcode.

In addition to these shortcodes, you can specify any code page that **iconv** supports, provided that you have on your operating system the translation tables with which to do it. Refer to "Introduction to data conversion" on page 147 for more information.

SRVERCP

Defines the code page that indicates how the resource is encoded. The TYPE=ENTRY SRVERCP operand overrides the TYPE=INITIAL SRVERCP operand.

The code page can be expressed as a shortcode.

In addition to these shortcodes, you can specify any code page that **iconv** supports, provided that you have on your operating system the translation tables with which to do it. Refer to "Introduction to data conversion" on page 147 for more information.

CDEPAGE

Is retained for compatibility with other CICS family members, but is ignored by CICS on Open Systems and CICS for Windows. On other CICS family members, it establishes a client/server code page pair by means of a single number. For example, 437 means CLINTCP=437, SRVERCP=037 and 932 means CLINTCP=932, SRVERCP=930.

XA={YES|NO}

Specifies whether CICS should use extended addressing. CICS on Open Systems and CICS for Windows supports this option for compatibility with other CICS products.

The DFHCNV TYPE=ENTRY macro

The TYPE=ENTRY macro is used for specifying how data conversion is to take place for a specific resource. One TYPE=ENTRY macro must exist for each resource name and type for which conversion is required. If a resource type and name is not present, CICS does not convert the data. The entry for one resource name and type is concluded by the next TYPE=ENTRY statement, or by the end of the table.

Syntax

DFHCNV TYPE=ENTRY ,RTYPE={FC|TD|TS|IC|PC} ,{RNAME=resourcename | XRNAME=xxxxxxxxxxxxxxxx} [,USREXIT={YES | **NO**}] [,CLINTCP=codepage] [,SRVERCP=codepage] [,CDEPAGE=codepage]

The options are defined as follows:

RTYPE

States the type of resource. RTYPE must be one of the following:

FC A file.

TD A transient data queue.

TS A temporary storage queue.

IC An interval control start with a **from** area.
PC A distributed link with COMMAREA.

RNAME

The eight-character resource name. CICS pads shorter names with blanks, and truncates longer names. The resource is one of the following:

- A file name (up to eight characters)
- A transient data queue name (up to eight characters)
- A temporary storage queue name (up to eight characters)
- An interval control start transaction identifier (up to four characters)
- A program name (up to eight characters)

XRNAME

TS and IC only. A 16-digit hexadecimal resource name. CICS pads shorter names with blanks, and truncates longer names.

USREXIT

Allows you to decide whether CICS is to call the user conversion exit. If you need the user conversion exit to convert some data for this resource, select **YES**, otherwise, select **NO**. Selecting **NO** eliminates the overheads that are related to calling the exit unnecessarily. See "Non-standard data conversion (DFHUCNV) for function shipping, DPL and asynchronous processing" on page 160 for more information.

CLINTCP

Use CLINTCP to specify the code page that you want used if the incoming function shipping request either does not specify a code page, or a code page is specified but is not valid for this operating system. The TYPE=ENTRY CLINTCP operand overrides the TYPE=INITIAL CLINTCP operand.

The code page can be expressed as a shortcode. See "Using the DFHCNV macros" on page 321 for a list of shortcodes and their equivalent code pages.

In addition to these shortcodes, you can specify any code page that **iconv** supports, provided that you have on your operating system the translation tables with which to do it. Refer to "Introduction to data conversion" on page 147 for more information.

SRVERCP

Specifies the code page that indicates how the resource is encoded. The TYPE=ENTRY SRVERCP operand overrides the TYPE=INITIAL SRVERCP operand.

The code page can be expressed as a shortcode. See "Using the DFHCNV macros" on page 321 for a list of shortcodes and their equivalent code pages.

In addition to these shortcodes, you can specify any code page that **iconv** supports, provided that you have on your operating system the translation tables with which to do it. See "Introduction to data conversion" on page 147 for more information.

CDEPAGE

Is retained for compatibility with other CICS family members, but is ignored by CICS on Open Systems and CICS for Windows. On other CICS family members, it establishes a client/server code page pair by means of a single number.

The DFHCNV TYPE=KEY macro

The DFHCNV TYPE=KEY macro indicates the start of conversions that are to be applied to a key. It is applicable only to an FC entry. You do not require this macro if access is only by RRN or RBA. If access is by key but no TYPE=KEY statement is present, CICS does not convert the key. You must also provide matching conversion details for the key as part of each SELECT that applies to this file. Otherwise, CICS might return an INVREQ condition on the file control EXEC request. When used, this should be the first statement in an ENTRY, and must be followed by one or more TYPE=FIELD statements.

Syntax

```
DFHCNV TYPE=KEY
```

The DFHCNV TYPE=SELECT macro

This macro is used to declare the selection criteria for a particular conversation.

Following each TYPE=SELECT instruction, is a set of TYPE=FIELD instructions that define the fields that are to be converted. Every TYPE=SELECT macro must be followed by at least one TYPE=FIELD macro.

Each SELECT statement compares the data with a given value to see whether it should be subject to a subsequent field conversion. You can have as many such comparisons as are necessary for each resource type and name.

If the data specification does not match the data that is in the record, the program skips to the next TYPE=SELECT, either until it finds one that does match and no further user data is converted; or until it gets to the OPTION=DEFAULT.

A default conversion must be available to be applied when no other is; this must be the last definition for each ENTRY.

Syntax

```
DFHCNV TYPE=SELECT ,OPTION={DEFAULT | COMPARE }
[,OFFSET=nnn] [,DATA=’dd...dd’] [,XDATA=’xx...xx’]
```

The options are defined as follows:

OPTION
States the basic selection options. CICS expects one of the following:

COMPARE
Indicates that the data should be converted by using the following field definitions only if it satisfies the defined comparison.

DEFAULT
Indicates that the data should all be converted by using the following fields if the previous SELECT COMPARE has not been chosen. Every resource must have an OPTION=DEFAULT entry as the last TYPE=SELECT option for that resource type and name; so you should have one TYPE=SELECT, OPTION=DEFAULT for each TYPE=ENTRY instruction.

OFFSET
You specify this option with COMPARE only, to define the byte offset in the record at which CICS should make the comparison. The maximum value is **65535**. You should specify this for all fields.

DATA You specify this option with COMPARE only, to define the data against which a comparison is to be made. This can be up to 254 contiguous SBCS character fields; not binary fields. The data that you specify for comparison here must belong to the code page that you specify in this entry's SRVERCP. If not, CICS makes unpredictable conversions. CICS converts this data from the SRVERCP code page to the CLINTCP code page before comparison (because the incoming data is in the code page of CLINTCP).

XDATA
You specify this option with COMPARE only, to define the hexadecimal data against which a comparison is to be made. This can be up to 254 hexadecimal digits (127 bytes). The length must be a whole number of bytes. CICS compares the incoming request against this hexadecimal field without conversion.

The DFHCNV TYPE=FIELD macro

The TYPE=FIELD macro specifies the position of a field, and the type and length of the data that it contains. One such macro must exist for each field that is in a resource record.

Syntax

DFHCNV TYPE=FIELD ,OFFSET=nnnn ,DATATYP={CHARACTER | PD | BINARY | USERDATA | GRAPHIC | NUMERIC} [,USRTYPE=xx] ,DATALEN=nnnn [,SOSI=YES | NO] [,LAST=YES]

The options are defined as follows:

OFFSET=*nnnn*
You specify the byte offset in the record at which the conversion should start, up to a maximum value of **65535**.

Note: For TYPE=KEY conversions, this is the byte offset from the start of the key, **not** from the start of the record.

DATATYP
The type of conversion that is to be done. CICS expects one of the following:

CHARACTER
Character fields are converted from SRVRCP to CLINTCP or from CLINTCP to SRVRCP, as required.

PD CICS does not convert packed decimal fields.

BINARY
No conversion occurs for binary fields.

USERDATA
The data conversion is done by user replaceable conversion program DFHUCNV. The USRTYPE option that is shown below allows you to pass a number to DFHUCNV that indicates how the data is to be converted.

GRAPHIC

Graphic fields contain Multi-Byte Character Set (MBCS) characters only. CICS adds a shift-out (SO) character (X'0E') to the start of the data, and a shift-in (SI) character ('0F') to the end of the data before passing it to iconv to be converted.

NUMERIC

CICS converts these fields between the local byte ordering and the byte order of the requesting system. Byte order information of the requesting system flows with the function shipping request if the requesting system is CICS on Open Systems or CICS for Windows. Otherwise the local region assumes the remote system is little endian and swaps the bytes as required. For more information about byte ordering, refer to "Numeric data conversion considerations" on page 149.

USRTYPE

The value given here is made available to the user-replaceable conversion program DFHUCNV. The values that you provide can be in the range 80 through 128 (X'50' through X'80'). The default value is 80 (X'50'). If more than one type of user-defined conversation is possible, you can use this value to specify to DFHUCNV what conversion is needed for each field.

The option is ignored if DATATYP=USERDATA is not specified.

DATALEN

The length of the data field that is to be converted, in bytes, up to a maximum value of **65535**. For variable length fields, specify the maximum possible length. For NUMERIC fields, only lengths 2 and 4 are valid.

SOSI This option is supported only for compatibility with other CICS products.

LAST=YES

Indicates that this definition is the last field for this TYPE=SELECT or TYPE=KEY statement.

The TYPE=ENTRY CLINTCP operand overrides the TYPE=INITIAL CLINTCP operand.

The DFHCNV TYPE=FINAL macro

The purpose of this macro is to conclude the conversion table definition. It must occur only once, as the last definition.

Appendix B. The conversation state tables

The state tables provide information for writing a DTP program. They show which commands can be issued from each conversation state and the state transitions that can occur and the EIB fields that can be set as a result of issuing a command.

How to use the state tables

The commands that you can issue, coupled with the EIB flags that can be set after execution, are shown in column 1 of each table. The possible conversation states are shown across the top of the table. The states correspond to the columns of the table. The intersection of row (command and EIB flag) and column (state) represents the state transition, if any, that occurs when that command returning a particular EIB flag is issued in that state.

A number at an intersection indicates the state number of the next state. Other symbols represent other conditions, as follows:

/ This state change cannot occur.

EIB* The EIB flag is any one that has not been covered in earlier rows, or it is irrelevant (but see the note on EIBSIG if you want to use ISSUE SIGNAL).

ab The command is not valid in this state. Issuing a command in a state in which it is not valid usually causes an ATCV abend.

= Remains in current state.

E End of conversation.

The state numbers

The state numbers are defined as follows:

State 1: Allocated
State 2: Send
State 3: Pendreceive
State 4: Pendfree
State 5: Receive
State 6: Confreceive
State 7: Confsend
State 8: Conffree
State 9: Syncreceive
State 10: Syncsend
State 11: Syncfree
State 12: Free
State 13: Rollback

The state conversation table notes

The following notes apply to all three state tables:

[1] EIBSIG has been omitted. This is because its use is optional and is entirely a matter of agreement between the two conversation partners. In the worst case, it can occur at any time after every command that affects the EIB flags. However, when used for the purpose for which it was intended, it usually occurs after a SEND command. Its priority in the sequence of testing depends on the role that you give it in the application.

[2] RECEIVE NOTRUNCATE returns a zero value in EIBCOMPL to indicate that the user buffer was too small to contain all the data that was received from the partner transaction. Normally, you would continue to issue RECEIVE NOTRUNCATE commands until the last information of data is passed to you, which is indicated by EIBCOMPL EIB* FF. If NOTRUNCATE is not specified, and the data area that is specified by the RECEIVE command is too small to contain all the data received, CICS truncates the data and sets the LENGERR condition.

[3] Equivalent to SEND INVITE WAIT followed by RECEIVE.

[4] Equivalent to SEND INVITE WAIT [FROM] followed by RECEIVE.

[5] Equivalent to SEND LAST WAIT followed by FREE.

[6] Equivalent to WAIT followed by RECEIVE.

[7] Before a CONVID is allocated, no conversation occurs, therefore no conversation state exists. The EXEC CICS ALLOCATE command is not shown in the tables. After ALLOCATE is successful, the front-end transaction starts the new conversation in **allocated**.

[8] ISSUE SIGNAL sets the partner’s EIBSIG flag unless it is entering the free state. ISSUE SIGNAL when sent in other states can also set the partner’s EIBSIG flag, the behavior being determined by the underlying SNA implementation.

[9] The back-end transaction starts in **receive** after the front-end transaction has issued CONNECT PROCESS.

[10] No data can be included with SEND CONFIRM.

[11] These commands cause CICS to return the INVREQ condition. This is to conform to the APPC architecture.

[12] The commands SYNCPOINT and SYNCPOINT ROLLBACK do not relate to any particular conversation. They are propagated on all the synchronization level 2 conversations that are currently active for the task.

[13] The state of each conversation after rollback depends on several factors:

- The system with which you are communicating. Some earlier versions of CICS handle rollback differently from how CICS handles it now.
- The conversation state at the beginning of the current distributed unit of work. This state is the one that is adopted in accordance with the APPC architecture. CICS follows the architecture.

A conversation might be in **free** after rollback if it has been terminated abnormally because of conversation failure or deallocate abend being received.

After a sync point or rollback, it is advisable to determine the conversation state before issuing any further commands against the conversation.

[14] This results, not in an ATCV abend, but in an INVREQ return code.

[15] This causes an A30A abend, not an ATCV.

