

Transaction Processing Facility



# ACF/SNA Data Communications Reference

*Version 4 Release 1*



Transaction Processing Facility



# ACF/SNA Data Communications Reference

*Version 4 Release 1*

**Note!**

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

**Seventh Edition (June 2002)**

This is a major revision of, and obsoletes, SH31-0168-05 and all associated technical newsletters.

This edition applies to Version 4 Release 1 Modification Level 0 of IBM Transaction Processing Facility, program number 5748-T14, and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

IBM welcomes your comments. Address your comments to:

IBM Corporation  
TPF Systems Information Development  
Mail Station P923  
2455 South Road  
Poughkeepsie, NY 12601-5400  
USA

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

© **Copyright International Business Machines Corporation 1994, 2002. All rights reserved.**

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

# Contents

<b>Figures</b> . . . . .	xi
<b>Tables</b> . . . . .	xv
<b>Notices</b> . . . . .	xvii
Trademarks . . . . .	xvii
<b>About This Book</b> . . . . .	xix
Before You Begin . . . . .	xix
Who Should Read This Book . . . . .	xix
Conventions Used in the TPF Library . . . . .	xix
How to Read the Syntax Diagrams . . . . .	xx
Related Information . . . . .	xxiii
IBM Transaction Processing Facility (TPF) 4.1 Books . . . . .	xxiii
Miscellaneous IBM Books . . . . .	xxiv
Online Information . . . . .	xxv
How to Send Your Comments . . . . .	xxv
<b>Introduction to TPF SNA Support</b> . . . . .	1
TPF SNA Overview . . . . .	1
Basic SNA Terms and Concepts . . . . .	3
<b>TPF Message Processing Flow Overview</b> . . . . .	5
TPF Inbound Message Flow . . . . .	5
Protocol Handling . . . . .	6
Presentation Handling . . . . .	6
TPF Outbound Message Flow . . . . .	7
<b>TPF SNA</b> . . . . .	11
SNA Data Transfer . . . . .	11
Common Characteristics of NCP and CTC Data Transfer . . . . .	11
NCP Data Transfer . . . . .	11
Unique Characteristics of CTC Data Transfer . . . . .	11
Multiple-Domain Networks . . . . .	12
Subarea (PU_5) Environment . . . . .	12
Cross-Domain Links . . . . .	12
Cross-Domain Resource Manager (CDRM) . . . . .	14
Cross-Domain Sessions . . . . .	14
Low-Entry Networking (PUT 2.1) Environment . . . . .	15
Control Points . . . . .	16
Control LUs . . . . .	16
3174 APPN Considerations . . . . .	16
Advanced Peer-to-Peer Networking (APPN) Environment . . . . .	16
Control Points . . . . .	16
Message Routing . . . . .	16
<b>TPF Applications</b> . . . . .	19
Considerations for Developing Applications That Access NCB Records . . . . .	19
TPF as Host Node SLU . . . . .	19
Activating LU-LU Sessions with TPF . . . . .	20
<b>TPF Application Programming Considerations</b> . . . . .	21
Function Management Header (FMH) . . . . .	22

Programming Considerations for Session Initiation . . . . .	22
Synchronizing Messages with Sequence Numbers . . . . .	23
3600 Application Considerations . . . . .	25
Data Transmission from 3600 to a TPF Application. . . . .	25
Data Transmission from a TPF Application to a 3600 . . . . .	25
3270 Application Considerations . . . . .	26
Non-SNA 3270 Application Considerations . . . . .	26
SNA 3270 Application Considerations . . . . .	26
Host Node Application Considerations . . . . .	26
IMS Relay Application Considerations . . . . .	28
CICS Relay Application Considerations . . . . .	29
X.25 NPSI Support . . . . .	29
NPSI Message Length Considerations . . . . .	30
Request Unit Chaining . . . . .	31
RU Sizes . . . . .	31
NPSI GATE/Fast Transaction Processing Interface (GATE/FTPI). . . . .	32
Message Traffic Multiplexing . . . . .	33
Input Processing . . . . .	34
Multiplexing Output Traffic . . . . .	35
Application Program Interface . . . . .	35
Network Definition. . . . .	35
Operations . . . . .	40
Activating the TPF CTCP . . . . .	40
Activating Multi-Channel Links . . . . .	41
Traces . . . . .	42
TPF Mapping Support . . . . .	42
<b>Airlines Line Control (ALC) Support through SNA . . . . .</b>	<b>45</b>
TPF Host Interfaces . . . . .	45
Session Control via VTAM SSCP . . . . .	46
Command Support . . . . .	48
Addressing an ALC Device . . . . .	48
TPF/NEF SNA Multiple Host Support. . . . .	49
X.25 ALC Interface for Switched or Permanent Virtual Circuits (XALCI) . . . . .	50
User-Replaceable PSV Routine. . . . .	51
Network Definition. . . . .	51
XALCI Configuration . . . . .	56
<b>TPF Advanced Program-to-Program Communications . . . . .</b>	<b>59</b>
TPF/APPC Components . . . . .	59
TPF/APPC Installation Checklist . . . . .	60
General Installation Tasks . . . . .	60
Installation Tasks for Low-Entry Networking (LEN) . . . . .	63
Defining TPF/APPC LUs to the Network . . . . .	63
Defining TPF/APPC LUs to a VTAM Subsystem. . . . .	66
Defining a TPF/APPC LU to a PS/2 Workstation (Using a Generic Name) . . . . .	66
TPF/APPC Control Blocks. . . . .	67
TPF/APPC Conversation Control Block . . . . .	67
TPF/APPC Session Control Block . . . . .	68
TPF/APPC Work Blocks . . . . .	68
TPF/APPC Work Block . . . . .	68
TPF/APPC Change Number of Sessions Work Block . . . . .	68
TPF/APPC Partner Work Block . . . . .	69
TPF/APPC Half-Session to Presentation Services Record . . . . .	69
Presentation Services . . . . .	69
TPF Transaction Program ATTACH Interface . . . . .	69

Conversation Verbs . . . . .	71
Change Number of Sessions. . . . .	78
Finite State Machine . . . . .	80
Generating the Side Information Table for Mapped Conversations . . . . .	83
Creating the Input File . . . . .	85
Running CHQI . . . . .	92
Side Information Table Offline Program Output . . . . .	93
Loading the Side Information Data Set to TPF . . . . .	96
Resource Manager . . . . .	98
Session Manager . . . . .	99
Initiating a Session . . . . .	99
Session Termination . . . . .	99
Message Flow . . . . .	99
Inbound Message Queuing . . . . .	100
System Restart . . . . .	101
Session Considerations . . . . .	101
Conversation Considerations . . . . .	102
Subsystem Considerations . . . . .	102
Considerations for a Tightly Coupled Environment . . . . .	102
Considerations for a Loosely Coupled Environment . . . . .	103
Loosely Coupled Complex Example. . . . .	103
Loosely Coupled Installation Checklist . . . . .	105
User Exits . . . . .	107
Transaction Program Design Considerations . . . . .	107
Traditional LU 6.2 Conversations . . . . .	107
Pipeline LU 6.2 Conversations. . . . .	108
Shared LU 6.2 Conversations . . . . .	110
Sample Transaction Programs . . . . .	113
Sample Transaction Program Functions . . . . .	114
Requesting TPF Transaction Program . . . . .	114
Requested TPF Transaction Program . . . . .	116
Sample Requesting TPF Transaction Program . . . . .	117
Sample Requested TPF Transaction Program . . . . .	121
<b>High-Performance Routing (HPR) Support . . . . .</b>	<b>127</b>
Benefits of Using HPR Support . . . . .	127
HPR Node Types . . . . .	127
ANR Labels . . . . .	128
Activating Links . . . . .	130
RTP Connections . . . . .	131
TCIDs. . . . .	134
Starting RTP Connections . . . . .	134
Deactivating RTP Connections. . . . .	134
Displaying RTP Connections . . . . .	134
Starting LU-LU Sessions . . . . .	135
ROUTE_SETUP Process . . . . .	136
LU-LU Session Activation Flows . . . . .	138
Session Addresses . . . . .	140
Path Switches. . . . .	141
Path Switch Process . . . . .	141
Path Switch Timer . . . . .	143
Detecting Network Failures . . . . .	144
Short Request Timer . . . . .	144
Alive Timer . . . . .	145
NLPs . . . . .	145
NHDRs . . . . .	147

THDRs . . . . .	149
Data . . . . .	151
HPR Control Messages . . . . .	151
Network Considerations for NLPs . . . . .	152
HPR Control Blocks . . . . .	152
RTPCB Table . . . . .	152
HPRSAT . . . . .	155
HPRMT . . . . .	156
Relationship with Other SNA Control Blocks. . . . .	157
Host IPL Considerations . . . . .	163
RTP Connection Resynchronization Process . . . . .	163
CP-CP Session Failures . . . . .	167
Flow Control . . . . .	168
ARB Pacing . . . . .	168
RTP Output Queue . . . . .	170
HPR Output Messages . . . . .	172
Selective Retransmission. . . . .	173
Retransmitting Output Messages . . . . .	174
Requesting That Input Messages Be Retransmitted . . . . .	174
Segmentation and Reassembly . . . . .	174
Segmenting Output Messages . . . . .	175
Reassembling Input Messages . . . . .	176
Installation and Tuning . . . . .	176
Diagnostic Information. . . . .	177
Sense Codes Unique to the TPF System . . . . .	177
<b>TPF/SNA Control Block Structures . . . . .</b>	<b>179</b>
Resource Vector Table (RVT) and Related Control Block Structures . . . . .	179
How the RVT Is Organized . . . . .	179
RVT-Related Control Block Structures . . . . .	181
Node Control Block (NCB) Records and Related Structures . . . . .	183
Types of NCB Records . . . . .	183
NCB-Related Control Block Structures . . . . .	184
Displaying Information about NCB Records and Related Structures . . . . .	187
Initializing NCB Records . . . . .	187
Reclaiming NCB Directory Records and NCB Records . . . . .	187
Increasing the Number of NCB Directory Records in the TPF System . . . . .	188
Performance Considerations for Accessing NCB Records. . . . .	190
Developing Applications That Retrieve NCB Records . . . . .	190
Allocating or Retrieving a Scratch Pad Area (SPA) for a Dynamic LU . . . . .	190
<b>Defining SNA Resources to the TPF System . . . . .</b>	<b>193</b>
Using the OSTG Program to Define SNA Resources . . . . .	193
Important Considerations. . . . .	194
Loading Resource Definitions by Performing a Fresh Load . . . . .	194
Forcing a Fresh Load during the Next IPL . . . . .	195
Loading Resource Definitions by Performing a Dynamic Load . . . . .	195
Falling Back to the Old Resource Definitions . . . . .	196
Displaying Status Information about the Load Functions . . . . .	198
Using the ZNDYN ADD Command to Define SNA Resources . . . . .	198
Restrictions. . . . .	198
Important Considerations. . . . .	198
Using the ZNDYN CHANGE Command to Change SNA Resource Definitions . . . . .	199
Using Dynamic LU Support to Define SNA Resources . . . . .	199
Defining Remote LU Resources . . . . .	199
Defining Remote ALS Resources. . . . .	200