[16] Although ISSUE PREPARE can return with the conversation in either **syncsend**, **syncreceive**, or **syncfree**, the only commands that are allowed on that conversation following an ISSUE PREPARE are SYNCPOINT and SYNCPOINT ROLLBACK. All other commands abend ATCV.

[17] Following an ISSUE ABEND of a synchronization level 2 conversation, all other synchronization level 2 conversations become state 13.

For more information, see:

- “Writing programs for CICS DTP” on page 286
- “Synchronization level 0 conversation state table”
- “Synchronization level 1 conversation state table” on page 335
- “Synchronization level 2 conversation state table” on page 338
- The *TXSeries for Multiplatforms Application Programming Reference*

Synchronization level 0 conversation state table

Table 55. Synchronization level 0 conversation state table (states 1 to 8)

Command issued, EIB flag returned[1]	**State 1**	**State 2**	**State 3**	**State 4**	**State 5**	**State 6**	**Satet 7**	**State 8**
CONNECT PROCESS, EIBERR + EIBFREE	12	ab	ab	ab	ab	/	/	/
CONNECT PROCESS[9], EIB*	2	ab	ab	ab	ab	/	/	/
SEND (any valid form), EIBERR + EIBFREE	ab	12	ab	ab	ab	/	/	/
SEND (any valid form), EIBERR	ab	5	ab	ab	ab	/	/	/
SEND INVITE WAIT, EIB*	ab	5	ab	ab	ab	/	/	/
SEND INVITE, EIB*	ab	5	ab	ab	ab	/	/	/
SEND LAST WAIT, EIB*	ab	12	ab	ab	ab	/	/	/
SEND LAST, EIB*	ab	4	ab	ab	ab	/	/	/
SEND WAIT, EIB*	ab	=	ab	ab	ab	/	/	/
SEND, EIB*	ab	=	ab	ab	ab	/	/	/
RECEIVE, EIBERR + EIBFREE	ab	12[3]	12[6]	ab	12	/	/	/
RECEIVE, EIBERR	ab	5[3]	5[6]	ab	=	/	/	/
RECEIVE, EIBFREE	ab	12[3]	12[6]	ab	12	/	/	/
RECEIVE, EIBRECV	ab	5[3]	5[6]	ab	=	/	/	/
RECEIVE, EIB*	ab	=[3]	2[6]	ab	2	/	/	/
CONVERSE, EIBERR + EIBFREE	ab	12[3]	12[6]	ab	12	/	/	/
CONVERSE, EIBERR	ab	5[3]	5[6]	ab	=	/	/	/
CONVERSE, EIBFREE	ab	12[3]	12[6]	ab	12	/	/	/
CONVERSE, EIBRECV	ab	5[3]	5[6]	ab	=	/	/	/
CONVERSE NOTRUNCATE, EIBCOMPL[2]	ab	5[3]	5[6]	ab	=	/	/	/
CONVERSE, EIB*	ab	=[3]	2[6]	ab	2	/	/	/
ISSUE ERROR, EIBFREE	ab	12	12	ab	12	/	/	/
ISSUE ERROR, EIBERR	ab	5	5	ab	5	/	/	/
ISSUE ERROR, EIB*	ab	2	2	ab	2	/	/	/
ISSUE ABEND, EIB*	ab	12	12	12	12	/	/	/
ISSUE SIGNAL[8], EIB*	ab	=	=	ab	=	/	/	/
WAIT CONVID, EIB*	ab	=	5	12	ab	/	/	/
FREE, EIB*	E	E[5]	ab	E	ab	/	/	/

Table 55. Synchronization level 0 conversation state table (states 1 to 8) (continued)

Command issued, EIB flag returned[1]	State 1	State 2	State 3	State 4	State 5	State 6	Satet 7	State 8
Key to states: State 1: Allocated State 2: Send State 3: Pendreceive State 4: Pendfree State 5: Receive State 6: Confreceive State 7: Confsend State 8: Conffree Footnotes are given in “The state conversation table notes” on page 331.								

Table 56. Synchronization level 0 conversation state table (states 9 to 13)

Command issued, EIB flag returned[1]	State 9	State 10	State 11	State 12	State 13	Command returns
CONNECT PROCESS, EIBERR + EIBFREE	/	/	/	ab	/	Immediately
CONNECT PROCESS[9], EIB*	/	/	/	=	/	Immediately
SEND (any valid form), EIBERR + EIBFREE	/	/	/	ab	/	After error detected
SEND (any valid form), EIBERR	/	/	/	ab	/	After error detected
SEND INVITE WAIT, EIB*	/	/	/	ab	/	After data flows
SEND INVITE, EIB*	/	/	/	ab	/	After data buffered
SEND LAST WAIT, EIB*	/	/	/	ab	/	After data flows
SEND LAST, EIB*	/	/	/	ab	/	After data buffered
SEND WAIT, EIB*	/	/	/	ab	/	After data flows
SEND, EIB*	/	/	/	ab	/	After data buffered
RECEIVE, EIBERR + EIBFREE	/	/	/	ab	/	After error detected
RECEIVE, EIBERR	/	/	/	ab	/	After error detected
RECEIVE, EIBFREE	/	/	/	ab	/	After error detected
RECEIVE, EIBRECV	/	/	/	ab	/	When data available
RECEIVE, EIB*	/	/	/	ab	/	When data available
CONVERSE, EIBERR + EIBFREE	/	/	/	ab	/	After error detected
CONVERSE, EIBERR	/	/	/	ab	/	After error detected
CONVERSE, EIBFREE	/	/	/	ab	/	After error detected
CONVERSE, EIBRECV	/	/	/	ab	/	When data available
CONVERSE NOTRUNCATE, EIBCOMPL[2]	/	/	/	ab	/	When data available
CONVERSE, EIB*	/	/	/	ab	/	When data available
ISSUE ERROR, EIBFREE	/	/	/	ab	/	After response from partner
ISSUE ERROR, EIBERR	/	/	/	ab	/	After response from partner
ISSUE ERROR, EIB*	/	/	/	ab	/	After response from partner
ISSUE ABEND, EIB*	/	/	/	ab	/	Immediately
ISSUE SIGNAL[8], EIB*	/	/	/	ab	/	Immediately

Table 56. Synchronization level 0 conversation state table (states 9 to 13) (continued)

Command issued, EIB flag returned[1]	State 9	State 10	State 11	State 12	State 13	Command returns
WAIT CONVID, EIB*	/	/	/	ab	/	Immediately
FREE, EIB*	/	/	/	E	/	Immediately

Key to states:

State 9: Syncreceive
State 10: Syncsend
State 11: Syncfree
State 12: Free
State 13: Rollback

Footnotes are given in “The state conversation table notes” on page 331.

Synchronization level 1 conversation state table

Table 57. Synchronization level 1 conversation state table (states 1 to 8)

Command issued, EIB flag returned[1]	State 1	State 2	State 3	State 4	State 5	State 6	State 7	State 8
CONNECT PROCESS, EIBERR + EIBFREE	12	ab	ab	ab	ab	ab	ab	ab
CONNECT PROCESS[9], EIB*	2	ab	ab	ab	ab	ab	ab	ab
SEND (any valid form), EIBERR + EIBFREE	ab	12	12	12	ab	ab	ab	/
SEND (any valid form), EIBERR	ab	5	5	5	ab	ab	ab	ab
SEND INVITE WAIT, EIB*	ab	5	ab	ab	ab	ab	ab	ab
SEND INVITE CONFIRM, EIB*	ab	5	ab	ab	ab	ab	ab	ab
SEND INVITE, EIB*	ab	3	ab	ab	ab	ab	ab	ab
SEND LAST WAIT, EIB*	ab	12	ab	ab	ab	ab	ab	ab
SEND LAST CONFIRM, EIB*	ab	12	ab	ab	ab	ab	ab	ab
SEND LAST, EIB*	ab	4	ab	ab	ab	ab	ab	ab
SEND WAIT, EIB*	ab	=	ab	ab	ab	ab	ab	ab
SEND CONFIRM, EIB*	ab	=	5	12[10]	ab	ab	ab	ab
SEND, EIB*	ab	=	ab	ab	ab	ab	ab	ab
RECEIVE, EIBERR + EIBFREE	ab	12[3]	12[6]	ab	12	ab	ab	ab
RECEIVE, EIBERR	ab	5[3]	5[6]	ab	=	ab	ab	ab
RECEIVE, EIBCONF + EIBFREE	ab	8[3]	8[6]	ab	8	ab	ab	ab
RECEIVE, EIBCONF + EIBRECV	ab	6[3]	6[6]	ab	6	ab	ab	ab
RECEIVE, EIBCONF	ab	7[3]	7[6]	ab	7	ab	ab	ab
RECEIVE, EIBFREE	ab	12[3]	12[6]	ab	12	ab	ab	ab
RECEIVE, EIBRECV	ab	5[3]	5[6]	ab	5	ab	ab	ab
RECEIVE, EIB*	ab	=[3]	2[6]	ab	2	ab	ab	ab
CONVERSE[4], EIBERR + EIBFREE	ab	12[3]	12[6]	ab	12	ab	ab	ab
CONVERSE[4], EIBERR	ab	5[3]	5[6]	ab	=	ab	ab	ab

Table 57. Synchronization level 1 conversation state table (states 1 to 8) (continued)

Command issued, EIB flag returned[1]	State 1	State 2	State 3	State 4	State 5	State 6	State 7	State 8
CONVERSE[4], EIBCONF + EIBFREE	ab	8[3]	8[6]	ab	8	ab	ab	ab
CONVERSE[4], EIBCONF + EIBRECV	ab	6[3]	6[6]	ab	6	ab	ab	ab
CONVERSE[4], EIBCONF	ab	7[3]	7[6]	ab	7	ab	ab	ab
CONVERSE[4], EIBFREE	ab	12[3]	12[6]	ab	12	ab	ab	ab
CONVERSE[4], EIBRECV	ab	5[3]	5[6]	ab	5	ab	ab	ab
CONVERSE[4] NOTRUNCATE, EIBCOMPL[2]	ab	5[3]	5[6]	ab	5	ab	ab	ab
CONVERSE[4], EIB*	ab	=[3]	2[6]	ab	2	ab	ab	ab
ISSUE CONFIRMATION, EIB*	ab	ab	ab	ab	ab	5	2	12
ISSUE ERROR, EIBFREE	ab	12	12	ab	12	12	12	12
ISSUE ERROR, EIBERR	ab	5	5	ab	5	5	5	5
ISSUE ERROR, EIB*	ab	2	2	ab	2	2	2	2
ISSUE ABEND, EIB*	ab	12	12	12	12	12	12	12
ISSUE SIGNAL[8], EIB*	ab	=	=	ab	=	=	=	=
WAIT CONVID, EIB*	ab	=	5	12	ab	ab	ab	ab
FREE, EIB*	E	E[5]	ab	E	ab	ab	ab	ab

Key to states:

State 1: Allocated
State 2: Send
State 3: Pendreceive
State 4: Pendfree
State 5: Receive
State 6: Confreceive
State 7: Confsend
State 8: Conffree

Footnotes are given in “The state conversation table notes” on page 331.

Table 58. Synchronization level 1 conversation state table (states 9 to 13)

Command issued, EIB flag returned[1]	State 9	State 10	State 11	Satte 12	State 13	Command returns
CONNECT PROCESS, EIBERR + EIBFREE	/	/	/	ab	/	Immediately
CONNECT PROCESS[9], EIB*	/	/	/	ab	/	Immediately
SEND (any valid form), EIBERR + EIBFREE	/	/	/	ab	/	After error flow detected
SEND (any valid form), EIBERR	/	/	/	ab	/	After error flow detected
SEND INVITE WAIT, EIB*	/	/	/	ab	/	After data flows
SEND INVITE CONFIRM, EIB*	/	/	/	ab	/	After response from partner
SEND INVITE, EIB*	/	/	/	ab	/	After data buffered
SEND LAST WAIT, EIB*	/	/	/	ab	/	After data flows
SEND LAST CONFIRM, EIB*	/	/	/	ab	/	After response from partner
SEND LAST, EIB*	/	/	/	ab	/	After data buffered

Table 58. Synchronization level 1 conversation state table (states 9 to 13) (continued)

Command issued, EIB flag returned[1]	State 9	State 10	State 11	Satte 12	State 13	Command returns
SEND WAIT, EIB*	/	/	/	ab	/	After data flows
SEND CONFIRM, EIB*	/	/	/	ab	/	After response from partner
SEND, EIB*	/	/	/	ab	/	After data buffered
RECEIVE, EIBERR + EIBFREE	/	/	/	ab	/	After error detected
RECEIVE, EIBERR	/	/	/	ab	/	After error detected
RECEIVE, EIBCONF + EIBFREE	/	/	/	ab	/	After confirm flow detected
RECEIVE, EIBCONF + EIBRECV	/	/	/	ab	/	After confirm flow detected
RECEIVE, EIBCONF	/	/	/	ab	/	After confirm flow detected
RECEIVE, EIBFREE	/	/	/	ab	/	After error detected
RECEIVE, EIBRECV	/	/	/	ab	/	When data available
RECEIVE, EIB*	/	/	/	ab	/	When data available
CONVERSE[4], EIBERR + EIBFREE	/	/	/	ab	/	After error detected
CONVERSE[4], EIBERR	/	/	/	ab	/	After error detected
CONVERSE[4], EIBCONF + EIBFREE	/	/	/	ab	/	After confirm flow detected
CONVERSE[4], EIBCONF + EIBRECV	/	/	/	ab	/	After confirm flow detected
CONVERSE[4], EIBCONF	/	/	/	ab	/	After confirm flow detected
CONVERSE[4], EIBFREE	/	/	/	ab	/	After error detected
CONVERSE[4], EIBRECV	/	/	/	ab	/	When data available
CONVERSE[4] NOTRUNCATE, EIBCOMPL[2]	/	/	/	ab	/	When data available
CONVERSE[4], EIB*	/	/	/	ab	/	When data available
ISSUE CONFIRMATION, EIB*	/	/	/	ab	/	Immediately
ISSUE ERROR, EIBFREE	/	/	/	ab	/	After response from partner
ISSUE ERROR, EIBERR	/	/	/	ab	/	After response from partner
ISSUE ERROR, EIB*	/	/	/	ab	/	After response from partner
ISSUE ABEND, EIB*	/	/	/	ab	/	Immediately
ISSUE SIGNAL[8], EIB*	/	/	/	ab	/	Immediately
WAIT CONVID, EIB*	/	/	/	ab	/	Immediately
FREE, EIB*	/	/	/	E	/	Immediately

Key to states:

State 9: Syncreceive
State 10: Syncsend
State 11: Syncfree
State 12: Free
State 13: Rollback

Footnotes are given in “The state conversation table notes” on page 331.

Synchronization level 2 conversation state table

Table 59. Part 1 of Synchronization level 2 conversation state table (states 1 to 8)

Command issued, EIB flag returned[1]	1	2	3	4	5	6	7	8
CONNECT PROCESS, EIBERR + EIBFREE	12	ab	ab	ab	ab	ab	ab	ab
CONNECT PROCESS[9], EIB*	2	ab	ab	ab	ab	ab	ab	ab
SEND (any valid form), EIBERR + EIBSYNRB	ab	13	13	ab	ab	ab	ab	ab
SEND (any valid form), EIBERR + EIBFREE	ab	12	12	ab	ab	ab	ab	ab
SEND (any valid form), EIBERR	ab	5	5	ab	ab	ab	ab	ab
SEND INVITE WAIT, EIB*	ab	5	ab	ab	ab	ab	ab	ab
SEND INVITE CONFIRM, EIB*	ab	5	ab	ab	ab	ab	ab	ab
SEND INVITE, EIB*	ab	3	ab	ab	ab	ab	ab	ab
SEND LAST WAIT[11], EIB*	ab	ab[11]	ab[11]	ab	ab	ab	ab	ab
SEND LAST CONFIRM[11], EIB*	ab	ab	ab	ab	ab	ab	ab	ab
SEND LAST, EIB*	ab	4	ab	ab	ab	ab	ab	ab
SEND WAIT, EIB*	ab	=	ab	ab	ab	ab	ab	ab
SEND CONFIRM, EIB*	ab	=	5[10]	ab[11]	ab	ab	ab	ab
SEND, EIB*	ab	=	ab	ab	ab	ab	ab	ab
RECEIVE, EIBERR + EIBSYNRB	ab	13[3]	13[6]	ab	13	ab	ab	ab
RECEIVE, EIBERR + EIBFREE	ab	12[3]	12[6]	ab	12	ab	ab	ab
RECEIVE, EIBERR	ab	5[3]	5[6]	ab	=	ab	ab	ab
RECEIVE, EIBSYNC + EIBFREE	ab	11[3]	11[6]	ab	11	ab	ab	ab
RECEIVE, EIBSYNC + EIBRECV	ab	9[3]	9[6]	ab	9	ab	ab	ab
RECEIVE, EIBSYNC	ab	10[3]	10[6]	ab	10	ab	ab	ab
RECEIVE, EIBCONF + EIBFREE	ab	8[3]	8[6]	ab	8	ab	ab	ab
RECEIVE, EIBCONF + EIBRECV	ab	6[3]	6[6]	ab	6	ab	ab	ab
RECEIVE, EIBCONF	ab	7[3]	7[6]	ab	7	ab	ab	ab
RECEIVE, EIBFREE	ab	12[3]	12[6]	ab	12	ab	ab	ab
RECEIVE, EIBRECV	ab	5[3]	5[6]	ab	=	ab	ab	ab
RECEIVE, EIB*	ab	=[3]	2[6]	ab	2	ab	ab	ab
CONVERSE[4], EIBERR + EIBSYNRB	ab	13[3]	13[6]	ab	13	ab	ab	ab
CONVERSE[4], EIBERR + EIBFREE	ab	12[3]	12[6]	ab	12	ab	ab	ab
CONVERSE[4], EIBERR	ab	5[3]	5[6]	ab	=	ab	ab	ab
CONVERSE[4], EIBSYNC + EIBFREE	ab	11[3]	11[6]	ab	11	ab	ab	ab
CONVERSE[4], EIBSYNC + EIBRECV	ab	9[3]	9[6]	ab	9	ab	ab	ab
CONVERSE[4], EIBSYNC	ab	10[3]	10[6]	ab	10	ab	ab	ab
CONVERSE[4], EIBCONF + EIBFREE	ab	8[3]	8[6]	ab	8	ab	ab	ab

Table 59. Part 1 of Synchronization level 2 conversation state table (states 1 to 8) (continued)

Command issued, EIB flag returned[1]	1	2	3	4	5	6	7	8
CONVERSE[4], EIBCONF + EIBRECV	ab	6[3]	6[6]	ab	6	ab	ab	ab
CONVERSE[4], EIBCONF	ab	7[3]	7[6]	ab	7	ab	ab	ab
CONVERSE[4], EIBFREE	ab	12[3]	12[6]	ab	12	ab	ab	ab
CONVERSE[4], EIBRECV	ab	5[3]	5[6]	ab	=	ab	ab	ab

Key to states:

State 1: Allocated
State 2: Send
State 3: Pendreceive
State 4: Pendfree
State 5: Receive
State 6: Confreceive
State 7: Confsend
State 8: Conffree

Footnotes are given in “The state conversation table notes” on page 331.

Table 60. Part 1 of Synchronization level 2 conversation state table (states 9 to 13)

Command issued, EIB flag returned[1]	State 9	State 10	State 11	State 12	State 13	Command returns
CONNECT PROCESS, EIBERR + EIBFREE	ab	ab	ab	ab	ab	Immediately
CONNECT PROCESS[9], EIB*	ab	ab	ab	ab	ab	Immediately
SEND (any valid form), EIBERR + EIBSYNRB	ab	ab	ab	ab	ab	After error flow detected
SEND (any valid form), EIBERR + EIBFREE	ab	ab	ab	ab	ab	After error flow detected
SEND (any valid form), EIBERR	ab	ab	ab	ab	ab	After error flow detected
SEND INVITE WAIT, EIB*	ab	ab	ab	ab	ab	After data flow
SEND INVITE CONFIRM, EIB*	ab	ab	ab	ab	ab	After response from partner
SEND INVITE, EIB*	ab	ab	ab	ab	ab	After data buffered
SEND LAST WAIT[11], EIB*	ab	ab	ab	ab	ab	After data flows
SEND LAST CONFIRM[11], EIB*	ab	ab	ab	ab	ab	After response from partner
SEND LAST, EIB*	ab	ab	ab	ab	ab	After data buffered
SEND WAIT, EIB*	ab	ab	ab	ab	ab	After data flows
SEND CONFIRM, EIB*	ab	ab	ab	ab	ab	After response from partner
SEND, EIB*	ab	ab	ab	ab	ab	After data buffered
RECEIVE, EIBERR + EIBSYNRB	ab	ab	ab	ab	ab	After data flows
RECEIVE, EIBERR + EIBFREE	ab	ab	ab	ab	ab	After error detected
RECEIVE, EIBERR	ab	ab	ab	ab	ab	After error detected
RECEIVE, EIBSYNC + EIBFREE	ab	ab	ab	ab	ab	After sync flow detected

Table 60. Part 1 of Synchronization level 2 conversation state table (states 9 to 13) (continued)

Command issued, EIB flag returned[1]	State 9	State 10	State 11	State 12	State 13	Command returns
RECEIVE, EIBSYNC + EIBRECV	ab	ab	ab	ab	ab	After sync flow detected
RECEIVE, EIBSYNC	ab	ab	ab	ab	ab	After sync flow detected
RECEIVE, EIBCONF + EIBFREE	ab	ab	ab	ab	ab	After confirm flow detected
RECEIVE, EIBCONF + EIBRECV	ab	ab	ab	ab	ab	After confirm flow detected
RECEIVE, EIBCONF	ab	ab	ab	ab	ab	After confirm flow detected
RECEIVE, EIBFREE	ab	ab	ab	ab	ab	After error flow detected
RECEIVE, EIBRECV	ab	ab	ab	ab	ab	When data available
RECEIVE, EIB*	ab	ab	ab	ab	ab	When data available
CONVERSE[4], EIBERR + EIBSYNRB	ab	ab	ab	ab	ab	After data flows
CONVERSE[4], EIBERR + EIBFREE	ab	ab	ab	ab	ab	After error detected
CONVERSE[4], EIBERR	ab	ab	ab	ab	ab	After error detected
CONVERSE[4], EIBSYNC + EIBFREE	ab	ab	ab	ab	ab	After sync flow detected
CONVERSE[4], EIBSYNC + EIBRECV	ab	ab	ab	ab	ab	After sync flow detected
CONVERSE[4], EIBSYNC	ab	ab	ab	ab	ab	After sync flow detected
CONVERSE[4], EIBCONF + EIBFREE	ab	ab	ab	ab	ab	After confirm flow detected
CONVERSE[4], EIBCONF + EIBRECV	ab	ab	ab	ab	ab	After confirm flow detected
CONVERSE[4], EIBCONF	ab	ab	ab	ab	ab	After confirm flow detected
CONVERSE[4], EIBFREE	ab	ab	ab	ab	ab	After error flow detected
CONVERSE[4], EIBRECV	ab	ab	ab	ab	ab	When data available

Key to states:

State 9: Syncreceive
State 10: Syncsend
State 11: Syncfree
State 12: Free
State 13: Rollback

Footnotes are given in “The state conversation table notes” on page 331.

Table 61. Part 2 of Synchronization level 2 conversation state table (states 1 to 8)

Command issued, EIB flag returned[1]	**State 1**	**State 2**	**State 3**	**State 4**	**State 5**	**State 6**	**State 7**	**State 8**
ISSUE CONFIRMATION, EIB*	ab	ab	ab	ab	ab	5	2	12
ISSUE ERROR, EIBFREE	ab	12	12	ab	12	12	12	12
ISSUE ERROR, EIBERR + EIBSYNRB	ab	13	13	ab	13	13	13	13
ISSUE ERROR, EIBERR	ab	5	5	ab	5	5	5	5
ISSUE ERROR, EIB*	ab	2	2	ab	2	2	2	2
ISSUE ABEND[17], EIB*	ab	12	12	12	12	12	12	12
ISSUE SIGNAL[8], EIB*	ab	=	=	ab	=	=	=	=
ISSUE PREPARE, EIBERR + EIBSYNRB	ab[14]	13	13	13	ab[14]	ab[14]	ab[14]	ab[14]
ISSUE PREPARE, EIBERR + EIBFREE	ab[14]	12	12	12	ab[14]	ab[14]	ab[14]	ab[14]
ISSUE PREPARE, EIBERR	ab[14]	5	5	5	ab[14]	ab[14]	ab[14]	ab[14]
ISSUE PREPARE, EIB*	ab[14]	10[16]	9[16]	11[16]	ab[14]	ab[14]	ab[14]	ab[14]
SYNCPOINT[12], EIBRLDBK	=	2 or 5[13]	2 or 5[13]	2 or 5[13]	ab[15]	ab[15]	ab[15]	ab[15]
SYNCPOINT[12], EIB*	=	=	5	12	ab[15]	ab[15]	ab[15]	ab[15]
SYNCPOINT ROLLBACK[12], EIB	=	2 or 5[13]	2 or 5[13]	2 or 5[13]	2 or 5[13]	2 or 5[13]	2 or 5[13]	2 or 5[13]
WAIT CONVID, EIB*	ab	=	5	=	ab	ab	ab	ab
FREE, EIB*	E	=[11]	=[11]	=	ab	ab	ab	ab
CONVERSE[4] NOTRUNCATE, EIBCOMPL (5)	ab	5[3]	5[6]	ab	=	ab	ab	ab
CONVERSE[4], EIB*	ab	=[3]	2[6]	ab	2	ab	ab	ab

Key to states:

State 1: Allocated
State 2: Send
State 3: Pendreceive
State 4: Pendfree
State 5: Receive
State 6: Confreceive
State 7: Confsend
State 8: Conffree

Footnotes are given in “The state conversation table notes” on page 331.