<b>Activating and Deactivating Resources</b>	201
Activating and Deactivating a Shared NCP	201
Starting and Stopping Application Programs	201
Activating and Deactivating Cross-Domain Resource Managers	202
Activating and Deactivating Control LU-Logon Manager Sessions	204
Activating and Deactivating APPN CP-CP Sessions	204
CP-CP Session Considerations When Running Loosely Coupled	205
Activating LU-LU Sessions	206
Activating LU-LU Sessions Other Than LU 6.2	207
Activating LU 6.2 Sessions	207
PU 5 Environment	208
PU 2.1 LEN Environment	208
PU 2.1 APPN Environment	208
Mixed PU 5 and PU 2.1 Environment	209
<b>LU-LU Sessions in an APPN Network</b>	211
Predefining TPF LUs to the APPN Network	211
APPN LU Registration Process	211
Remote Initiated LU-LU Sessions	212
<b>Network Services Application Interfaces</b>	213
ISHLL Macro	213
Session Management Request Services	213
Starting Session Management Request Services	213
Requesting Session Resynchronization	214
Requesting Session Termination Interface	215
Session Status Awareness Services	215
Activating Session Status Awareness Services	216
Starting Session Started Notification	216
Starting Session Ended Notification	217
<b>PU 5 FID4 Considerations</b>	219
NCP Support	219
Channel Contact	219
NCP Considerations	221
CTC Support	222
Pre-Channel Contact/Priming	222
Channel Contact	223
Loosely Coupled Considerations	227
Session Initiation	227
VTAM Considerations	227
Class of Service, Virtual Routes, and Transmission Priority	227
Other VTAM Considerations	228
VTAM Considerations for NCP 5.3	228
LU-LU Session VR Assignment for CTC	228
Virtual Route (VR) Activation	228
Virtual Route Deactivation	229
Network Flow Control	230
Software IPL Considerations	231
Hardware IPL Considerations	232
Extended Network Addressing	233
Network Definition	233
SNA Network Interconnection (SNI) Considerations	233
<b>PU 2.1 Considerations</b>	237
General Information	237

Session Control . . . . .	237
Session Identifiers . . . . .	237
Extended BIND . . . . .	238
SNA Restart and ALS Discovery . . . . .	239
37x5 Considerations . . . . .	240
37x5 and APPN Considerations . . . . .	240
37x5 and LEN Considerations . . . . .	241
Non-37x5 Considerations . . . . .	244
Non-37x5 and APPN Considerations . . . . .	244
Non-37x5 and LEN Considerations . . . . .	244
3174 Considerations . . . . .	244
<b>SNA Message Protocol . . . . .</b>	<b>247</b>
Response Protocol . . . . .	247
Recoverable and Non-Recoverable Messages . . . . .	247
Single-Segment Messages . . . . .	247
Chained and Segmented Messages. . . . .	248
Multithread Processing . . . . .	250
Bracket Support . . . . .	250
Unsolicited Messages . . . . .	250
Unsolicited Messages Destined for a LNIATA/LEID . . . . .	250
Unsolicited Messages Destined for a RID. . . . .	250
3614/3624 Message Processing . . . . .	251
3614/3624 Session Initiation and Application Considerations. . . . .	251
<b>User Routines . . . . .</b>	<b>253</b>
Process Selection Vector (PSV) . . . . .	253
Input Message PSV Processing . . . . .	253
Output Message PSV Processing . . . . .	254
PSV Interface . . . . .	255
PSV Output Message Queueing . . . . .	259
Defining PSV Routines . . . . .	259
Logical End-Point Identifiers, Terminals, and PSV Routines . . . . .	260
Data Flow Control (DFC) Considerations . . . . .	260
User DFC Interface . . . . .	261
Outbound Message Interface . . . . .	262
Pacing Considerations. . . . .	262
Undeliverable Message Considerations . . . . .	262
Response Protocols/Error Handling . . . . .	262
Input Message Router Exit . . . . .	265
<b>Queue Manager . . . . .</b>	<b>269</b>
Queue Manager Interface . . . . .	270
Input Interface. . . . .	270
Output Interface . . . . .	272
General Return Codes . . . . .	273
Detailed Return Codes . . . . .	273
<b>Diagnostic Aids. . . . .</b>	<b>275</b>
Trace Functions . . . . .	275
Operating System Traces . . . . .	275
Path Information Unit (PIU) Trace Facility. . . . .	275
SNA I/O Trace Facility. . . . .	275
Reliability and Serviceability. . . . .	276
Error Detection and Feedback. . . . .	276
Hardware Error Recovery . . . . .	277

NCP Slowdown . . . . .	277
CTC Slowdown . . . . .	277
<b>Data Collection/Reduction and Test Tools . . . . .</b>	<b>279</b>
<b>Appendix A. Logical Unit Status (LUSTAT) . . . . .</b>	<b>281</b>
<b>Appendix B. General Format of the SNA BIND Command . . . . .</b>	<b>283</b>
BIND Images for TPF Supported Secondary Logical Units . . . . .	283
Acceptable BIND Image For a Local Host Node SLU . . . . .	284
<b>Appendix C. Interface Requirements for System Utility Programs . . . . .</b>	<b>287</b>
RID and RVT Conversions . . . . .	287
Retrieving NCB and SPA Data Records (CSNB) . . . . .	287
Select A Thread Utility Routine (SELEC) . . . . .	288
Select A Thread Utility Program (CSF0) . . . . .	288
<b>Appendix D. Using the Path Information Unit (PIU) Trace Facility . . . . .</b>	<b>291</b>
About the PIU Trace Table . . . . .	291
Starting the PIU Trace Facility and Specifying Which Data to Trace . . . . .	291
Stopping the PIU Trace Facility . . . . .	292
Defining How Much of the RU to Store in the PIU Trace Table . . . . .	293
Writing the PIU Trace Table to a Real-Time Tape . . . . .	293
Displaying Information about the PIU Trace Facility . . . . .	294
Examples . . . . .	294
Displaying the PIU Trace Table Online . . . . .	295
Creating a Compacted Display of the PIU Trace Table . . . . .	296
Creating a Formatted Display of the PIU Trace Table . . . . .	297
Using the Offline PIU Print (PIUPRT) Utility to Create a PIUPRT Report . . . . .	301
Sample JCL for the PIUPRT Utility . . . . .	302
Defining the PIUPRT Report . . . . .	303
PIUPRT Utility Return Codes . . . . .	312
RH Indicators . . . . .	312
Including the PIU Trace Table in System Error Dumps . . . . .	313
<b>Appendix E. VTAM Mode Table Entries. . . . .</b>	<b>315</b>
<b>Appendix F. TPF Message Processing Flow User Extensions . . . . .</b>	<b>317</b>
TPF Inbound Message Flow Extensions . . . . .	317
TPF Outbound Message Flow Extensions . . . . .	319
ROUTC Exit . . . . .	319
<b>Appendix G. Sample TPF CTCP Implementation Using PSVs . . . . .</b>	<b>321</b>
Introduction to CTCP Functions . . . . .	321
Generalized Access to X.25 Transport Extension (GATE) . . . . .	322
User CTCP Control Blocks . . . . .	322
X.25 Network Control Block. . . . .	324
X.25 Link Control Block (XLCB) . . . . .	324
Virtual Circuit Control Block (VCCB). . . . .	324
User Terminal Control Blocks . . . . .	325
End Point Control Block (EPCB) . . . . .	325
Contact Point Control Block (CPCB) . . . . .	326
Control Block Residence . . . . .	326
TPF SNA Session Awareness . . . . .	327
Session Start . . . . .	328
Session End . . . . .	328

TPF X.25 Command Message Flows . . . . .	328
PSV Processing of NPSI/GATE Commands . . . . .	329
TPF X.25 Data Message Flows . . . . .	335
CTCP Input Data Message Processing . . . . .	335
CTCP Output Data Message Processing . . . . .	338
<b>Appendix H. SNA Command Flow.</b> . . . . .	345
PU 5 and PU 2.1 LEN Session Activation. . . . .	345
CDRM-CDRM Session Activation. . . . .	345
PU 5 and PU 2.1 Session Deactivation . . . . .	360
APPN Session Activation. . . . .	374
APPN LU-LU Session Deactivation . . . . .	383
APPN-Subarea Migration Flows . . . . .	384
More on SNA Command Flow . . . . .	391
<b>Appendix I. TPF Sense Code Processing</b> . . . . .	393
SNA Timeout Processing. . . . .	395
<b>Index</b> . . . . .	397

# Figures

1. TPF Inbound Message Flow . . . . .	5
2. TPF Outbound Message Flow . . . . .	8
3. Cross-Domain Communication Link . . . . .	13
4. Multiple Cross-Domain Links . . . . .	14
5. Cross-Domain LU to LU Session Data Flow . . . . .	15
6. TPF Sent Set-and-Test Sequence Numbers Action Code . . . . .	23
7. 3601 Set-and-Test Sequence Numbers Action Code Response . . . . .	24
8. 3601 STSN Response . . . . .	24
9. TPF Action Resulting from 3601 STSN Response . . . . .	25
10. FTPI Message Blocking Format . . . . .	33
11. FTPI Command Content . . . . .	34
12. RSC Definition Statement for an FTPI LU . . . . .	36
13. RSC Coding Example Statement for an FTPI LU . . . . .	37
14. VTAM Log Mode Entry for FTPI . . . . .	37
15. Organization of NPSI Macros . . . . .	38
16. Organization of X25.CPL Macro . . . . .	38
17. CTCP Pseudo LU Coding Example . . . . .	39
18. X.25 MCH Macro . . . . .	39
19. X.25 Link Definition . . . . .	40
20. Pseudo Link Activation . . . . .	41
21. Multi-Channel Link Activation . . . . .	42
22. Multiple Host Network in NEF . . . . .	50
23. RSC Statement Definition Statement for an FTPI LU . . . . .	52
24. RSC Coding Example Statement for an XALCI LU . . . . .	52
25. XALCI PSV Entry Shipped with TPF . . . . .	53
26. Create PSV Table Macro . . . . .	53
27. Create PSV Entry Macro . . . . .	53
28. Sample XALCI PSV Definition Statements . . . . .	53
29. XALCI Header Formats . . . . .	55
30. XALCI Configuration . . . . .	56
31. TPF/APPC Component Flow Diagram . . . . .	60
32. Defining TPF/APPC LUs to VTAM across a PU 5 Connection . . . . .	64
33. Defining TPF/APPC Alias LU Names to VTAM across a PU 5 CTC Connection . . . . .	64
34. Defining TPF/APPC LUs across PU 2.1 LEN Connections . . . . .	65
35. Defining TPF/APPC LUs across PU 2.1 APPN Connections . . . . .	66
36. Defining a TPF/APPC LU to CICS . . . . .	66
37. TPF Transaction Program ATTACH Interface . . . . .	70
38. ALLOCATE Verb . . . . .	73
39. CONFIRM Verb . . . . .	73
40. CONFIRMED Verb . . . . .	74
41. DEALLOCATE Verb . . . . .	74
42. FLUSH Verb . . . . .	74
43. GET_ATTRIBUTES Verb . . . . .	75
44. GET_TYPE Verb . . . . .	75
45. POST_ON_RECEIPT Verb . . . . .	75
46. PREPARE_TO_RECEIVE Verb . . . . .	76
47. RECEIVE Verb . . . . .	76
48. REQUEST_TO_SEND Verb . . . . .	76
49. SEND_DATA Verb . . . . .	77
50. SEND_ERROR Verb . . . . .	77
51. TEST Verb . . . . .	77
52. WAIT Verb . . . . .	78
53. Generating the Side Information Table . . . . .	84

54. Example of an Input File for CHQI . . . . .	90
55. TPF Console Display . . . . .	91
56. Example CHQI Output Listing . . . . .	95
57. Loading a Side Information Offline Generated Tape . . . . .	97
58. Inbound Queue . . . . .	101
59. Loosely Coupled Complex Example for TPF/APPC Sessions . . . . .	105
60. Traditional LU 6.2 Conversations . . . . .	108
61. Pipeline LU 6.2 Conversations . . . . .	110
62. Shared LU 6.2 Conversations. . . . .	112
63. Sample Network with APPN TGs and HPR ANR Labels . . . . .	129
64. Sample Networks and RTP Connections. . . . .	132
65. Combination of HPR Support and APPN. . . . .	133
66. ROUTE_SETUP Process . . . . .	137
67. LU-LU Session Activation, HPR Support Is Not Used . . . . .	138
68. LU-LU Session Activation, a New RTP Connection Is Started . . . . .	139
69. LU-LU Session Activation, an Existing RTP Connection . . . . .	139
70. LU-LU Session Activation, Part of the Route . . . . .	139
71. Session Address Usage . . . . .	140
72. Path Switch Process . . . . .	142
73. Comparing an NLP to PIUs . . . . .	146
74. Routing an NLP through the Network . . . . .	148
75. RTPCB Table Layout . . . . .	153
76. LU RVT Entries for PU 5 and PU 2.1 . . . . .	158
77. HPR LU RVT Entries, RTP Connections Are Active . . . . .	160
78. HPR LU RVT Entries, Path Switch Is in Progress . . . . .	161
79. HPR LU RVT Entries, Path Switch Is Completed. . . . .	162
80. RTP Connection Resynchronization, before the IPL . . . . .	165
81. RTP Connection Resynchronization, after the IPL . . . . .	166
82. HPR SOUTC Block Types . . . . .	173
83. Segmenting an Output Message . . . . .	175
84. RVT Example . . . . .	180
85. RNHCT, RNHPT, and RNHET Example . . . . .	182
86. NCB Directory Record Example . . . . .	186
87. The ISHLL Macro . . . . .	213
88. NCP Gen with TPF Channel-Attached as a PU Type 5 Node . . . . .	222
89. XID Processing Example 1. . . . .	224
90. XID Processing Example 2. . . . .	224
91. XID Processing Example 3. . . . .	225
92. NCP Gen with TPF Channel-Attached as a PU Type 2.1 LEN Node . . . . .	242
93. Input Message Processing . . . . .	253
94. Extended TPF Outbound Message Flow, ROUTC and PSV Exits. . . . .	255
95. Compacted Display of the PIU Trace Table. . . . .	297
96. Formatted Display of the PIU Trace Table . . . . .	301
97. JCL for the PIUPRT Utility . . . . .	302
98. Compacted PIUPRT Report . . . . .	308
99. Formatted PIUPRT Report . . . . .	312
100. VTAM Mode Table Entries - Non-3270 LU0. . . . .	315
101. VTAM Mode Table Entries - 3270s . . . . .	316
102. Extended TPF Inbound Message Flow . . . . .	318
103. Extended TPF Outbound Message Flow. . . . .	319
104. TPF/NPSI Sessions Involving a CTCP . . . . .	321
105. TPF X.25 Network Elements and Corresponding User Control Blocks . . . . .	324
106. TPF CTCP Control Block Structure. . . . .	327
107. Control Block Relationship for Inbound CALL_REQUEST Command . . . . .	330
108. Generic TPF Inbound X.25 CTCP Command/Reply Flow. . . . .	331
109. Control Block Relationship for Inbound CLEAR Command . . . . .	332

110. Control Block Relationship for CALL_OUT Request . . . . .	333
111. Generic TPF Outbound X.25 CTCP Command Processing Flow . . . . .	334
112. Control Block Relationship for Inbound CALL_CONFIRM Command . . . . .	334
113. Soft Copy Input Data Flow . . . . .	336
114. Hard Copy Input Data Flow . . . . .	337
115. Soft Copy Output Data Flow . . . . .	339
116. Hard Copy Enqueue, Dequeue, and Transmit . . . . .	340
117. Hard Copy Enqueue and Exit . . . . .	341
118. Hard Copy Queue Wake Up . . . . .	342
119. Hard Copy Repeat Last Message/Negative Acknowledgment . . . . .	343
120. CDRM-CDRM Session Started by the TPF System . . . . .	345
121. CDRM-CDRM Session Started by the Remote SSCP . . . . .	346
122. CDRM-CDRM Session Started by VTAM . . . . .	346
123. CDRM-CDRM Session Started from the TPF side . . . . .	347
124. APPL-APPL Session Started by TPF (PU 5) . . . . .	348
125. APPL-APPL Session Started by Remote LU (PU 5) . . . . .	349
126. APPL-APPL Session Started by Remote LU (PU 5) and is Queued . . . . .	350
127. APPL-APPL Session Started by Remote LU (PU 2.1 LEN) . . . . .	350
128. Host-Node SLU Session Started by TPF (PU 5) . . . . .	351
129. Host-Node SLU Session Started by Remote LU (PU 2.1 LEN) . . . . .	352
130. PU 5 FMMR-FMMR Session Initiation from the PLU . . . . .	353
131. PU 5 FMMR-FMMR Session Initiation from the SLU . . . . .	354
132. PU 2.1 LEN FMMR-FMMR Session Initiation . . . . .	354
133. 3270 Session Started by Remote Terminal (PU 5) . . . . .	355
134. 3270 Session Started by Remote Terminal (PU 2.1 LEN) . . . . .	356
135. PU 5 LU 6.2 Session Started by TPF PLU . . . . .	357
136. PU 5 LU 6.2 Session Started by Remote SLU . . . . .	357
137. PU 5 LU 6.2 Session Started by TPF SLU . . . . .	358
138. PU 5 LU 6.2 Session Started by Remote PLU . . . . .	359
139. PU 2.1 LEN LU 6.2 Session Started by TPF . . . . .	359
140. PU 2.1 LEN LU 6.2 Session Started by Remote SLU . . . . .	360
141. Normal CDRM-CDRM Session Deactivation . . . . .	361
142. Immediate CDRM-CDRM Session Deactivation . . . . .	362
143. Forced CDRM-CDRM Session Deactivation . . . . .	363
144. Normal APPL-APPL Session Deactivation . . . . .	364
145. Immediate APPL-APPL Session Deactivation . . . . .	365
146. Forced APPL-APPL Session Deactivation . . . . .	366
147. Normal Host-Node SLU Session Deactivation . . . . .	367
148. Immediate Host-Node SLU Session Deactivation . . . . .	368
149. Forced Host-Node SLU Session Deactivation . . . . .	369
150. Normal or Immediate FMMR-FMMR Session Deactivation . . . . .	370
151. Forced FMMR-FMMR Session Deactivation . . . . .	371
152. Normal LU 6.2 Session Deactivation . . . . .	372
153. Immediate LU 6.2 Session Deactivation . . . . .	373
154. Forced LU 6.2 Session Deactivation . . . . .	374
155. CP-CP Session Activation . . . . .	375
156. Session Started by TPF PLU (APPN) . . . . .	376
157. Session Started by TPF SLU (APPN) . . . . .	377
158. Session Started by Remote SLU (APPN), RSCV Provided . . . . .	377
159. Session Started by Remote SLU (APPN), RSCV Not Provided . . . . .	378
160. Session Started by Remote SLU (APPN), Session Queued . . . . .	379
161. Session Started by Remote PLU (APPN) . . . . .	379
162. Session Started by Remote PLU (APPN), Session Queued . . . . .	380
163. Printer Sharing, Request by TPF, Printer Available . . . . .	381
164. Printer Sharing, Request by TPF, Printer In Use . . . . .	382
165. Printer Sharing, TPF Requested to Release the Printer . . . . .	383

166. Normal Deactivation by Remote SLU (APPN) . . . . .	384
167. SLU Initiated, SLU in APPN, PLU in Subarea . . . . .	385
168. SLU Initiated, SLU in APPN, PLU in Subarea (PLU Location Unknown) . . . . .	386
169. PLU Initiated, SLU in APPN, PLU in Subarea . . . . .	387
170. SLU Initiated, PLU in APPN, SLU in Subarea (SC Provided) . . . . .	388
171. SLU Initiated, PLU in APPN, SLU in Subarea (SC Not Provided). . . . .	389
172. PLU Initiated, PLU in APPN, SLU in Subarea (SC Provided) . . . . .	390
173. PLU Initiated, PLU in APPN, SLU in Subarea (SC Not Provided). . . . .	391



---

## Tables

1. SNA Commands Supported for NPSI . . . . .	31
2. TPF/APPC Conversation Finite State Machine Matrix . . . . .	82
3. LU 6.2 Statistics for Processing 4 Messages . . . . .	113
4. LU 6.2 Statistics for Processing 100 Messages . . . . .	113
5. Comparing Traditional SNA Support to HPR Support . . . . .	127
6. Processing Output Message Queues . . . . .	170
7. CSU0 Interface . . . . .	213
8. Request Session Resynchronization Interface. . . . .	214
9. Request Session Termination Interface . . . . .	215
10. CSXD Interface . . . . .	216
11. Session Started Notification Interface . . . . .	216
12. Session Ended Notification Interface . . . . .	217
13. IPSVI - PSV Interface Dsect . . . . .	256
14. DFC ROUTC Interface . . . . .	261
15. Sample Negative Response Handler Interface . . . . .	263
16. LU0 (3600, NPSI) -RSP Analysis . . . . .	263
17. 3274/76 (LU1, LU2, LU3) Sense Code Table . . . . .	263
18. 3274/76 (LU1, LU2, LU3) LUSTAT SENSE TABLE . . . . .	264
19. Negative Response Processing Actions . . . . .	264
20. Queue Control Element Layout . . . . .	269
21. Queue Manager Parameter List . . . . .	270
22. Queue Manager Input Interface . . . . .	271
23. Queue Manager Output Interface . . . . .	273
24. BIND Images for TPF-Supported Secondary Logical Units . . . . .	283
25. Return Codes for the PIUPRT Utility . . . . .	312
26. Sense Codes and Actions . . . . .	393



---

## Notices

References in this book to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service in this book is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
USA

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  
Department 830A  
Mail Drop P131  
2455 South Road  
Poughkeepsie, NY 12601-5400  
USA

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

Any pointers in this book to non-IBM Web sites are provided for convenience only and do not in any way serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this book or accessed through an IBM Web site that is mentioned in this book.