Table 62. Part 2 of Synchronization level 2 conversation state table (states 9 to 13)

Command issued, EIB flag returned[1]	State 9	State 10	State 11	State 12	State 13	Command returns
ISSUE CONFIRMATION, EIB*	ab	ab	ab	ab	ab	Immediately
ISSUE ERROR, EIBFREE	12	12	12	ab	ab	After response from partner
ISSUE ERROR, EIBERR + EIBSYNRB	13	13	13	ab	ab	After response from partner
ISSUE ERROR, EIBERR	5	5	5	ab	ab	After response from partner
ISSUE ERROR, EIB*	2	2	2	ab	ab	After response from partner
ISSUE ABEND[17], EIB*	12	12	12	ab	ab	Immediately
ISSUE SIGNAL[8], EIB*	=	=	=	ab	ab	Immediately
ISSUE PREPARE, EIBERR + EIBSYNRB	ab[14]	ab[14]	ab[14]	ab[14]	ab[14]	After response from partner
ISSUE PREPARE, EIBERR + EIBFREE	ab[14]	ab[14]	ab[14]	ab[14]	ab[14]	After response from partner
ISSUE PREPARE, EIBERR	ab[14]	ab[14]	ab[14]	ab[14]	ab[14]	After error detected
ISSUE PREPARE, EIB*	ab[14]	ab[14]	ab[14]	ab[14]	ab[14]	After response from partner
SYNCPOINT[12], EIBRLDBK	2 or 5[13]	2 or 5[13]	2 or 5[13]	=	ab[15]	After response from partner
SYNCPOINT[12], EIB*	5	2	12	=	ab[15]	After response from partner
SYNCPOINT ROLLBACK[12], EIB	2 or 5[13]	2 or 5[13]	2 or 5[13]	=	2 or 5[13]	After rollback across LUW
WAIT CONVID, EIB*	ab	ab	ab	ab	ab	Immediately
FREE, EIB*	ab	ab	ab	E	ab	Immediately
CONVERSE[4] NOTRUNCATE, EIBCOMPL (5)	ab	ab	ab	ab	ab	When data available
CONVERSE[4], EIB*	ab	ab	ab	ab	ab	When data available

Key to states:

State 9: Syncreceive
State 10: Syncsend
State 11: Syncfree
State 12: Free
State 13: Rollback

Footnotes are given in "The state conversation table notes" on page 331.

Appendix C. Migrating DTP applications

Two areas are discussed in this chapter:

1. The differences between TXSeries for Multiplatforms application programming and CICS/MVS 2.1 application programming. These differences are discussed to allow you to modify non-TXSeries for Multiplatforms applications so that they can run in TXSeries for Multiplatforms. This is referred to as *migration*.
2. Writing applications in such a way as to allow them to run in TXSeries for Multiplatforms and in other CICS products. This is referred to as *portability*.

Migration and portability of DTP applications can be easily achieved through careful analysis of the differences between some of the CICS commands and network protocol. Those differences are discussed in the following sections.

Migrating to LU 6.2 APPC mapped conversations

TXSeries for Multiplatforms uses functions to implement APPC (LU 6.2) mapped conversations. If the application that you are converting was not written for LU 6.2 mapped conversations (for example, LU 6.1), you will need to convert it to LU 6.2 mapped conversations before migrating. Most of the CICS intercommunications documentation tell you how to convert for LU 6.2 conversations.

If you are converting MVS LU 6.1 DTP applications to run in TXSeries for Multiplatforms, you must first refer to the MVS books to convert to LU 6.2 before you convert that application for TXSeries for Multiplatforms.

Differences between SNA and TCP/IP for DTP

Apart from configuration, few differences exist between TXSeries for Multiplatforms and IBM mainframe-based CICS. Those differences are:

1. The EXEC CICS ALLOCATE command PROFILE(name) option.

   PROFILE(name) specifies the name (maximum of eight characters) of a mode that is defined to the SNA software on the local machine. This mode contains a set of session-processing options that CICS uses during the running of mapped commands for the remote region that is specified in the SYSID option. If the mode contains the name of a group of APPC sessions from which the conversation is to be allocated, a particular class of service can be selected.

   The PROFILE option is ignored over TCP/IP connections and by some versions of SNA.
2. The NOQUEUE and NOSUSPEND options have no effect on TCP/IP conversations because TCP/IP is not session based. Therefore, you will never get the SYSBUSY condition when using TCP/IP. When using TCP/IP, if a remote system is not available, SYSIDERR is returned by the CONNECT PROCESS command.
3. The network exchanges might be slightly different and you should not assume that state changes will occur. Always test the state of a conversation (use the STATE option, EXEC CICS EXTRACT ATTRIBUTES, or the EIB) before performing the next command.

Allocating a conversation

Allocating conversations works in two different ways, depending on which CICS platform you are using. In some CICS platforms, a session is allocated by the EXEC CICS ALLOCATE command and the remote transaction is attached by an EXEC CICS CONNECT PROCESS command. In other CICS platforms, including TXSeries for Multiplatforms:

1. The EXEC CICS ALLOCATE command only provides a conversation identifier (CONVID); it does not allocate a session.
2. The SNA MC_ALLOCATE verb is called from the EXEC CICS CONNECT PROCESS command instead of the EXEC CICS ALLOCATE command.
3. The SYSBUSY condition is returned by the EXEC CICS CONNECT PROCESS command rather than by the EXEC CICS ALLOCATE command.

This should be taken into consideration when the NOQUEUE or NOSUSPEND options are used with the EXEC CICS ALLOCATE command because sessions are not requested until the EXEC CICS CONNECT PROCESS command is used.

Use of conversation identifiers (CONVIDs)

In TXSeries for Multiplatforms, the CONVID that is returned by EXEC CICS ALLOCATE (or the back-end principal facility) is used for DTP only. You must not use the CONVID for any other purpose (such as naming Temporary Storage queues). The CONVID is valid only for the life of a transaction and bears no relationship to the SNA session that is in use.

State after backout

Following an EXEC CICS SYNCPOINT ROLLBACK, TXSeries for Multiplatforms returns synchronization level 2 conversations to the state that they were in at the start of the logical unit of work (LUW). Because of this, you cannot assume in which state a conversation will be after EXEC CICS SYNCPOINT ROLLBACK is used. In this case, use the EXEC CICS EXTRACT ATTRIBUTES command to determine the state of the conversation before performing the next command.

Terminating a synchronization level 2 conversation

A synchronization level 2 DTP application that is running on TXSeries for Multiplatforms, or the partner application of a TXSeries for Multiplatforms application that might be running on a different platform, must not issue any of the following commands:

- EXEC CICS FREE (from send or pendfree state)
- EXEC CICS SEND LAST WAIT
- EXEC CICS SEND CONFIRM (from pendfree state)
- EXEC CICS SEND LAST CONFIRM
- EXEC CICS WAIT (from pendfree state)

The correct way to terminate normally a synchronization level 2 conversation is to use the following sequence of commands:

- EXEC CICS SEND LAST
- EXEC CICS SYNCPOINT
- EXEC CICS FREE

The correct way to terminate abnormally a synchronization level 2 conversation is to use the following sequence of commands:

- EXEC CICS ISSUE ABEND
- EXEC CICS SYNCPOINT ROLLBACK
- EXEC CICS FREE

If you are migrating a synchronization level 2 application from another CICS platform, verify that it follows these rules.

Ensure also that the partner applications, which may not be TXSeries for Multiplatforms applications, also follow these rules.

The EXEC CICS WAIT and EXEC CICS SEND WAIT commands

Data is buffered in APPC systems to improve network efficiency. Buffered data is not received immediately by the partner transaction and the EXEC CICS WAIT and EXEC CICS SEND WAIT commands are used to flush this data.

Although the use of the WAIT command might cause a state change, it does **not** guarantee that the partner transaction is going to receive the data immediately. The reason for this is that data might still be buffered by the underlying network layers.

If a pair of transactions need to ensure timing considerations, you must use another mechanism besides WAIT.

Transaction identifiers

For outbound requests, up to 32 bytes are allowed, although it is expected that requests that are sent to another TXSeries for Multiplatforms region will be four bytes or less. The APPC architecture allows up to 64 bytes, but leaves each product free to set its own maximum. TXSeries for Multiplatforms complies by allowing 32 bytes, but this need concern you only if your TXSeries for Multiplatforms region is connected to a partner system that demands longer transaction identifiers.

For inbound requests, however, the transaction identifiers must be four bytes or less. TXSeries for Multiplatforms does not support inbound requests for transaction identifiers that are longer than four characters.

The slight difference between IBM mainframe-based CICS and TXSeries for Multiplatforms with the use of transaction identifiers, in that IBM mainframe-based CICS truncates down to four bytes while TXSeries for Multiplatforms does not.

Refer to the description of the PROCNAME attribute in the *TXSeries for Multiplatforms Application Programming Reference* to see how this is used.

Migration considerations for function shipping

If you use a transaction to route from one region to another, then use function shipping to route to another region (or back to the original region), the user ID that is used to access the remote resources is the user ID from the first region that initiated the function request. User IDs are considered only if the region that owns the remote resources has a **RemoteSysSecurity** value of “**trusted**” instead of “**local**”. For more information, see “Security and function shipping” on page 141.

Appendix D. Data conversion tables (CICS on Windows Systems and CICS for Solaris only)

Table 63 and Table 64 on page 351 show the permissible combinations of code pages that can be used to translate data from one encoding to another.

Table 65 on page 354 shows the permissible combinations of SBCS and DBCS code pages that can be used to produce further DBCS code pages.

Table 63. SBCS data conversion table

Code pages	Description
037 to 437	EBCDIC US English to PC-ASCII US
037 to 819	EBCDIC US English to ISO8859-1 Western European
037 to 850	EBCDIC US English to PC-ASCII Western European
037 to 860	EBCDIC US English to PC-ASCII Portuguese
037 to 863	EBCDIC US English to PC-ASCII Canadian French
273 to 437	EBCDIC German to PC-ASCII US
273 to 819	EBCDIC German to ISO8859-1 Western European
273 to 850	EBCDIC German to PC-ASCII Western European
277 to 437	EBCDIC Danish/Norwegian to PC-ASCII US
277 to 819	EBCDIC Danish/Norwegian to ISO8859-1 Western European
277 to 850	EBCDIC Danish/Norwegian to PC-ASCII Western European
277 to 865	EBCDIC Danish/Norwegian to PC-ASCII Scandinavian
278 to 437	EBCDIC Finnish/Swedish to PC-ASCII US
278 to 819	EBCDIC Finnish/Swedish to ISO8859-1 Western European
278 to 850	EBCDIC Finnish/Swedish to PC-ASCII Western European
278 to 865	EBCDIC Finnish/Swedish to PC-ASCII Scandinavian
280 to 437	EBCDIC Italian to PC-ASCII US
280 to 819	EBCDIC Italian to ISO8859-1 Western European
280 to 850	EBCDIC Italian to PC-ASCII Western European
284 to 437	EBCDIC Spanish to PC-ASCII US
284 to 819	EBCDIC Spanish to ISO8859-1 Western European
284 to 850	EBCDIC Spanish to PC-ASCII Western European
285 to 819	EBCDIC UK English to ISO8859-1 Western European
285 to 850	EBCDIC UK English to PC-ASCII Western European
285 to 437	EBCDIC UK English to PC-ASCII US
297 to 437	EBCDIC French to PC-ASCII US
297 to 819	EBCDIC French to ISO8859-1 Western European
297 to 850	EBCDIC French to PC-ASCII Western European
297 to 863	EBCDIC French to PC-ASCII Canadian French
420 to 864	EBCDIC Arabic to PC-ASCII Arabic
420 to 1046	EBCDIC Arabic to PC-ASCII Arabic

Table 63. SBCS data conversion table (continued)

Code pages	Description
420 to 1089	EBCDIC Arabic to ISO8859-6 Arabic
424 to 856	EBCDIC Hebrew to PC-ASCII Hebrew
424 to 862	EBCDIC Hebrew to PC-ASCII Hebrew
424 to 916	EBCDIC Hebrew to ISO8859-8 Hebrew
437 to 37	PC-ASCII US to EBCDIC US English
437 to 273	PC-ASCII US to EBCDIC German
437 to 277	PC-ASCII US to EBCDIC Danish/Norwegian
437 to 278	PC-ASCII US to EBCDIC Finnish/Swedish
437 to 280	PC-ASCII US to EBCDIC Italian
437 to 284	PC-ASCII US to EBCDIC Spanish
437 to 285	PC-ASCII US to EBCDIC UK English
437 to 297	PC-ASCII US to EBCDIC French
437 to 500	PC-ASCII US to EBCDIC International
437 to 819	PC-ASCII US to ISO8859-1 Western European
437 to 850	PC-ASCII US to PC-ASCII Western European
437 to 865	PC-ASCII US to PC-ASCII Scandinavian
437 to 1051	PC-ASCII US to ASCII roman8 for HP Western European
500 to 437	EBCDIC International to PC-ASCII US
500 to 819	EBCDIC International to ISO8859-1 Western European
500 to 850	EBCDIC International to PC-ASCII Western European
500 to 860	EBCDIC International to PC-ASCII Portuguese
500 to 861	EBCDIC International to PC-ASCII Icelandic
500 to 863	EBCDIC International to PC-ASCII Canadian French
500 to 865	EBCDIC International to PC-ASCII Scandinavian
813 to 869	ISO8859-7 Greek to PC-ASCII Greek
813 to 875	ISO8859-7 Greek to EBCDIC Greek
819 to 37	ISO8859-1 Western European to EBCDIC US English
819 to 284	ISO8859-1 Western European to EBCDIC Spanish
819 to 285	ISO8859-1 Western European to EBCDIC UK English
819 to 273	ISO8859-1 Western European to EBCDIC German
819 to 277	ISO8859-1 Western European to EBCDIC Danish/Norwegian
819 to 278	ISO8859-1 Western European to EBCDIC Finnish/Swedish
819 to 280	ISO8859-1 Western European to EBCDIC Italian
819 to 297	ISO8859-1 Western European to EBCDIC French
819 to 437	ISO8859-1 Western European to PC-ASCII US
819 to 500	ISO8859-1 Western European to EBCDIC International
819 to 850	ISO8859-1 Western European to PC-ASCII Western European
819 to 860	ISO8859-1 Western European to PC-ASCII Portuguese
819 to 861	ISO8859-1 Western European to PC-ASCII Icelandic
819 to 863	ISO8859-1 Western European to PC-ASCII Canadian French

Table 63. SBCS data conversion table (continued)

Code pages	Description
819 to 865	ISO8859-1 Western European to PC-ASCII Scandinavian
819 to 871	ISO8859-1 Western European to EBCDIC Icelandic
819 to 1051	ISO8859-1 Western European to ASCII roman8 for HP Western European
838 to 874	EBCDIC Thai SBCS to PC-ASCII Thai SBCS
850 to 37	PC-ASCII Western European to EBCDIC US English
850 to 284	PC-ASCII Western European to EBCDIC Spanish
850 to 285	PC-ASCII Western European to EBCDIC UK English
850 to 273	PC-ASCII Western European to EBCDIC German
850 to 277	PC-ASCII Western European to EBCDIC Danish/Norwegian
850 to 278	PC-ASCII Western European to EBCDIC Finnish/Swedish
850 to 280	PC-ASCII Western European to EBCDIC Italian
850 to 297	PC-ASCII Western European to EBCDIC French
850 to 437	PC-ASCII Western European to PC-ASCII US
850 to 500	PC-ASCII Western European to EBCDIC International
850 to 819	PC-ASCII Western European to ISO8859-1 Western European
850 to 860	PC-ASCII Western European to PC-ASCII Portuguese
850 to 861	PC-ASCII Western European to PC-ASCII Icelandic
850 to 863	PC-ASCII Western European to PC-ASCII Canadian French
850 to 865	PC-ASCII Western European to PC-ASCII Scandinavian
850 to 871	PC-ASCII Western European to EBCDIC Icelandic
850 to 1051	PC-ASCII Western European to ASCII roman8 for HP Western European
852 to 870	PC-ASCII Eastern European to EBCDIC Eastern Europe
852 to 912	PC-ASCII Eastern European to ISO8859-2 Eastern European
855 to 866	PC-ASCII Cyrillic to PC-ASCII Cyrillic # 2
855 to 880	PC-ASCII Cyrillic to EBCDIC Cyrillic
855 to 915	PC-ASCII Cyrillic to ISO8859-5 Cyrillic
855 to 1025	PC-ASCII Cyrillic to EBCDIC Cyrillic
856 to 424	PC-ASCII Hebrew to EBCDIC Hebrew
856 to 862	PC-ASCII Hebrew to PC-ASCII Hebrew
856 to 916	PC-ASCII Hebrew to ISO8859-8 Hebrew
857 to 920	PC-ASCII Turkish to ISO8859-9 Turkish
857 to 1026	PC-ASCII Turkish to EBCDIC Turkish
860 to 037	PC-ASCII Portuguese to EBCDIC US English
860 to 500	PC-ASCII Portuguese to EBCDIC International
860 to 819	PC-ASCII Portuguese to ISO8859-1 Western European
860 to 850	PC-ASCII Portuguese to PC-ASCII Western European
861 to 500	PC-ASCII Icelandic to EBCDIC International
861 to 819	PC-ASCII Icelandic to ISO8859-1 Western European
861 to 850	PC-ASCII Icelandic to PC-ASCII Western European
861 to 871	PC-ASCII Icelandic to EBCDIC Icelandic

Table 63. SBCS data conversion table (continued)

Code pages	Description
862 to 424	PC-ASCII Hebrew to EBCDIC Hebrew
862 to 856	PC-ASCII Hebrew to PC-ASCII Hebrew
862 to 916	PC-ASCII Hebrew to ISO8859-8 Hebrew
863 to 037	PC-ASCII Canadian French to EBCDIC US English
863 to 297	PC-ASCII Canadian French to EBCDIC French
863 to 500	PC-ASCII Canadian French to EBCDIC International
863 to 819	PC-ASCII Canadian French to ISO8859-1 Western European
863 to 850	PC-ASCII Canadian French to PC-ASCII Western European
864 to 420	PC-ASCII Arabic to EBCDIC Arabic
864 to 1046	PC-ASCII Arabic to PC-ASCII Arabic
864 to 1089	PC-ASCII Arabic to ISO8859-6 Arabic
865 to 277	PC-ASCII Scandinavian to EBCDIC Danish/Norwegian
865 to 278	PC-ASCII Scandinavian to EBCDIC Finnish/Swedish
865 to 437	PC-ASCII Scandinavian to PC-ASCII US
865 to 500	PC-ASCII Scandinavian to EBCDIC International
865 to 819	PC-ASCII Scandinavian to ISO8859-1 Western European
865 to 850	PC-ASCII Scandinavian to PC-ASCII Western European
866 to 855	PC-ASCII Cyrillic # 2 to PC-ASCII Cyrillic
866 to 880	PC-ASCII Cyrillic # 2 to EBCDIC Cyrillic
866 to 915	PC-ASCII Cyrillic # 2 to ISO8859-5 Cyrillic
866 to 1025	PC-ASCII Cyrillic # 2 to EBCDIC Cyrillic
869 to 813	PC-ASCII Greek to ISO8859-7 Greek
869 to 875	PC-ASCII Greek to EBCDIC Greek
870 to 852	EBCDIC Eastern Europe to PC-ASCII Eastern European
870 to 912	EBCDIC Eastern Europe to ISO8859-2 Eastern European
871 to 819	EBCDIC Icelandic to ISO8859-1 Western European
871 to 850	EBCDIC Icelandic to PC-ASCII Western European
871 to 861	EBCDIC Icelandic to PC-ASCII Icelandic
874 to 838	PC-ASCII Thai SBCS to EBCDIC Thai SBCS
875 to 813	EBCDIC Greek to ISO8859-7 Greek
875 to 869	EBCDIC Greek to PC-ASCII Greek
880 to 855	EBCDIC Cyrillic to PC-ASCII Cyrillic
880 to 866	EBCDIC Cyrillic to PC-ASCII Cyrillic # 2
880 to 915	EBCDIC Cyrillic to ISO8859-5 Cyrillic
912 to 852	ISO8859-2 Eastern European to PC-ASCII Eastern European
912 to 870	ISO8859-2 Eastern European to EBCDIC Eastern Europe
915 to 855	ISO8859-5 Cyrillic to PC-ASCII Cyrillic
915 to 866	ISO8859-5 Cyrillic to PC-ASCII Cyrillic # 2
915 to 880	ISO8859-5 Cyrillic to EBCDIC Cyrillic
915 to 1025	ISO8859-5 Cyrillic to EBCDIC Cyrillic

Table 63. SBCS data conversion table (continued)

Code pages	Description
916 to 424	ISO8859-8 Hebrew to EBCDIC Hebrew
916 to 856	ISO8859-8 Hebrew to PC-ASCII Hebrew
916 to 862	ISO8859-8 Hebrew to PC-ASCII Hebrew
920 to 857	ISO8859-9 Turkish to PC-ASCII Turkish
920 to 1026	ISO8859-9 Turkish to EBCDIC Turkish
1025 to 855	EBCDIC Cyrillic to PC-ASCII Cyrillic
1025 to 866	EBCDIC Cyrillic to PC-ASCII Cyrillic # 2
1025 to 915	EBCDIC Cyrillic to ISO8859-5 Cyrillic
1026 to 857	EBCDIC Turkish to PC-ASCII Turkish
1026 to 920	EBCDIC Turkish to ISO8859-9 Turkish
1046 to 420	PC-ASCII Arabic to EBCDIC Arabic
1046 to 864	PC-ASCII Arabic to PC-ASCII Arabic
1046 to 108	9PC-ASCII Arabic to ISO8859-6 Arabic
1051 to 437	ASCII roman8 for HP Western European to PC-ASCII US
1051 to 819	ASCII roman8 for HP Western European to ISO8859-1 Western European
1051 to 850	ASCII roman8 for HP Western European to PC-ASCII Western European
1089 to 420	ISO8859-6 Arabic to EBCDIC Arabic
1089 to 864	ISO8859-6 Arabic to PC-ASCII Arabic
1089 to 1046	ISO8859-6 Arabic to PC-ASCII Arabic

Table 64. DBSC and mixed data conversion table

Code pages	Description
37 to 897	SBCS EBCDIC US English to PC-ASCII Japan Data SBCS
37 to 904	SBCS EBCDIC US English to PC Traditional Chinese SBCS
37 to 1041	SBCS EBCDIC US English to PC Japanese - extended SBCS
37 to 1043	SBCS EBCDIC US English to PC Traditional Chinese - extended SBCS
37 to 1114	SBCS EBCDIC US English to PC Traditional Chinese - big 5 SBCS
290 to 897	SBCS EBCDIC Japanese Katakana SBCS to PC-ASCII Japan Data SBCS
290 to 1041	SBCS EBCDIC Japanese Katakana SBCS to PC Japanese - extended SBCS
300 to 301	DBCS EBCDIC Japanese DBCS to PC Japanese DBCS
300 to 941	DBCS EBCDIC Japanese DBCS to PC Japanese for open environment DBCS
301 to 300	DBCS PC Japanese DBCS to EBCDIC Japanese DBCS
301 to 941	DBCS PC Japanese DBCS to PC Japanese for open environment DBCS
833 to 891	SBCS EBCDIC Korean SBCS extended to PC-ASCII Korean SBCS
833 to 1040	SBCS EBCDIC Korean SBCS extended to PC-ASCII Korean - extended SBCS
833 to 1088	SBCS EBCDIC Korean SBCS extended to PC Korean SBCS - KS code
834 to 926	DBCS EBCDIC Korean DBCS to PC-ASCII Korean DBCS
834 to 951	DBCS EBCDIC Korean DBCS to PC Korean DBCS - KS code
835 to 927	DBCS EBCDIC Traditional Chinese DBCS to PC Traditional Chinese DBCS

Table 64. DBSC and mixed data conversion table (continued)

Code pages	Description
835 to 947	DBCS EBCDIC Traditional Chinese DBCS to PC Traditional Chinese DBCS
836 to 903	SBCS EBCDIC Simplified Chinese SBCS to PC Simplified Chinese SBCS
836 to 1042	SBCS EBCDIC Simplified Chinese SBCS to PC Simplified Chinese - extended SBCS
836 to 1115	SBCS EBCDIC Simplified Chinese SBCS to PC Simplified Chinese SBCS
837 to 928	DBCS EBCDIC Simplified Chinese DBCS to PC Simplified Chinese DBCS
837 to 1380	DBCS EBCDIC Simplified Chinese DBCS to PC Simplified Chinese DBCS
891 to 833	SBCS PC-ASCII Korean SBCS to EBCDIC Korean SBCS extended
891 to 1088	SBCS PC-ASCII Korean SBCS to PC Korean SBCS - KS code
897 to 37	SBCS PC-ASCII Japan Data SBCS to EBCDIC US English
897 to 290	SBCS PC-ASCII Japan Data SBCS to EBCDIC Japanese Katakana SBCS
897 to 1027	SBCS PC-ASCII Japan Data SBCS to EBCDIC Japanese Latin - extended SBCS
897 to 1041	SBCS PC-ASCII Japan Data SBCS to PC Japanese - extended SBCS
903 to 836	SBCS PC Simplified Chinese SBCS to EBCDIC Simplified Chinese SBCS
903 to 1042	SBCS PC Simplified Chinese SBCS to PC Simplified Chinese - extended SBCS
903 to 1115	SBCS PC Simplified Chinese SBCS to PC Simplified Chinese SBCS
904 to 37	SBCS PC Traditional Chinese SBCS to EBCDIC US English
904 to 1114	SBCS PC Traditional Chinese SBCS to PC Traditional Chinese - big 5 SBCS
926 to 834	DBCS PC-ASCII Korean DBCS to EBCDIC Korean DBCS
926 to 951	DBCS PC-ASCII Korean DBCS to PC Korean DBCS - KS code
927 to 835	DBCS PC Traditional Chinese DBCS to EBCDIC Traditional Chinese DBCS
927 to 947	DBCS PC Traditional Chinese DBCS to PC Traditional Chinese DBCS
928 to 837	DBCS PC Simplified Chinese DBCS to EBCDIC Simplified Chinese DBCS
928 to 1380	DBCS PC Simplified Chinese DBCS to PC Simplified Chinese DBCS
930 to 5050	EUC EBCDIC Japanese Katakana Kanji Mixed to euc Japanese Mixed 954
933 to 970	EUC EBCDIC Korean Mixed to euc Korean Mixed
935 to 1383	EUC EBCDIC Simplified Chinese Mixed to euc Simplified Chinese Mixed
937 to 964	EUC EBCDIC Traditional Chinese Mixed to euc Traditional Chinese Mixed
939 to 5050	EUC EBCDIC Japanese Latin Kanji Mixed to euc Japanese Mixed 954
941 to 300	DBCS PC Japanese for open environment DBCS to EBCDIC Japanese DBCS
941 to 301	DBCS PC Japanese for open environment DBCS to PC Japanese DBCS
942 to 5050	EUC PC Japanese PC Data Mixed - extended SBCS to euc Japanese Mixed 954
943 to 5050	EUC PC Japanese for open environment mixed to euc Japanese Mixed 954
947 to 835	DBCS PC Traditional Chinese DBCS to EBCDIC Traditional Chinese DBCS
947 to 927	DBCS PC Traditional Chinese DBCS to PC Traditional Chinese DBCS
948 to 964	EUC PC Traditional Chinese Mixed - extended SBCS to euc Traditional Chinese Mixed
949 to 970	EUC PC Korean Mixed - KS code to euc Korean Mixed