---

## Trademarks

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

ACF/VTAM  
Advanced Peer-to-Peer Networking  
APPN  
CICS  
DB2  
IBM  
OS/2  
PS/2  
RISC System/6000  
SAA  
Systems Application Architecture

System/370  
VTAM.

Other company, product, and service names may be trademarks or service marks of others.

---

## About This Book

This book describes the functions provided for the Systems Network Architecture (SNA) data communications area.

In this book, abbreviations are often used instead of spelled-out terms. Every term is spelled out at first mention followed by the all-caps abbreviation enclosed in parentheses; for example, Systems Network Architecture (SNA). Abbreviations are defined again at various intervals throughout the book. In addition, the majority of abbreviations and their definitions are listed in the master glossary in the *TPF Library Guide*.

---

## Before You Begin

See *TPF Concepts and Structures* for an overview of the TPF system.

See *IBM Systems Network Architecture Concepts and Products* and *IBM Systems Network Architecture Technical Overview* for information about SNA concepts and terminology.

---

## Who Should Read This Book

This book is intended for systems programmers who are responsible for SNA data communications support.

---

## Conventions Used in the TPF Library

The TPF library uses the following conventions:

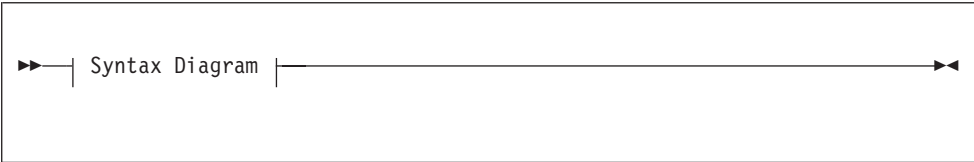
Conventions	Examples of Usage
<i>italic</i>	Used for important words and phrases. For example: A <i>database</i> is a collection of data.  Used to represent variable information. For example: Enter <b>ZFRST STATUS MODULE</b> <i>mod</i> , where <i>mod</i> is the module for which you want status.
<b>bold</b>	Used to represent text that you type. For example: Enter <b>ZNALS HELP</b> to obtain help information for the ZNALS command.  Used to represent variable information in C language. For example: <b>level</b>
monospaced	Used for messages and information that displays on a screen. For example: PROCESSING COMPLETED  Used for C language functions. For example: maskc  Used for examples. For example: maskc(MASKC_ENABLE, MASKC_IO);
<b><i>bold italic</i></b>	Used for emphasis. For example: You <b><i>must</i></b> type this command exactly as shown.

Conventions	Examples of Usage
<b><u>Bold underscore</u></b>	Used to indicate the default in a list of options. For example: <b>Keyword=OPTION1   <u>DEFAULT</u></b>
Vertical bar	Used to separate options in a list. (Also referred to as the OR symbol.) For example: <b>Keyword=Option1   Option2</b> <b>Note:</b> Sometimes the vertical bar is used as a <i>pipe</i> (which allows you to pass the output of one process as input to another process). The library information will clearly explain whenever the vertical bar is used for this reason.
CAPital LETters	Used to indicate valid abbreviations for keywords. For example: KEYWord=option
Scale	Used to indicate the column location of input. The scale begins at column position 1. The plus sign (+) represents increments of 5 and the numerals represent increments of 10 on the scale. The first plus sign (+) represents column position 5; numeral 1 shows column position 10; numeral 2 shows column position 20 and so on. The following example shows the required text and column position for the image clear card.  ...+....1....+....2....+....3....+....4....+....5....+....6....+....7...  LOADER    IMAGE   CLEAR  <b>Notes:</b> 1. The word LOADER must begin in column 1. 2. The word IMAGE must begin in column 10. 3. The word CLEAR must begin in column 16.

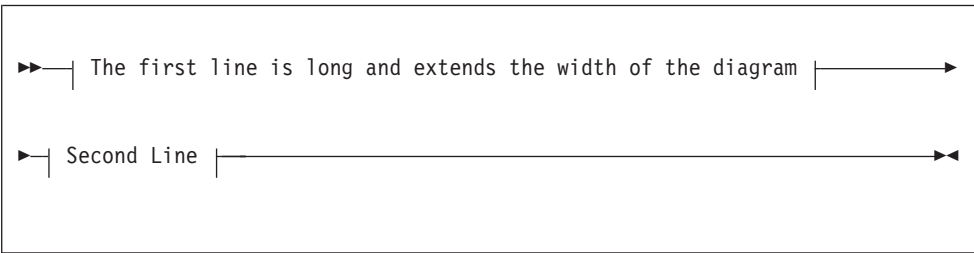
# How to Read the Syntax Diagrams

This section describes how to read the syntax diagrams (informally called *railroad tracks*) used in this book.

- Read the diagrams from left-to-right, top-to-bottom, following the main path line. Each diagram begins on the left with double arrowheads and ends on the right with 2 arrowheads facing each other.



- If a diagram is longer than one line, the first line ends with a single arrowhead and the second line begins with a single arrowhead.



- A word in all uppercase is a parameter that you must spell **exactly** as shown.

►►—PARAMETER—◄◄

- If you can abbreviate a parameter, the optional part of the parameter is shown in lowercase. (You must type the text that is shown in uppercase. You can type none, one, or more of the letters that are shown in lowercase.)

**Note:** Some TPF commands are case-sensitive and contain parameters that must be entered exactly as shown. This information is noted in the description of the appropriate commands.

►►—PARAMeter—◄◄

- A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

►►—*variable*—◄◄

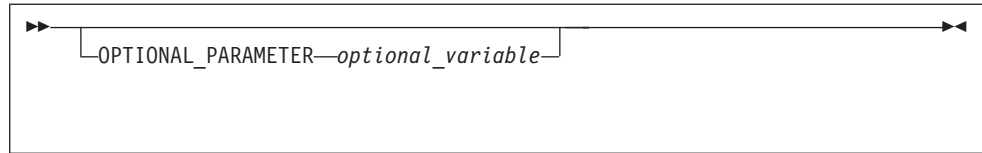
- Required parameters and variables are shown on the main path line. You must code required parameters and variables.

►►—REQUIRED\_PARAMETER—*required\_variable*—◄◄

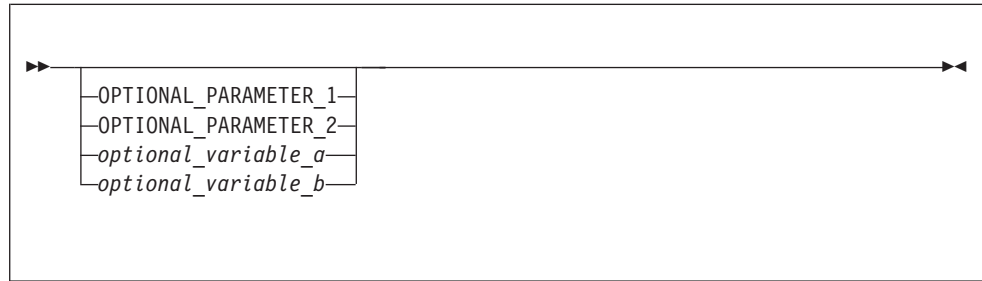
- If there is more than one mutually exclusive required parameter or variable to choose from, they are stacked vertically.

►►—  
—REQUIRED\_PARAMETER\_1—  
—REQUIRED\_PARAMETER\_2—  
—*required\_variable\_a*—  
—*required\_variable\_b*—◄◄

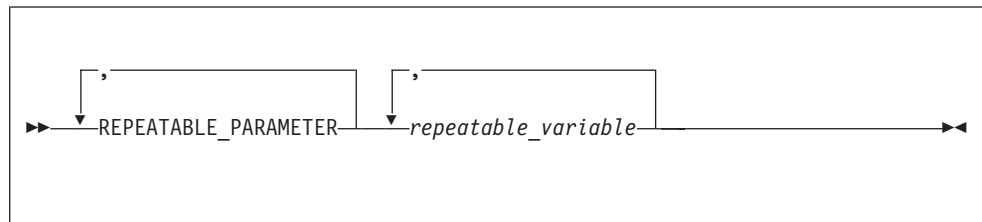
- Optional parameters and variables are shown below the main path line. You can choose not to code optional parameters and variables.



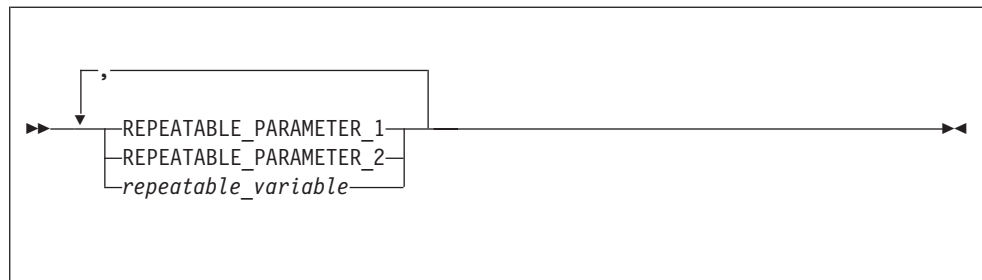
- If there is more than one mutually exclusive optional parameter or variable to choose from, they are stacked vertically below the main path line.



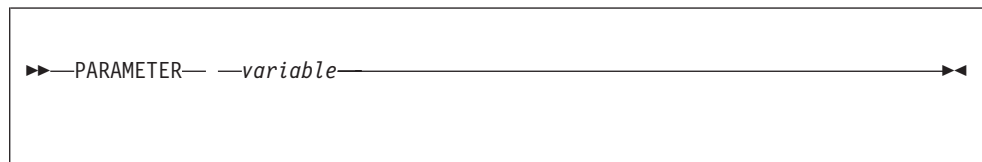
- An arrow returning to the left above a parameter or variable on the main path line means that the parameter or variable can be repeated. The comma (,) means that each parameter or variable must be separated from the next parameter or variable by a comma.



- An arrow returning to the left above a group of parameters or variables means that more than one can be selected, or a single one can be repeated.



- If a diagram shows a blank space, you must code the blank space as part of the syntax. In the following example, you must code **PARAMETER** variable.



- If a diagram shows a character that is not alphanumeric (such as commas, parentheses, periods, and equal signs), you must code the character as part of the syntax. In the following example, you must code **PARAMETER=(begin.end)**.



▶▶—PARAMETER=(begin.end)—▶▶

- Default parameters and values are shown above the main path line. The TPF system uses the default if you omit the parameter or value entirely.

▶▶—

DEFAULT
PARAMETER

—

0
variable

—▶▶

- References to syntax notes are shown as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.

▶▶—PARAMETER—▶▶<sup>(1)</sup>

**Notes:**

- 1    An example of a syntax note.

- Some diagrams contain *syntax fragments*, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

▶▶—| Reference to Syntax Fragment |—▶▶

**Syntax Fragment:**

—|1ST\_PARAMETER,2ND\_PARAMETER,3RD\_PARAMETER—|

---

## Related Information

A list of related information follows. For information on how to order or access any of this information, call your IBM representative.

### IBM Transaction Processing Facility (TPF) 4.1 Books

- *TPF ACF/SNA Network Generation*, SH31-0131
- *TPF C/C++ Language Support User's Guide*, SH31-0121
- *TPF Concepts and Structures*, GH31-0139
- *TPF Data Communications Services Reference*, SH31-0145

- *TPF General Macros*, SH31-0152
- *TPF System Installation Support Reference*, SH31-0149
- *TPF Main Supervisor Reference*, SH31-0159
- *TPF Migration Guide: Program Update Tapes*, GH31-0187
- *TPF Operations*, SH31-0162
- *TPF Program Development Support Reference*, SH31-0164
- *TPF System Performance and Measurement Reference*, SH31-0170
- *TPF System Generation*, SH31-0171
- *TPF System Macros*, SH31-0151.

## Miscellaneous IBM Books

- *ACF, NCP, ALCI General Information*, GH33-7012
- *CICS/DOS/VS Intercommunication Facilities Guide*, SC33-0133
- *Communications Manager/2 Network Administration and Subsystem Management Guide*, SC31-6168
- *Communications Manager/2 Workstation Installation and Configuration Guide*, SC31-7169
- *NCP/SSP/EP Resource Definition Reference* (order the correct version and release for your installation)
- *NPSI Host Programming*, SC30-3502
- *NPSI Planning and Installation*, SC30-3270
- *OS/2 Extended Edition APPC Programming Reference*, S01F-0295
- *OS/2 Extended Edition System Administration Guide for Communications*, G01F-0302
- *IBM Systems Network Architecture Advanced Peer-to-Peer Networking Architecture Reference*, SC30-3422
- *IBM Systems Network Architecture Concepts and Products*, GC30-3072
- *IBM Systems Network Architecture Format and Protocol Reference Manual: Architectural Logic*, SC30-3112
- *IBM Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*, SC30-3269
- *IBM Systems Network Architecture LU 6.2 Reference: Peer Protocols*, SC31-6808
- *IBM Systems Network Architecture Network Product Formats*, LY43-0081
- *IBM Systems Network Architecture Technical Overview*, GC30-3073
- *IBM Systems Network Architecture Transaction Programmer's Reference Manual for LU Type 6.2*, GC30-3084
- *IBM SNA Formats*, GA27-3136
- *Systems Application Architecture Common Programming Interface Communications Reference*, SC26-4399
- *VTAM Network Implementation Guide* (order the correct version and release for your installation)
- *VTAM Resource Definition Reference* (order the correct version and release for your installation)
- *3270 Data Stream Programmer's Reference*, GA23-0059
- *3600 Finance Communication System 3614 Programmer's Guide and Reference*, GC27-0010
- *3600 Finance Communication System 3624 Programmer's Guide*, GC66-0008.

## Online Information

- *Messages (Online)*
- *Messages (System Error and Offline).*

---

## 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 TPF information, use one of the methods that follow. Make sure you include the title and number of the book, the version of your product and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

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

- If you prefer to send your comments electronically, do either of the following:
  - Go to <http://www.ibm.com/tpf/pubs/tpfpubs.htm>.  
There you will find a link to a feedback page where you can enter and submit comments.
  - Send your comments by e-mail to [tpfid@us.ibm.com](mailto:tpfid@us.ibm.com)
- If you prefer to send your comments by mail, address your comments to:

IBM Corporation  
TPF Systems Information Development  
Mail Station P923  
2455 South Road  
Poughkeepsie, NY 12601-5400  
USA

- If you prefer to send your comments by FAX, use this number:
  - United States and Canada: 1 + 845 + 432 + 9788
  - Other countries: (international code) + 845 + 432 +9788



---

# Introduction to TPF SNA Support

Systems Network Architecture (SNA) is an IBM communications system framework. It is an architectural design that:

- Standardizes network operations
- Defines functional responsibility for each network component
- Defines protocol support of distributed functions.

SNA benefits include:

- Distribution of function
- Device and attachment independence
- Configuration flexibility.

This publication describes the TPF implementation of SNA, which allows you to interface TPF with SNA networks. Supporting SNA in a TPF system permits TPF application programs to communicate with devices that adhere to SNA standards.

In order to better understand the material covered in this publication, it is recommended that you read the following two sections. The first section is an overview of TPF SNA. The second section describes some basic SNA terminology.

---

## TPF SNA Overview

The SNA support provided allows TPF to communicate with other systems including other TPF systems, or devices (both SNA and non-SNA, such as ALC) across an SNA interface.

Networks generally have a mix of systems, usually ACF/VTAMs and TPF. It is required that an ACF/VTAM system, **not** TPF, be designated as the network manager (that is, the owner of the communications networks' resources), even if all or most of the data traffic is directed toward TPF.

Connections to the network can be both PU 5 connections and PU 2.1 connections. TPF as a PU 5 node connects to a communication controller running NCP or to an SNA channel-to-channel (CTC) device.

The TPF system can connect to a PU 2.1 network as either a low-entry networking (LEN) node or Advanced Peer-to-Peer Networking (APPN) node. In an APPN environment, the TPF system can only be an end node (EN), not a network node (NN).

TPF Network Control Program (NCP) support depends on:

- Subarea support (T5), which requires SNA Network Interconnection (SNI)
- T2.1 low-entry networking (LEN) support, which requires NCP Version 4 Release 3, or later
- T2.1 Advanced Peer-to-Peer Networking (APPN) support, which requires NCP Version 6 Release 2, or later.

All subsequent occurrences of NCP in the text of this book are superseded by the previous information, unless otherwise specified.

The SNA Network Interconnection (SNI) feature is required in the NCP to connect TPF as a PU 5 node. Format ID 4 (FID4) path information units (PIUs) are used to send messages across the connection. TPF's connection uses a single virtual route on the TPF side of the gateway, but sessions on the VTAMs side can take advantage of the full SNA functions available.

For SNA channel-to-channel (CTC), TPF appears to a remote host as a PU 5 (T5) node using FID4 PIUs. TPF and the remote system exchange identifications (XIDs) and are able to establish sessions across a maximum of 2 virtual routes using the CTC connection. Throughout the remainder of this publication, CTC will be used as an abbreviation for any SNA channel-to-channel link, device, and others.

TPF as a PU 2.1 node connects to NCP using the FID2 interface. As a PU 2.1 node, TPF no longer has a unique subarea but appears to the network as a PU in the subarea of each channel, attached NCP.

The TPF system also connects to a 3174 Advanced Peer-to-Peer Networking (APPN) controller as a PU 2.1 node. Sessions can be established between TPF applications and as many as 255 terminals (LUs) on a token ring connected to and managed by the 3174 controller. When the TPF system connects as a LEN node, only sessions using the LU 6.2 protocol are supported, and the sessions must be started by the remote LUs. When the TPF system connects as an APPN node and the 3174 is defined as an APPN network node, all LU protocols are supported.

The TPF system also connects to a RISC System/6000 system as a PU 2.1 node. When the TPF system connects as a LEN node, only sessions using the LU 6.2 protocol are supported, and the sessions must be started by the remote LUs. When the TPF system connects as an APPN node and the RISC System/6000 system is defined as an APPN network node, all LU protocols are supported.

For an Airlines Line Control (ALC) network, you must install the:

- Network Extension Facility (NEF2 PRPQ #P85025) or the ALC Interface (ALCI) feature of ACF/NCP
- TPF-provided Airline X.25 (AX.25) support with the NCP Packet Switching Interface (NPSI)
- TPF-provided XALCI support with the NCP Packet Switching Interface General Access to X.25 Transport Extension/Fast Transaction Processing Interface (NPSI GATE/FTPI).

**Note:** NEF/ALCI and NPSI are separate product, each with their own set of publications. This publication addresses only the support provided by TPF. Before generating the TPF system, read the publications of the other products to define the new network. See *NPSI Planning and Installation* and *NPSI Host Programming* for information on NPSI and GATE/FTPI.

Both TPF SNA and ALC support are required for TPF NEF, AX.25, and XALCI. SNA LUs can access new applications<sup>1</sup>. For 3270 SNA terminals, a pseudo address (LEID) can be defined to enable these terminals to access old applications. ALC terminals, using TPF-provided NEF/AX.25/XALCI support, can be used with old or new applications.

---

1. Old applications require a LNIATA (LEID) interface, while new applications can use either RIDs (Resource Identifiers) or LNIATAs (LEIDs).

TPF can also interface with LUs using the LU 6.2 protocol. These LUs can access only applications that have been defined as LU 6.2 capable.

**Note:** TPF's LU 6.2 support is provided through the TPF Advanced Program-to-Program Communications (TPF/APPC) support package. Refer to "TPF Advanced Program-to-Program Communications" on page 59 for details about this support.

TPF supports the X.25 interface through the support of NCP Packet Switching Interface (NPSI). TPF provides connectivity support for the following Logical Link Control (LLC) types:

- LLC=0
- LLC=3
- LLC=4
- LLC=5

The High Performance Option (HPO) feature requires the TPF SNA support. HPO contains 2 components: the multiple database function (MDBF) and the loosely coupled facility (LC). These may be used separately or in combination.

When generating either of these components, the options needed for the system must be chosen ahead of time. In either case, the generation procedure involves the use of new macros plus new keywords on base generation macros. MDBF systems require a generation of each subsystem in addition to generating the base system. For LC systems only one base generation is performed, regardless of the number of processors in the complex.

---

## Basic SNA Terms and Concepts

A network addressable unit (NAU) is a resource managed by the communications system. NAUs provide a window through which users access the communications system network. An NAU is the origin and/or destination for information transmitted through the network. Each NAU has a unique network name that is translated into a unique address.

**Note:** Throughout this and related TPF publications, the terms node and NAU have the same meaning.

The communications program must formally connect two NAUs before data may be exchanged. This is referred to as a *session*. The four types of NAUs are:

- *System service control point (SSCP)* - The SSCP is a focal point within the SNA network. SSCPs manage the network, coordinate operator and problem-determination requests, and provide general support for users of the network. Multiple SSCPs divide the network into *domains* of control. Each SSCP controls the logical units and physical units within its domain.
- *Physical unit (PU)* - A physical unit is associated with each network resource (host application, communications cluster, logical unit) defined to the SSCP.
- *Logical unit (LU)* - A port through which users access the resources of the network. In SNA, both application programs and terminal operators can be defined as logical units. One or more logical units may be situated within a physical unit. SNA requires a hierarchical (primary-secondary) relationship between LUs in session. The primary logical unit (PLU) and the secondary logical unit (SLU) differ in the functions assigned to each. Primary logical units are responsible for:
  - Error recovery: After an error occurs, PLUs ensure communication is correctly reestablished. SLUs may only request error recovery.

- Establishing the LU to LU session: either an SLU or a PLU may request an LU-LU session.

Logical units are further defined as either *host node* or *non-host node*. Host node LUs reside in the CPU; non-host node LUs reside in the cluster controller. A cluster controller is a device that controls input/output operations for multiple devices (terminals) attached to it. Host node LUs may be either primary or secondary.

- *Control point (CP)* - A special LU that resides in each APPN node. CPs manage APPN networks and perform the tasks that the SSCPs perform in a PU 5 network. A control point in an APPN node communicates with an adjacent control point using special sessions called CP-CP sessions.