Table 64. DBSC and mixed data conversion table (continued)

Code pages	Description
950 to 964	EUC PC Traditional Chinese Mixed - big5 to euc Traditional Chinese Mixed
951 to 834	DBCS PC Korean DBCS - KS code to EBCDIC Korean DBCS
951 to 926	DBCS PC Korean DBCS - KS code to PC-ASCII Korean DBCS
964 to 937	EUC euc Traditional Chinese Mixed to EBCDIC Traditional Chinese Mixed
964 to 948	EUC euc Traditional Chinese Mixed to PC Traditional Chinese Mixed - extended SBCS
964 to 950	EUC euc Traditional Chinese Mixed to PC Traditional Chinese Mixed - big5
970 to 933	EUC euc Korean Mixed to EBCDIC Korean Mixed
970 to 949	EUC euc Korean Mixed to PC Korean Mixed - KS code
1027 to 897	SBCS EBCDIC Japanese Latin - extended SBCS to PC-ASCII Japan Data SBCS
1027 to 1041	SBCS EBCDIC Japanese Latin - extended SBCS to PC Japanese - extended SBCS
1040 to 833	SBCS PC-ASCII Korean - extended SBCS to EBCDIC Korean SBCS extended
1040 to 1088	SBCS PC-ASCII Korean - extended SBCS to PC Korean SBCS - KS code
1041 to 37	SBCS PC Japanese - extended SBCS to EBCDIC US English
1041 to 290	SBCS PC Japanese - extended SBCS to EBCDIC Japanese Katakana SBCS
1041 to 897	SBCS PC Japanese - extended SBCS to PC-ASCII Japan Data SBCS
1041 to 1027	SBCS PC Japanese - extended SBCS to EBCDIC Japanese Latin - extended SBCS
1042 to 836	SBCS PC Simplified Chinese - extended SBCS to EBCDIC Simplified Chinese SBCS
1042 to 903	SBCS PC Simplified Chinese - extended SBCS to PC Simplified Chinese SBCS
1043 to 37	SBCS PC Traditional Chinese - extended SBCS to EBCDIC US English
1043 to 1114	SBCS PC Traditional Chinese - extended SBCS to PC Traditional Chinese - big 5 SBCS
1088 to 833	SBCS PC Korean SBCS - KS code to EBCDIC Korean SBCS extended
1088 to 891	SBCS PC Korean SBCS - KS code to PC-ASCII Korean SBCS
1088 to 1040	SBCS PC Korean SBCS - KS code to PC-ASCII Korean - extended SBCS
1114 to 37	SBCS PC Traditional Chinese - big 5 SBCS to EBCDIC US English
1114 to 904	SBCS PC Traditional Chinese - big 5 SBCS to PC Traditional Chinese SBCS
1114 to 1043	SBCS PC Traditional Chinese - big 5 SBCS to PC Traditional Chinese - extended SBCS
1115 to 836	SBCS PC Simplified Chinese SBCS to EBCDIC Simplified Chinese SBCS
1115 to 903	SBCS PC Simplified Chinese SBCS to PC Simplified Chinese SBCS
1380 to 837	DBCS PC Simplified Chinese DBCS to EBCDIC Simplified Chinese DBCS
1380 to 928	DBCS PC Simplified Chinese DBCS to PC Simplified Chinese DBCS
1381 to 1383	EUC PC Simplified Chinese Mixed to euc Simplified Chinese Mixed
1383 to 935	EUC euc Simplified Chinese Mixed to EBCDIC Simplified Chinese Mixed

Table 64. DBSC and mixed data conversion table (continued)

Code pages	Description
1383 to 1381	EUC euc Simplified Chinese Mixed to PC Simplified Chinese Mixed
5050 to 930	EUC euc Japanese Mixed 954 to EBCDIC Japanese Katakana Kanji Mixed
5050 to 939	EUC euc Japanese Mixed 954 to EBCDIC Japanese Latin Kanji Mixed
5050 to 942	EUC euc Japanese Mixed 954 to PC Japanese PC Data Mixed - extended SBCS
5050 to 943	EUC euc Japanese Mixed 954 to PC Japanese for open environment mixed

Table 65. SBCS and DBCS data conversion table

Mixed	SBCS	DBCS
930	290	300
931	37	300
932	897	301
933	833	834
934	891	926
935	836	837
936	903	928
937	37	835
938	904	927
9391	027	300
9421	041	301
9431	041	941
9441	040	926
9461	042	928
9481	043	927
9491	088	951
9501	114	947
13811	115	1380
964	euc	
970	euc	
1383	euc	
5050	euc	

Bibliography

- *TXSeries for Multiplatforms Using IBM Communications Server for AIX with CICS*, SC34-6642
- *TXSeries for Multiplatforms Using IBM Communications Server for Windows Systems with CICS*, SC09-4470
- *TXSeries for Multiplatforms Using Microsoft SNA Server with CICS*, SC09-4471
- *TXSeries for Multiplatforms Using HP-UX SNAplus2 with CICS*, SC09-4586
- *TXSeries for Multiplatforms Using SNAP-IX for Solaris with CICS*, SC09-4472
- *CICS Family: Interproduct Communication*, SC33-0824
- *CICS Universal Client: Client Administration*, SC33-1792
- *CICS Administration Guide*
- *TXSeries for Multiplatforms Application Programming Reference*, SC34-6640
- *CICS Family: API Structure*, SC33-1007
- *CICS for OS/2 Intercommunication Guide*, SC33-0826
- *Communicating from CICS/400*, SC33-0828
- *TXSeries for Multiplatforms Administration Reference*, SC34-6641
- *IBM 3270 Information Display Programmer's Reference*,
- *TXSeries for Multiplatforms Messages and Codes*, SC34-6639
- *TXSeries for Multiplatforms SFS Server and PPC Gateway Server: Advanced Administration*, SC34-6627
- *TXSeries for Multiplatforms Concepts and Planning*, SC34-6631
- *TXSeries for Multiplatforms Problem Determination Guide*, SC34-6636
- *TXSeries for Multiplatforms Application Programming Guide*, SC34-6634

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the document. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

IBM Corporation
ATTN: Software Licensing
11 Stanwix Street
Pittsburgh, PA 15222
U.S.A.

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

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks and service marks

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

Advanced Peer-to-Peer Networking®	AIX
AS/400®	CICS
CICS/400	CICS/6000
CICS/ESA	CICS/MVS
CICS/VSE	CICSPlex®

C-ISAM™	Database 2™
DB2	DB2 Universal Database™
GDDM®	IBM
IBM Registry™	IMS
Informix®	Language Environment®
MVS	MVS/ESA
OS/390®	OS/2
OS/400®	RACF
RETAIN®	RISC System/6000®
RS/6000	SOM®
Systems Application Architecture®	System/390®
TXSeries	TCS
VisualAge®	VSE/ESA™
VTAM	WebSphere®
z/OS	

Domino®, Lotus®, and LotusScript are trademarks or registered trademarks of Lotus Development Corporation in the United States, other countries, or both.

ActiveX, Microsoft, Visual Basic, Visual C++, Visual J++, Visual Studio, Windows, Windows NT®, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Acucorp and ACUCOBOL-GT are registered trademarks of Acucorp, Inc. in the United States, other countries, or both.

Pentium® is a trademark of Intel Corporation in the United States, other countries, or both.

This software contains RSA encryption code.

Other company, product, and service names may be trademarks or service marks of others.

Index

Numerics

15a00001/8890000 Program Error Purging 223
15a00002/15a00101 Allocate Busy 223
15a00002/15a00102 Allocate Failure 224
15a00003/15a00109 Allocate Timed Out 224
15a00004/15a00103 Invalid Convid 224
15a00004/15a00104 Invalid Mode/Partner LU alias 224
15a00004/15a00105 Invalid Parameter 225
15a00004/15a0010d Synclevel Not Supported Locally 224
15a00004/15a00110 Exchange log name failed 225
15a00005/15a00108 State Error 225
15a00005/15a00111 Backout required 225
15a00006/15a00106 Communications Protocol Specific Error 226
15a00006/15a0010c Local SNA not initialized 226
15a00007/10086021 Transaction Unknown on Remote System 226
15a00007/10086031 PIP Data Not Supported by the Remote System 226
15a00007/10086032 PIP Data Incorrectly Specified 226
15a00007/10086034 Conversation Type Mismatch 226
15a00007/10086041 Sync Not Supported by Remote Program or System 226
15a00007/80f6051 Security Violation 226
15a00007/84b6031 Remote Transaction Temporarily Not Available 227
15a00007/84c0000 Remote Transaction Not Available 227
15a00007/8640000 Conversation Abnormal Termination 227
15a00007/8640001 Conversation Abnormal Termination 227
15a00007/8640002 Conversation Abnormal Termination 227
15a00007/a0000100 Conversation Failure 226
15a00008/15a00107 Conversation Terminated Normally 227
15a00009/8240000 Backed Out 228
15a0000a/15a0010a Unsupported 223

A

abend codes
- A30A 332
- ASP3 314
- ATCV 331
- ATNI 261
- AXFX abend on CICS for MVS/ESA 219

abnormal termination
- APPC mapped conversations 299

ActivateOnStartup attribute (LD) 27
advanced program-to-program communications (APPC) 3
- DTP applications 4

AIX SMIT
- changing attributes PPC Gateway server 199
- configuring CD entry 39, 52, 86, 106
- configuring GD entry 103
- configuring LD entry 31, 82
- configuring RD entry 83, 104
- creating PPC Gateway server 195
- deleting PPC Gateway server 200
- PPC Gateway server system management 195
- starting PPC Gateway server 197
- stopping PPC Gateway server 198

AIX SNA Server/6000 9
ALLOCATE 304
- APPC mapped conversations 286, 332

AllocateTimeout attribute (CD) 34, 47, 96, 99, 106
allocating a session
- APPC mapped conversations 287
- using ATI 287

already verified 133
alternate facility 281
alternate facility, defined 279
APPC mapped conversations
- abnormal termination 295, 299
- ALLOCATE command 286, 332
- ASSIGN command 289
- attaching partner transactions 288
- CONNECT PROCESS command 288
- CONVERSE command 295, 301
- CONVID (acquiring) 286
- ending one 298
- EXTRACT PROCESS command 288, 289
- FREE command 298, 299
- front-end transaction 286
- ISSUE ABEND command 295, 299
- ISSUE ERROR command 295
- RECEIVE command 301
- SEND command 291
- starting 286
- state table, synchronization level 0 333
- state table, synchronization level 1 335
- state table, synchronization level 2 338
- synchronization level 0, state table 333
- synchronization level 1, state table 335
- synchronization level 2, state table 338

application design 282, 283
application programming
- APPC mapped conversations 286
- asynchronous processing 277
- distributed program link (DPL) 247
- distributed transaction processing (DTP) 279
- function shipping 259
- transaction routing 264

APPLID 25, 68
APPLID passed with START command 275
ASCII to EBCDIC data conversion 19, 147
ASP3 abend 314
ASSIGN command
- APPC mapped conversations 289
- APPLID option usage 275
- in application owning system 265

asynchronous processing 273, 274, 275, 276, 277
- application programming 277
- data conversion 156
- deferred sending of START request 275
- defining remote transactions 120
- definition of 3
- description of 20
- example application 276
- including start request in an LUW 276
- initiating asynchronous processing 273
- passing information with the START command 274
- security considerations 273
- single transaction satisfying multiple start requests 276
- START/RETRIEVE interface 273
- started transaction 276

attaching partner transactions
- APPC mapped conversations 288

attaching secure sessions 125
audit messages 232
authentication 126
authorizing user access to CICS
- signing on from a remote system using CESN 142

auto start PPC Gateway server 191
autoinstall of CD entries 55
automatic transaction initiation (ATI) 266
- APPC mapped conversations 287
- defining resources 116
- EIBTRMID field 111
- EIBTRMID field of 111
- FacilityType attribute 111
- in asynchronous processing 274
- principal facility 111
- resource definitions 116

automatic transaction initiation (ATI) *(continued)*
restriction with shipped terminal definitions 117
transient data trigger level 111
TriggeredTransId attribute 111
autostart PPC Gateway server 183, 186, 197
availability of remote regions 304
AXFX abend on CICS for MVS/ESA 219

B

back-end DTP transaction availability 109
back-end transaction 285
backing out changes 314, 316
performance effect 282
to recoverable resources 282, 308
backout 16, 308, 314, 316
definition 279
effect on performance 282
of recoverable resources 282
state after 344
backup of remote data 22
balancing work load 116, 267
basic mapping support (BMS)
with transaction routing 264
BDAM files, accessed by DPL 248
BDAM, restriction 257
big-endian machines 147
BIND
password 125
request 125
security, implementing 125
bind-time security
description of 14
planning for 14
byte ordering 149

C

CANCEL command 274
CCIN 55
CECI 214
cell directory services (CDS)
name for CICS region 25
name for PPC Gateway server 176
CESN 142
CGWY 177, 218
checking security 123
checking the conversation state of a transaction 294
choosing intercommunication facilities 20
CICS family TCP/IP 5
authenticating systems 124
configuring CD entry 34
configuring connections 26
configuring LD entry 27
functions supported 6
CICS on Open Systems and CICS for Windows
availability of 304
CICS OS/2 panels 44
CICS OS/2 to TXSeries for Multiplatforms data conversion 19
CICS region principal 25, 68
CICS_PPCGWY_SERVER 186, 197, 200
CICS_PPCGWY_VG 185
CICS_SNA_DEFAULT_MODE 110
CICS_SNA_THREAD_POOL_SIZE 111
CICS-supplied transactions
CGWY 177
cicsadd 210
cicscp command set 182
cicscp create ppcgwy_server 182
cicscp destroy ppcgwy_server 184
cicscp start ppcgwy_server 183
cicscp stop ppcgwy_server 184
cicscvt 159
cicsdelete 210
cicsget 188
cicsppcgwy 186, 188, 230
cicsppcgwycreate 184, 188
cicsppcgwydestroy 188
cicsppcgwylock 189
cicsppcgwyshut 187
CICSREGION 31, 39, 52, 82, 103
cicsupdate 188, 210
client/server model 281
client/server, defined 279
code page 147, 153, 157, 159, 321
cold start PPC Gateway server 183, 186, 191, 197
commands
ALLOCATE 304
APPC mapped conversations 304
CONNECT PROCESS 304
data integrity 296
FREE 304
INQUIRE CONNECTION 304
RETURN 304
SEND 304
COMMAREA, used with DPL 248
commit 16
commits, defined 279
committing changes
to recoverable resources 307
communication methods 5
CICS family TCP/IP 5
local SNA support 8, 25, 67, 70
PPC Gateway server 9, 25, 67, 89, 175
TCP/IP 5, 25, 47
communication pairs, defined 279
Communications Definitions (CD) 210
autoinstall 55
configuring for CICS family TCP/IP 34
configuring for CICS family TCP/IP (SMIT) 39
configuring for CICS PPC TCP/IP 47
configuring for CICS PPC TCP/IP (SMIT) 52
configuring for local SNA 75, 79, 86
configuring for local SNA (SMIT) 86
configuring for PPC Gateway server SNA 96, 99, 106
configuring for PPC Gateway server SNA (SMIT) 106
example 46, 53, 75, 79, 86, 96, 99, 106
Communications Server 8
Communications Server for AIX
tracing 230
configuration problems 144
configuring the PPC Gateway server 177
confirm 16
confirm, defined 279
confreceive state, defined 279
CONNECT command 304
CONNECT PROCESS command
APPC mapped conversations 288
PIPLENGTH option 288
PIPLIST option 288
connection security 126, 127
connections
local SNA support 70
SNA and a PPC Gateway server 89
TCP/IP 47
ConnectionType attribute (CD) 75, 79, 86, 96, 99, 106
conversation identifiers 344
conversation state 280
conversation state, defined 279
conversation-level security 133
conversation, defined 279, 280
CONVERSE command
APPC mapped conversations 295, 301
CICS conversations 295
conversion of data between systems 19
conversion templates 324
CONVID (acquiring) 286
CONVID option
APPC mapped conversations 289, 291
WAIT command 291
CONVIDs 344
CRTE 142, 214

D

data conversion 19
ASCII to EBCDIC 19
asynchronous processing 152
avoiding data conversion 158
between systems 19
byte ordering 149
byte swapping 150
CICS OS/2 TXSeries for Multiplatforms 19
code pages 147, 153
conversion table 147
conversion templates 324
DFHTRUC 165
DFHUCNV 160
distributed program link 152
distributed transaction processing 169
EBCDIC to ASCII 19
function shipping 156
key templates 324
numeric data 149
RemoteCodePageTR attribute 164
SBCS, DBCS, MBCS 148
shortcodes 157
transaction routing 164

data conversion *(continued)*
  TXSeries for Multiplatforms to CICS OS/2 19
data conversion exits
  DFHTRUC 164
  DFHUNCV 160
  function shipping 160
  transaction routing 164
data integrity 5, 282, 296
  definition 279
  function shipping 258
  safegaurding 284
DB2, restriction 257
DBCS
  double byte character set 148
DefaultSNAModeName attribute (CD) 34, 47, 75, 79, 86, 96, 99, 106
DefaultUserId attribute 137
DefaultUserId attribute (RD) 265
deferred sending, START NOCHECK 275
deferred transmission 299
  APPC mapped conversations 291
design of intercommunication setup 5
designing for recovery 284
DFHCCINX 55, 58, 62
DFHCNV macro
  creating a template from 159
  TYPE=ENTRY 321, 325
  TYPE=FIELD 321, 328
  TYPE=FINAL 321, 329
  TYPE=INITIAL 321, 324
  TYPE=KEY 321, 327
  TYPE=SELECT 321, 327
DFHTRUC 165
DFHUCNV 160
distribued program link (DPL)
  dynamic distributed program link user exit 249
  user exit 249
distributed processes 281
  defininition 279
  designing 283
distributed program link (DPL) 247
  BDAM files, accessed by 248
  COMMAREA 248
  data conversion 156
  definition of 3
  description of 20
  IMS databases, accessed by 248
  overview of 247
  performance optimization 248
  serial connections 248
  SQL databases, accessed by 248
  subset 249
  ways of using DPL 252
distributed transaction processing (DTP)
  definition of 3
  description of 21
  migration of applications 343
distributed unit of work 16, 282
distributed unit of work, defined 279
DL/I databases
  accessed by DPL 248
  restriction 257
dynamic distributed program link user exit 249, 252
dynamic transaction routing 116, 266, 267

E

EBCDIC to ASCII data conversion 19, 147
EIB fields 309
  EIBCOMPL, APPC mapped conversations 301
  EIBCONF, APPC mapped conversations 301
  EIBEOC, APPC mapped conversations 301
  EIBERR 291, 308
  EIBERR, APPC mapped conversations 295, 300
  EIBERR, CICS conversations 293
  EIBERRCD 291
  EIBERRCD, APPC mapped conversations 295, 300
  EIBERRCD, CICS conversations 293
  EIBFREE 291, 309
  EIBFREE, APPC mapped conversations 295, 300
  EIBNODAT, APPC mapped conversations 301
  EIBNODAT, CICS conversations 293
  EIBRCODE, APPC mapped conversations 300
  EIBRECV, APPC mapped conversations 301
  EIBRLDBK 308, 316, 317
  EIBRSRCE 287
  EIBSIG, APPC mapped conversations 295, 300
  EIBSYNC 307, 317
  EIBSYNC, APPC mapped conversations 300
  EIBSYNRB 308, 317
  EIBSYNRB, APPC mapped conversations 300, 301
EIBCOMPL flag
  APPC mapped conversations 301
EIBCONF flag
  APPC mapped conversations 301
EIBEOC flag
  APPC mapped conversations 301
EIBERR flag 300, 308
  APPC mapped conversations 291, 295
  CICS conversations 293
EIBERRCD field
  APPC mapped conversations 291, 295, 300
  CICS conversations 293
EIBFREE flag 309
  APPC mapped conversations 291, 295, 300
EIBNODAT flag
  APPC mapped conversations 301
  CICS conversations 293
EIBRCODE field
  APPC mapped conversations 300
EIBRECV flag
  APPC mapped conversations 301
EIBRLDBK flag 308, 316, 317
EIBRSRCE field
  APPC mapped conversations 287
EIBSIG flag
  APPC mapped conversations 295, 300
EIBSYNC flag 307, 317
  APPC mapped conversations 300
EIBSYNRB flag 308, 317
  APPC mapped conversations 300, 301
EIBTRMID field 111
ending a conversation
  APPC mapped session 298
ensuring data integrity 5
environment va riables
  CICS_SNA_DEFAULT_MODE 110
environment variables
  CICS_PPCGWY_SERVER 186, 197, 200
  CICS_PPCGWY_VG 185
  CICS_SNA_THREAD_POOL_SIZE 111
  CICSREGION 31, 82, 103
ERZ045006W message occurs 145
exceptional conditions
  function shipping 260
exchange log names 180, 204, 205
  problems 219
EXEC CICS ALLOCATE 286
EXEC CICS commands
  summary 304
EXEC CICS CONNECT PROCESS 288, 289
EXEC CICS CONVERSE 295
EXEC CICS ISSUE ABEND 296, 297, 299
EXEC CICS ISSUE CONFIRMATION 296, 297
EXEC CICS ISSUE ERROR 295, 296, 297
EXEC CICS ISSUE SIGNAL 295
EXEC CICS RECEIVE 292, 293
EXEC CICS RETRIEVE 277
EXEC CICS RETURN 16
EXEC CICS SEND 291
EXEC CICS SEND CONFIRM 296, 297
EXEC CICS SEND CONFIRM INVITE 297
EXEC CICS SEND CONFIRM LAST 297
EXEC CICS SEND INVITE 292, 295
EXEC CICS SEND INVITE WAIT 292
EXEC CICS SEND LAST 297, 299
EXEC CICS SEND LAST CONFIRM 299
EXEC CICS SEND LAST WAIT 299
EXEC CICS START 277
EXEC CICS SYNCPOINT 16, 298, 299
EXEC CICS SYNCPOINT ROLLBACK 16, 298
EXEC CICS WAIT CONVID 292
execution-time choice 257
explicit naming of remote system 257
EXTRACT ATTRIBUTES 344
EXTRACT ATTRIBUTES command 281, 285
EXTRACT PROCESS command
  APPC mapped conversations 288, 289

F

FacilityId attribute 287
FacilityType attribute 111, 287
failures
  back-end transaction 291
  intersystem session 279, 284
  notification of 291
file control
  function shipping 258, 259
files, securing 126, 127
flowing user IDs outbound 130
FREE command
  APPC mapped conversations 298, 299
FREE command use of 304
front-end transaction 285
  APPC mapped conversations 286
function shipping
  application programming 259
  BDAM, restriction 257
  data conversion 156
  DB2, restriction 257
  defining remote resources 113
    files 113
    temporary storage queues 115
    transient data queues 114
  definition of 3
  description of 20
  DL/I, restriction 257
  exceptional conditions 260
  EXEC CICS START 141, 142
  execution-time choice 257
  explicit naming of remote system 257
  file control 258, 259
  IMS, restriction 257
  local and remote names 258
  migration considerations 345
  mirror transaction 142
  mirror transaction abnormal termination 261
  overview 257
  remote system, naming 257
  security 141
  START command 141, 142
  synchronization 258
  SYNCPOINT, use in 259
  temporary storage 260
  transient data 258, 260
  transparent to application 257
Function shipping
  execution-time choice 273
  explicit naming of remote system 273
  Timeouts 261
function shipping data conversion
  DFHUCNV 160
  non-standard conversion 160
  program parameters 161
  standard conversion 159
  the supplied program 160

G

gateway
  planning for 5
Gateway Definitions (GD) 210
  configuring for PPC Gateway server SNA 95, 98, 103
  configuring for PPC Gateway server SNA (SMIT) 103
  example 95, 98, 103
GatewayCDSName attribute (GD) 95, 98, 103
GatewayLUName attribute 68
GatewayLUName attribute (GD) 95, 98, 103
GatewayName attribute (CD) 34, 47, 96, 99, 106
GatewayPrincipal attribute 126
GatewayPrincipal attribute (GD) 95, 98, 103
GDS ISSUE PREPARE command 308

H

help 213

I

IBM mainframe-based CICS
  availability of 304
IBM TXSeries Administration Tool
  changing attributes PPC Gateway server 193
  creating PPC Gateway server 190
  destroying PPC Gateway server 194
  PPC Gateway server system management 190
  starting PPC Gateway server 191
  stopping PPC Gateway server 193
IMS databases, accessed by DPL 248
IMS, restriction 257
inbound, defined 279
indirect links for transaction routing 266
indoubt condition 109
INQUIRE CONNECTION command 304
INQUIRE FILE command 258
integrity of data 282
intercommunication design 5
intercommunication security 123
intersystem requests, security of 123
intersystem security
  configuring for 136
  implementing 136
INVITE option
  SEND command (APPC mapped) 292
ISCINVREQ 260
IsShippable attribute 117
ISSUE ABEND command
  APPC mapped conversations 295
ISSUE ERROR command
  APPC mapped conversations 295
ISSUE PREPARE command 308

L

LINK command 247
link keys, definition of 127
link security 126
link security *(continued)*
  configuring for 136
  description of 14
  implementing 136
  planning for 14
LinkUserId attribute 136
LinkUserId attribute (CD) 34, 47, 75, 79, 86, 96, 99, 106, 265
Listener Definitions (LD) 210
  configuring for CICS family TCP/IP 27
  configuring for CICS family TCP/IP (SMIT) 31
  configuring for local SNA 75, 77, 82
  configuring for local SNA (SMIT) 82
  example 32, 45, 46, 75, 77, 82
ListenerName attribute (CD) 34
little-endian machines 147
local and remote names of resources 258
local Logical Unit (LU) name 25, 68
local queuing of START commands 275
local SNA support 5, 8, 70
Local TPN Profile 109
local transaction profile used 109
LocalLUName attribute 68
LocalLUName attribute (RD) 75, 78, 83
LocalNetworkname attribute (RD) 75, 78, 83
LocalSysId attribute (RD) 25, 68, 75, 78, 83, 96, 98, 104
logical unit (LU) name 68
Logical Unit (LU) name 25
logical unit of work (LUW) 16, 208, 279, 282
LU 6.2 3
LU-LU verification 125
LUTYPE6.2 3
LUW (logical unit of work) 208, 282