An *adjacent link station (ALS)* is the term used to describe a connection between the TPF system and a channel-attached NCP. With TPF's support of PU 2.1, the NCP appears to TPF as an ALS. ALSs exchange identification (XID) to discover each others characteristics. XID processing is also done for 3174 APPN connections and closely follows PU 2.1 ALS processing.

With SNA CTC, a channel contact procedure is implemented between TPF and channel-attached T5 nodes; that is, an explicit PU5 to PU5 session exists. This PU-PU session is comparable to the PU-PU sessions supported between NCPs. A formal channel protocol between the physical units is used. The session establishment and termination is initiated by the T5 node at the direction of its SSCP.

The TPF system can connect to a PU 2.1 network as either a low-entry networking (LEN) node or an Advanced Peer-to-Peer Networking (APPN) node, but not both. There is a systemwide switch indicating the mode in which the TPF system is running:

- *LEN mode*. The TPF system will connect to a PU 2.1 network as a low-entry networking (LEN) node.
- *APPN mode*. The TPF system will connect to a PU 2.1 network as an Advanced Peer-to-Peer Networking (APPN) node.



# TPF Message Processing Flow Overview

## TPF Inbound Message Flow

Figure 1 illustrates TPF's inbound message flow for ALC and SNA terminals. See the *TPF Data Communications Services Reference* additional information on message routing and message flow.

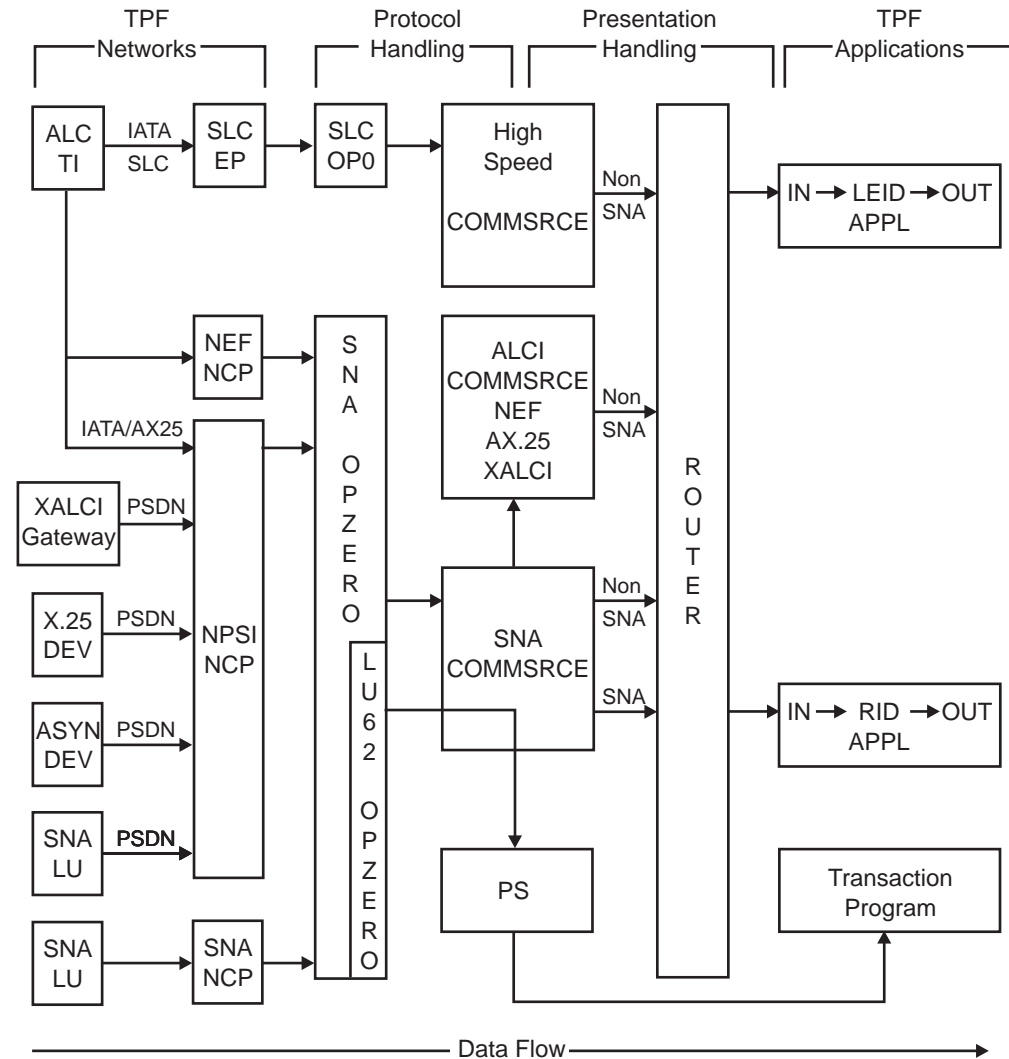


Figure 1. TPF Inbound Message Flow

The TPF inbound message flow consists of three components:

1. I/O Handling - Execution of the channel programs to transfer data between TPF and its local communications controllers.
2. Protocol Handling - Processing such as message format conversion, code translation, and error handling.
3. Presentation Handling - The delivery of the message to the application. This component also handles requests for logon, logoff and unsolicited messages.

## Protocol Handling

The primary function of this component is to move the message from an I/O buffer into a core block and then obtain an Entry Control Block (ECB). The core block containing the message is attached to the ECB and the message is passed to either:

- The Communications Source (COMMSRCE) program for data message handling.
- The output message transmitter (OMT or LMT) for response handling. Typically if the response is an acknowledgement, the next message on queue is sent. When the response is a negative acknowledgment, the message is retransmitted.
- The SNA Command handler for command request and response handling.

Other ancillary Protocol Handling functions include:

- Starting optional system routines for tracing messages during their processing.
- Updating data collection counters for offline performance analysis.
- Setting the ECB flag byte to indicate the origin of the message to assist in dump analysis.
- Setting the ECB field EBROUT to the Logical End-Point Identifier (LEID) or RID of the terminal to allow System Error processing in event of a failure.

## Presentation Handling

TPF passes a data message to the Communications Source (COMMSRCE) program that handles its specific line discipline or terminal type. The data message can be input to an application, a logon/logoff request, or an unsolicited message request. In general, COMMSRCE converts the terminal's network address into an address used internally by TPF. The internal address format is a RID for SNA sources or an LEID for non-SNA sources (terminals). If the destination of the message from an SNA source is a non-SNA application, the RID format is then converted to an LEID format. The message is then edited to remove imbedded control characters and examined to determine whether it is a data, logon/logoff, or request for unsolicited messages. The edited message is passed to one of three system services for subsequent processing:

1. Data messages are passed to the Router program.
2. Logon or logoff messages are passed to the Log Processor.
3. Request for unsolicited messages are passed to the Unsolicited Message Package.

### ROUTER Processing

The ROUTER ensures that the message can be delivered to the application. The program checks to verify that the:

- Application is available and active
- System and subsystem state allow the message to be delivered.

If the message cannot be delivered, then a message is returned to the originator stating why the message is undeliverable. The text of the messages returned by the system is:

- APPLIC appl NOT ACTIVE
- SYS.ERR ON MSG FROM appl MSG LOST
- ACP CANNOT ACCESS appl LOGOFF
- APPLIC appl NOT AVAIL. LOGOFF
- CANNOT ACCESS appl SYS IN 1052
- RESTRICTED. CRAT TERM INPUT ONLY

- FILE ERR. CANNOT DELIVER MSG TO appl
- CANNOT ACCESS appl SYS IN RESTART
- CANNOT ACCESS appl PROCESSOR INACTIVE

**Note:** The lower case letters 'appl' in these descriptions are replaced with the actual four character TPF application name when the message is returned to the originator.

If the message can be delivered, the application is entered with the message attached to level 0 of the ECB and a Routing Control Parameter List (RCPL) in EBW000 of the ECB.

The application is responsible for decoding the transaction code and starting the appropriate programs. If the message was from a 3270 terminal and the application requires ALC type input, the 3270 Simulator package may be called to reformat the message text. When processing is complete the application issues a ROUTC macro to send a reply to the terminal.

---

## TPF Outbound Message Flow

Figure 2 on page 8 shows TPF's outbound message flow for ALC and SNA terminals. See the *TPF Data Communications Services Reference* for additional information on message routing and message flow.

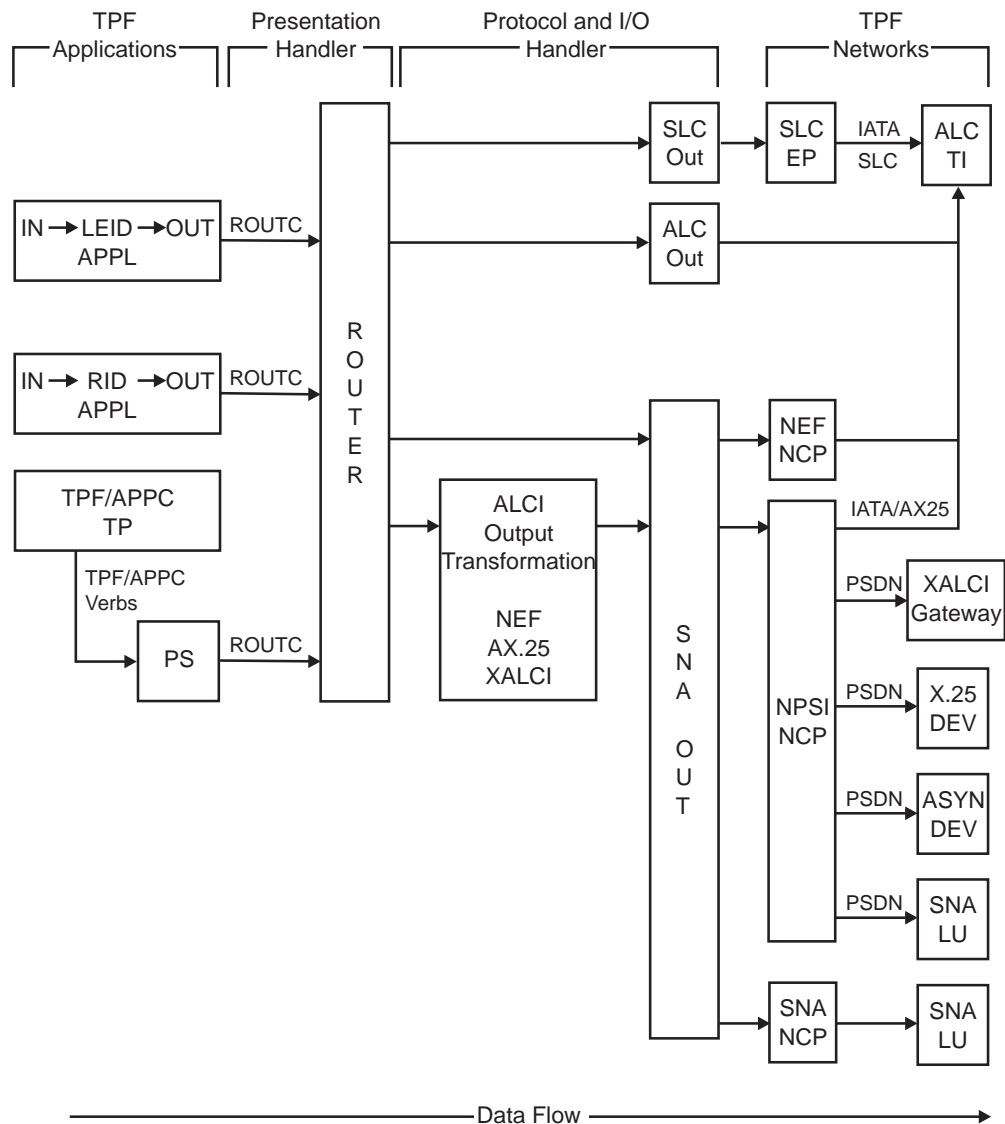


Figure 2. TPF Outbound Message Flow

The output message processing starts when the application issues a ROUTC macro. The operands of the ROUTC macro require the user to supply a routing control parameter list (RCPL) and a message block. The RCPL contains the origin and destination of the message, and the message block contains the data stream as it should be sent to the destination. The ROUTC macro code performs the following:

- Checks the validity of the RCPL and message block format.
- Determines if the destination of the message is one of the following:
  - Application
  - LU
  - Terminal.
- Starts the appropriate routine to send the message.

The message is queued on DASD before transmission if:

- The destination is a hardcopy device.
- Safestoring is required to allow retransmission if an error occurs.
- Output is suspended while waiting for a pacing response.

Queued messages are sent when the transmission media is available. These additional functions are provided for destinations that require queuing the message:

- The operator can request retransmission of the last queued message. For example, retransmission to a printer may be requested when the wrong form was loaded.
- The operator may direct messages to another printer if the original destination is inoperable. In this case, messages already queued for the inoperable printer are moved to the new destination queue and subsequent ROUTC requests are automatically sent to the new destination.
- The system automatically retransmits a message if an acknowledgment is not received after 1 minute.



### SNA Data Transfer

The 37x5 communications controller is a programmable control unit, a piece of hardware. It assumes many of the line handling and processing functions of the network. The network control program (NCP) regulates operation of the 37x5. Macro instructions, coded on input cards, define the network attached to the 37x5. The macro instructions are assembled and the resulting object deck is stored on file. This load module must then be loaded to the 37x5 from an MVS/VTAM system.

### Common Characteristics of NCP and CTC Data Transfer

The standard SNA data unit is called a path information unit (PIU). To transmit data between TPF and the NCP, data must be in this format. The PIU contains routing information in addition to the data itself. PIUs follow in a certain order, based on their virtual route sequence number. PIUs with a virtual route sequence number that is higher or lower than the next expected virtual route sequence number are discarded. TPF builds a PIU for each message transferred to the NCP. TPF also executes the appropriate channel program to transfer the PIUs to the NCP.

### NCP Data Transfer

The NCP is responsible for successful delivery of each message. The NCP presents an attention interrupt over the channel to TPF indicating that data is ready for transfer. TPF reads input data on a time-controlled basis. Therefore, it does not acknowledge every interrupt. Time-controlled reading of NCP input data prevents a single NCP from monopolizing the resources of the TPF system. Control of the network attached to an NCP occurs at two levels: TPF directs control of the network at channel speed, while the NCP controls line handling and data transfer.

### Unique Characteristics of CTC Data Transfer

Data transfer on a SNA CTC connection consists of a matching set of CCWs (write/read on one side and read/write on the other side) with a number of 4K indirect data address word (IDAW) buffers. Either side can initiate a write by beginning the channel program with a write control (WCTL) CCW. The other side receives an attention interrupt from the CTC hardware (for example, 3088); in return, the other side issues a channel program beginning with a sense command byte (SCB) CCW. The read/write sequence of the channel program is fixed at exchange identification (XID) time.

As part of the data transfer, an 8-byte control field is passed by each side as the first 8 bytes of data. Control information, including byte counts and status (such as XID, error retry, and slowdown) is used by each side to determine how to process the data received.

TPF supports a maximum buffer size of 64K. Blocking is used to transmit or receive multiple PIUs on a single I/O operation. TPF initializes I/O operations on the CTC connection when any of the following conditions occur:

- There are enough PIUs to fill the write buffer.
- The SNA polling time interval elapses and output PIUs are queued. The SNA polling time interval can be from 10 milliseconds to 50 milliseconds and is defined with the SNAPOLL parameter on the SNAKEY macro.
- An attention interrupt has been received.

**Note:** You should be aware of these conditions when specifying the value for the VTAM DELAY operand.

---

## Multiple-Domain Networks

A multiple-domain network is a cohesive, coordinated network composed of 2 or more interconnected domains. Networking is a term used to describe communications among different domains. Single-domain networks can support many concurrent applications via shared resources. Although 2 or more single-domain networks can coexist, even using the same CPU and communications controller, the networks are independent of each other. They lack the ability to communicate with each other. Communication between domains is only possible in multiple-domain networks. This type of communication is referred to as cross-domain communication. TPF supports communication with other domains. These include:

- A second TPF domain
- An ACF/Virtual Telecommunications Access Method (VTAM) domain

Before domains can communicate, a path must be established between the cross-domain resource managers (CDRMs). A path consists of a single NCP attached to 2 hosts, or multiple NCPs each attached to a host or another NCP where the NCPs are connected via SDLC lines. A path can also consist of an SNA CTC connection.

---

## Subarea (PU\_5) Environment

### Cross-Domain Links

A cross-domain link interconnects locally attached NCPs and hosts. Cross-domain links are shown in Figure 3 on page 13. SNA CTC support requires the channel-attached major node to reside in the same network as VTAM, with no SNI support.

**Note:** In communications terms, local or locally attached refers to direct channel attachment. Remote or remotely attached refers to a link (communications line) attachment.



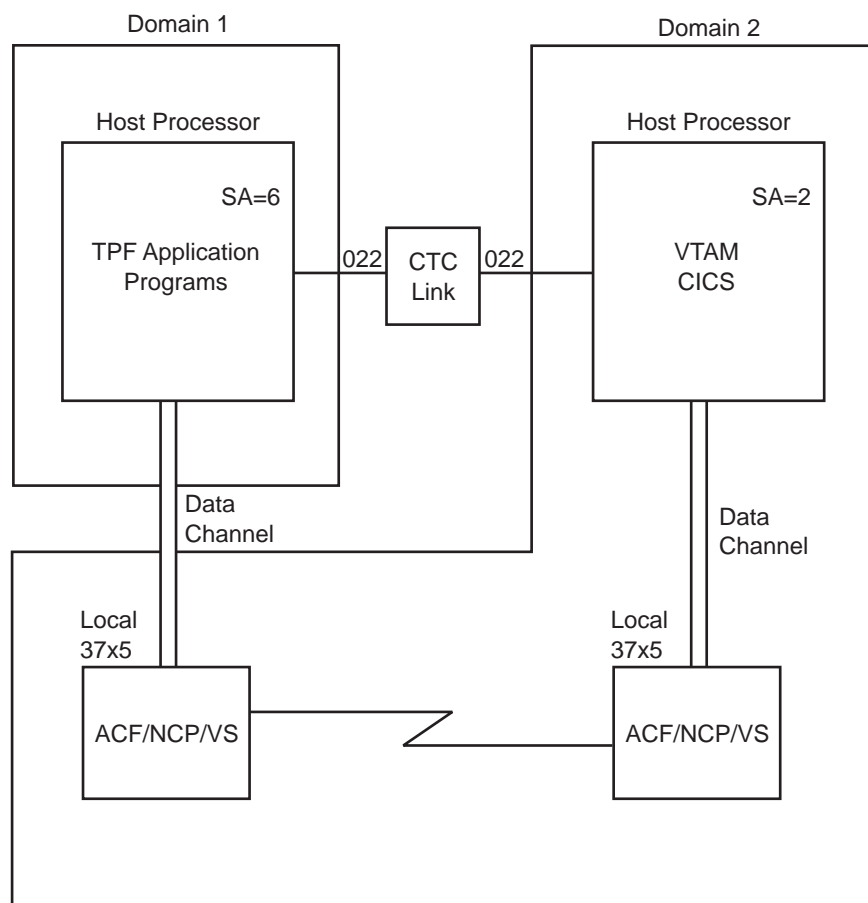


Figure 3. Cross-Domain Communication Link

Cross-domain links present an identical appearance to each domain. Both domains share equally in enabling a cross-domain link. Multiple links provide additional “bandwidth” among domains. Therefore, all interdomain communication does not depend on 1 link. Figure 4 represents a network with multiple links.

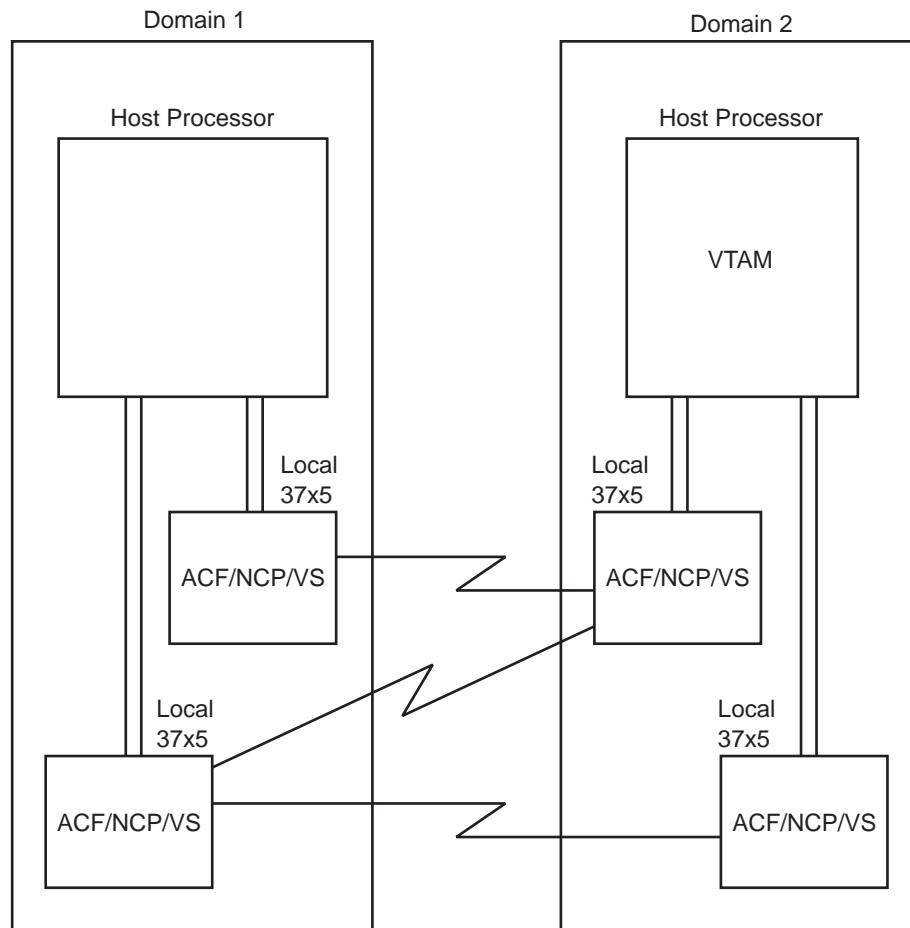


Figure 4. Multiple Cross-Domain Links

A cross-domain link must be enabled before cross-domain communication can take place. The network operators in each domain jointly enable the link.