M

mapped conversations
  migration of 343
master/slave, defined 279
MBCS
  multi byte character set 148
migration
  distributed transaction processing (DTP) 343
  DTP 343
  function shipping considerations 345
  transaction identifiers 345
  transactions to transaction routing environment 264
mirror transaction
  abnormal termination 142
  security 141
mirror transaction abnormal termination 261
model
  client/server 281
  peer-to-peer 281
modename 110
  default modename 110
  DefaultSNAModeName (CD) 110

N

name of region 25, 47, 68
network failures and data integrity 5
network monitoring 22
network name
  LocalNetworkName 69
  RemoteNetworkName 35, 50
network protocols 3, 5
NOAUTH condition 142
NOCHECK parameter 277
  improving performance 275
  local queuing 275
  START command 275

O

operational issues 22
OPERKEYS
  option on ASSIGN command 265
outbound users, flowing 130
outbound, defined 279
OutboundUserIds attribute 130
OutboundUserIds attribute (CD) 34, 47, 75, 79, 86, 96, 99, 106, 265
overview of security 123

P

partner transaction, connecting 288
password verification 133
peer-to-peer model 281
peer-to-peer, defined 279
performance issues 19
performance optimization, DPL 248
persistent verification 133
PIPLENGTH option
  CONNECT PROCESS command 288
PIPLIST option
  CONNECT PROCESS command 288
planning 5
  ASCII to EBCDIC data conversion 19
  choosing intercommunication facilities 20
  CICS on Open Systems to CICS OS/2 data conversion 19
  CICS OS/2 to TXSeries for Multiplatforms data conversion 19
  conversion of data between systems 19
  data conversion between systems 19
  data integrity 5
  EBCDIC to ASCII data conversion 19
  gateway 5
  intercommunication facilities 20
  network protocols 5
  performance issues 19
  security of resources 14
  SNA 5
  synchronization 5
  Systems Network Architecture 5
  TCP/IP 5
  Transmission Control Protocol/Internet Protocol 5
PPC Executive 5
PPC Gateway server 5, 9, 89

PPC Gateway server *(continued)*
  altering CICS intercommunication definitions 210
  autostart 183, 186, 191, 197
  canceling pending resynchronizations (ppcadmin) 206
  CGWY 177
  changing attributes 188
  changing attributes (AIX SMIT) 199
  changing attributes (IBM TXSeries Administration Tool) 193
  cicscp command set 182
  cicscp create ppcgwy_server 182
  cicscp destroy ppcgwy_server 184
  cicscp start ppcgwy_server 183
  cicscp stop ppcgwy_server 184
  cicsppcgwy 186, 188
  cicsppcgwycreate 184, 188
  cicsppcgwydestroy 188
  cicsppcgwylock 176
  cicsppcgwyshut 187
  cold start 183, 186, 191, 197
  configuring 177
  creating 184, 188
  creating (AIX SMIT) 195
  creating (cicscp) 182
  creating (IBM TXSeries Administration Tool) 190
  deleting 188
  deleting (AIX SMIT) 200
  deleting (cicscp) 184
  deleting (IBM TXSeries Administration Tool) 194
  Gateway Server Definitions (GSD) 177
  listing attributes 188
  listing running 189
  lock 189
  messages 228
  overview 175
  planning for 5
  ppcadmin administration command set 200
  requesting XLN process (ppcadmin) 205
  running (view) 189
  starting 186, 188, 230
  starting (AIX SMIT) 197
  starting (cicscp) 183
  starting (IBM TXSeries Administration Tool) 191
  stopping 187
  stopping (AIX SMIT) 198
  stopping (cicscp) 184
  stopping (IBM TXSeries Administration Tool) 193
  trace 228
  tracing 230
  unlock 189
  viewing attributes 188
  viewing CICS configuration (ppcadmin) 203
  viewing intersystem requests (ppcadmin) 207
  viewing LUWs (ppcadmin) 208
  viewing pending resynchronizations (ppcadmin) 206

PPC Gateway server *(continued)*
  viewing XLN process status (ppcadmin) 204
ppcadmin 180, 200
ppcadmin cancel resync 206
ppcadmin force xln 205
ppcadmin list convs 207
ppcadmin list luentries 203
ppcadmin list luws 208
ppcadmin list xlns 204
ppcadmin query conv 207
ppcadmin query luentry 203
ppcadmin query luw 208
ppcadmin query resync 206
ppcgwy 189
preparing a partner for sync point 308
principal facility 111, 281
principal facility, defined 279
PRINSYSID
  option on ASSIGN command 265
problem determination support 22
problems 144
program development
  APPC mapped conversations 286
programming
  APPC mapped conversations 286
programs, securing 126, 127
PROTECT parameter
  START command 276
Protocol attribute (LD) 27, 75, 77, 82
protocols 5
  SNA 25, 67, 70, 89
  TCP/IP 25, 47
pseudo-conversational transactions
  with transaction routing 264

R

RBA files, note about 258
RDO 210
RECEIVE command
  APPC mapped conversations 301
receive state, defined 279
record lengths for remote files 114
recoverable resources 282
  canceling changes to 282, 308
  committing changes to 282, 307
region
  name 68
Region Definitions (RD)
  configuring for local SNA 75, 78, 83
  configuring for local SNA (SMIT) 83
  configuring for PPC Gateway server SNA 96, 98, 104
  configuring for PPC Gateway server SNA (SMIT) 104
  example 75, 78, 83, 96, 98, 104
region name 25, 47
remote and local names of resources 258
remote files
  defining 113
  file names 114
  record lengths 114
Remote Procedure Call (RPC)
  authenticated 126
  PPC Gateway server to region communication 177, 179

Remote Procedure Call (RPC) *(continued)*
region to PPC Gateway server communication 177, 178
remote regions, availability 304
remote system security 126
remote system, naming 257
remote temporary storage queues
defining 115
remote terminals
terminal identifiers 117
terminal names 118
remote transactions
deadlock time out 120
defining for asynchronous processing 120
defining for transaction routing 119
security 120
transaction names 120
Transaction Work Area (TWA) 120
remote transient data queues
defining 114
RemoteLUName attribute 117
RemoteLUName attribute (CD) 34, 47, 75, 79, 86, 96, 99, 106
RemoteName attribute 118
RemoteNetworkName attribute (CD) 34, 47, 75, 79, 86, 96, 99, 106
RemoteSysEncrypt attribute (CD) 34, 47
RemoteSysId attribute 118, 257
RemoteSysSecurity attribute 136
RemoteSysSecurity attribute (CD) 34, 47, 75, 79, 86, 96, 99, 106, 265
RemoteTCPAddress attribute (CD) 34
RemoteTCPPort attribute (CD) 34
rerouting transactions 116, 267
resource definition 210
resource definition online
shippable terminal definitions 117
resource definitions
ATI 116
automatic transaction initiation (ATI) 116
transaction routing 116
resource security
configuring for 136
implementing 136
resource security level 126, 127
resources used 22
resynchronization 180, 206
RETRIEVE command 277
RETURN 16
RETURN command use of 304
rollback 16, 282
rollback, defined 279
RSLKey attribute 136
RSLKeyList 127
RSLKeyList attribute 136
RSLKeyMask 126
RSLKeyMask attribute 136
RSLKeyMask attribute (CD) 34, 47, 75, 79, 86, 96, 99, 106
RuntimeProtection attribute 126

S

SBCS
single byte character set 148
security
ASSIGN command 265
asynchronous processing 273
attaching sessions 125
BIND password 125
BIND request 125
bind-time, implementing 125
checking 123
checklist 143
configuration problems 144
connection 126
files, securing 126, 127
flowing user IDs outbound 130
function shipping 141
function shipping, considerations for the user 141
intercommunication 123
intersystem requests 123
intersystem, configuring for 136
link 126
link keys, definition of 127
link keys, retrieving 265
LU-LU verification 125
mirror transaction 141
outbound users, flowing 130
OutboundUserIds attribute 130
overview of 123
planning 14
problems 144
programs, securing 126, 127
remote sign on 142
remote system 126
resource security level 126, 127
RSLKeyList 127
session 125
signing on, remote 142
SNA 123
SNA bind-time 125
symptoms 144
temporary storage, securing 126, 127
terminals, securing 126, 127
transaction security level 126, 127
transient data, securing 126, 127
TSLKeyList attribute 127
TSLKeyMask attribute 126
user 127
userid, retrieving 265
users, securing 126, 127
SEND command 304
APPC mapped conversations 291, 292
send state, defined 279
serial connections
distributed program link (DPL) 248
function shipping 257
server and IBM CICS Client 4
session security 125
sessions
allocating under ATI 287
SET FILE command 258
shippable terminal definitions 117
shippable terminal, transaction routing 263
shortcodes
data conversion 157
SMIT
changing attributes PPC Gateway server 199
creating PPC Gateway server 195
deleting PPC Gateway server 200
PPC Gateway server system management 195
starting PPC Gateway server 197
stopping PPC Gateway server 198
SMIT panels
configuring CD entries 39, 52, 86, 106
configuring GD entries 103
configuring LD entries 31, 82
configuring RD entries 83, 104
SNA 5, 8, 9, 282
attaching sessions 125
BIND password 125
BIND request 14, 125
bind-time security 125
bind-time security, implementing 125
local Logical Unit (LU) name 25, 68
Local TPN Profile 109
logical unit (LU) name 25, 68
LU-LU verification 125
planning for 5
security 123
session security 125
synchronization level 16
TCP/IP, differences between for DTP 343
TPN Profile 109
transaction availability 109
use of 3
SNAConnectName attribute (CD) 34, 47, 75, 79, 86, 96, 99, 106
SNAModeName attribute (TD) 75, 79, 86, 96, 99, 106
SQL databases, accessed by DPL 248
START command 142, 277
START/RETRIEVE 273, 275, 276
deferred sending 275
improving performance 275
local queuing 275
passing information with the START command 274
RETRIEVE command 276
retrieving data sent with START command 276
TERMID parameter 277
terminal acquisition 277
state after backout 344
state of a conversation 280
STATE option 281, 285
state tables
synchronization level 0 333
synchronization level 1 335
synchronization level 2 338
using 331
state transition, defined 279
state variable 279, 281
structured distributed transactions 283
subset of calls for DPL 249
supplied transactions
CGWY 177
surrogate terminal 264
symptoms 144

sync point 16, 282
- preparing a partner for 308

sync point, defined 279
synchronization 5, 282
- function shipping 258
- levels of 282

synchronization level 5, 16, 282
- definitions of 5
- described 5
- SNA support for 8, 9, 70, 89
- TCP/IP support for 5

synchronization level 2
- problems 219

synchronization point, defined 279
synchronization, defined 279
synchronize, defined 279
SYNCPOINT 16
SYNCPOINT command 307
SYNCPOINT ROLLBACK 16, 344
SYNCPOINT ROLLBACK command 308
syncreceive state, defined 279
SYSID 25, 47, 68
SYSIDERR 277
SYSIDERR condition, on START command 276
system availability 22
system design 5
system failures and data integrity 5

T

TCP/IP 5
- configuring connections 47
- naming your region 25
- planning for 5
- SNA, differences between for DTP 343
- transaction availability 109
- use of 3

TCPAddress attribute (LD) 27
TCPService attribute (LD) 27
temporary storage
- function shipping 260

temporary storage, securing 126, 127
TERMID parameter, START command 277
terminal definition
- transaction routing 263

terminal not found user exit 266
terminals, securing 126, 127
termination, abnormal
- APPC mapped conversations 300

testing the conversation state 303
TPN profile 109
TPNSNAProfile attribute 109
tracing
- PPC Gateway server 230

transaction
- availability 109
- back-end 285
- back-end DTP 109
- front-end 285
- performance considerations 109
- profiles used 109

Transaction Definitions (TD)
- transaction routing 263

transaction identifiers, migration of 345
transaction processing
- definition of 3

transaction routing
- alias for terminals 118
- application programming 264
- ATI 266
- basic mapping support 264
- data conversion 164
- defining remote resources 116
- defining remote resources, terminals 118
- defining remote resources, transactions 119
- defining resources 116
- description of 20
- dynamic 116, 267
- indirect links 266
- overview of 263
- pseudo-conversational transactions 264
- resource definitions 116
- surrogate terminal 264
- terminal, initiating 263
- transaction, initiated 263
- use of ASSIGN command in application owning system 265

transaction security level 126, 127
transient data
- function shipping 258, 260
- principal facility 111
- remote 111
- securing 126, 127
- trigger level 111

TriggeredTransId attribute 111
TSLKey attribute 136
TSLKeyList attribute 127, 136
TSLKeyMask attribute 126, 136
TSLKeyMask attribute (CD) 34, 47, 75, 79, 86, 96, 99, 106
two-phase commit 109
TXSeries for Multiplatforms to CICS OS/2 data conversion 19

U

unit of work (UOW) 16
user access to CICS
- security 142

user exits
- DFHTRUC 164
- DFHUCNV 160
- dynamic distributed program link 252
- dynamic transaction routing 116, 266, 267
- terminal not found 266

user securing 127
user security 127
- configuring for 136
- description of 14
- ESM, description of 14
- External Security Manager, description of 14
- implementing 136
- planning for 14

user, securing 126
USERID
- option on ASSIGN command 265

V

verifying passwords 133
VSAM RBA files, note about 258

W

WAIT 276
- APPC mapped conversations 288, 291

WAIT option (APPC mapped)
- SEND command 291

work-load balancing 116, 267

X

XLN 180, 204, 205
- problems 219

Xno 219

SC34-6644-00

Spine information:

IBM TXSeries for Multiplatforms CICS Intercommunication Guide Version 6.0 SC34-6644-00