**Note:** The network operators must activate the link before the path can be enabled.

## Cross-Domain Resource Manager (CDRM)

The system services control point (SSCP) manages the resources for a single domain. In this capacity, the SSCP is often called a domain resource manager (DRM). The DRM activates the network and controls session startup and termination. In a multiple-domain network, SSCP (DRM) functions are extended to include management of cross-domain communication. A cross-domain resource manager (CDRM) to CDRM session must be established before data can be transmitted between domains. In this type of session, the CDRMs work in tandem to activate, maintain, and terminate sessions between LUs in their respective domains.

## Cross-Domain Sessions

A cross-domain LU-LU session must be established before a TPF logical unit can communicate with a second domain's logical unit. Only after the SSCP to physical unit and SSCP to logical unit sessions are established in the respective single domains can a cross-domain session be initiated. Requests for a cross-domain LU-LU session also require that a session first be established between the SSCPs in the 2 domains. The processing to request a cross-domain session is outlined

below. To aid in understanding the processing paths, LUs and SSCPs are numbered. The numbers have no additional significance.

- Logical unit 1 (LU1) sends a control message to its SSCP (SSCP1) requesting a session with the host LU in domain 3. The format of the control message depends on the type of LU. For example, a 3601 LU (application) sends an SNA initiate-self command; a 3270 terminal operator enters a USS (unformatted system services) logon.
- SSCP1 interprets the control message and converses with the CDRM (SSCP3) owning the host logical unit in domain 3. SSCP3 either accepts or rejects the SSCP1 session initiation request.
- If the logical unit is available, SSCP1 and SSCP3 will cooperate to establish an LU-LU session. Once the session is established, data may be exchanged between the LU end points.
- If the host LU in domain 3 is not available (not active or its session limit is exceeded), SSCP3 rejects the session initiation request. SSCP1 informs LU1 of the request rejection.

Figure 5 shows the data flow paths for a cross-domain LU-LU session.

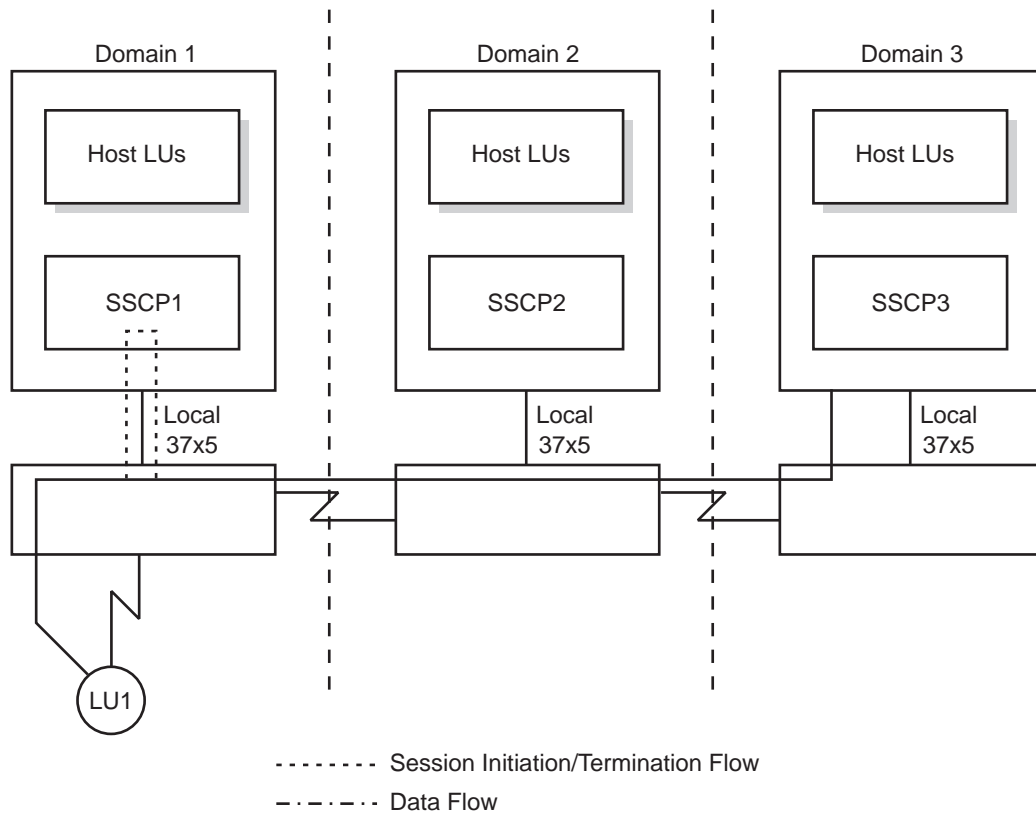


Figure 5. Cross-Domain LU to LU Session Data Flow

## Low-Entry Networking (PUT 2.1) Environment

See *IBM Systems Network Architecture Advanced Peer-to-Peer Networking Architecture Reference* for additional information about the PU 2.1 environment.

## Control Points

With PU 2.1 support, the SSCP is replaced by a control point (CP). The control point serves as a means of identifying the node externally. The CP name is used in qualifying communications with other nodes in the network.

## Control LUs

Control LUs (CLUs) are the mechanisms used to establish LU-LU sessions in the PU 2.1 LEN environment. A VTAM application program called the logon manager (LM) appears to the TPF system as a CLU and has a CLU-CLU session with a TPF CLU over which session initiation requests are passed. Any terminal that is not at the level where it can send a BIND, must request an LU-LU session from its control point. For TPF applications, the VTAM control point views all TPF applications as running under the control of the logon manager. Therefore, it forwards all requests (CINITs) to the logon manager to be serviced. The logon manager includes the CINIT in a request sent on the CLU-CLU session to the TPF CLU that contains the TPF application. TPF then generates and sends the BIND request to the originating LU to establish the session.

When the TPF system is operating in LEN mode, the TPF CP name is different for each link connected to the TPF system. This allows a unique CLU to be defined for each link that is part of the logon manager design.

## 3174 APPN Considerations

When TPF is connected to a 3174 Advanced Peer-to-Peer Networking (APPN) controller, there is no VTAM system involved. Without a CLU-CLU session to transport session initiation requests, TPF has no way of sending a BIND request through the 3174 to the PS/2 computer, PS/2 devices, PS/2 units on the token ring. The PS/2 computer, PS/2 devices, PS/2 units must be independent LUs, capable of sending a BIND request to TPF. Only sessions using the LU 6.2 protocol are supported for a 3174 APPN connection.

---

## Advanced Peer-to-Peer Networking (APPN) Environment

See *IBM Systems Network Architecture Advanced Peer-to-Peer Networking Architecture Reference* for additional information about the PU 2.1 environment.

## Control Points

With PU 2.1 support, the SSCP is replaced by a control point (CP). When the TPF system is operating in APPN mode, there is one CP name for the entire TPF loosely coupled complex. One TPF processor has CP-CP sessions with an adjacent APPN network node (NN) and manages all the resources for the complex.

CP-CP sessions are a pair of LU 6.2 sessions between adjacent CPs. LU 6.2 support must be installed for the TPF system to be able to connect to the network as an APPN node. See "TPF/APPC Installation Checklist" on page 60 for information about installing LU 6.2 support on the TPF system.

---

## Message Routing

The function management message router (FMMR) transmits messages from terminals or logical units attached to 1 processor to applications in a second processor. Here, the communication path is always directed through the first processor. This type of message routing is called *level 2* networking.

In a TPF SNA environment, FMMR is only required for non-SNA terminals. However, FMMR can be extended to make use of the cross-domain SNA communication paths (SNA logical units communicate directly with applications in other processors).

The FMMR operates as a host node LU in a TPF system. Therefore, to communicate with the FMMR in another processor, a formal LU-LU session must be established. Message recovery is not provided in an FMMR to FMMR session. Users who require message recovery should communicate with other domains via TPF host node secondary LU support.



---

## TPF Applications

In TPF terminology, an application is defined as a collection of programs is a software package. When a TPF/SNA network is defined, a logical unit is associated with each application. Each LU has a unique network address and network name. TPF requires a node control block (NCB) to control the functions of each LU.

You must define all TPF applications, which are also known as local applications, using the offline ACF/SNA table generation (OSTG) program. The TPF system associates a scratchpad area (SPA) control record with each application that is defined using the OSTG program. This SPA is for the exclusive use of the application.

Non-SNA devices require different control records: the routing control block (RCB) and the agent assembly area (AAA). TPF provides an interface program that converts non-SNA address formats for applications written before the advent of SNA. Applications that are SNA-oriented are called NCB/SPA dependent. Applications that are not SNA-oriented are called RCB/AAA dependent. The discussion throughout this publication pertains to NCB/SPA dependent programs unless otherwise stated.

In the network, TPF applications are viewed as primary logical units. However, a user may optionally request that an application also appear as a secondary logical unit. Here, the application assumes a primary LU status when communicating with an SLU; a secondary LU status when communicating with a PLU. As a secondary LU (also known as an SLU THREAD), the communication rate among applications is limited to the communication rate of the session protocol supported. To ease this limitation, TPF allows the user to specify a number of communication paths (threads) when communicating with a primary LU. Each path (thread) is assigned a unique network address and network name. See *TPF ACF/SNA Network Generation* for information describing the ANT deck and application SLU threads.

Another way of overcoming this limitation is to use parallel sessions provided with TPF/APPC. See "TPF Advanced Program-to-Program Communications" on page 59 for additional information about TPF/APPC.

---

## Considerations for Developing Applications That Access NCB Records

When developing your own applications, you must use the CSNB segment to access NCB records. This segment returns the address of the appropriate 381-byte fixed file NCB record or 381-byte long-term pool file NCB record for an LU, or it can return the NCB record itself.

See "Retrieving NCB and SPA Data Records (CSNB)" on page 287 for more information about the CSNB segment.

---

## TPF as Host Node SLU

TPF applications may appear as host node secondary LUs to be compatible with certain types of IMS and CICS logical units. With this support, users can write applications to communicate with CICS, IMS, and/or a second TPF system via a standard SNA interface program. CICS, IMS, and TPF data formatting requirements

are resolved in the user's applications. TPF transmits the data, processes errors, and provides the SNA interface segments. TPF host node SLU support allows you to perform:

<b>Relay routing</b>	Non-SNA terminals, in a TPF domain, access an IMS or CICS database or both.
<b>Transaction routing</b>	A TPF application dynamically determines which input messages should be forwarded to another system for processing.
<b>Data acquisition</b>	The ability to request data from an IMS or CICS database or both. This data is usually needed to respond to a request from a logical unit in the TPF domain.

---

## Activating LU–LU Sessions with TPF

An LU-LU session must be established before data can pass between a TPF application and a logical unit. The following list different ways to establish a session:

1. A secondary LU is defined as permanently in session with a TPF application (VTAM AUTOLOG facility).
2. A remote LU requests a session with a TPF application from its control point.
3. The TPF operator requests a session be established using the ZNETW ACT command. (This is valid only for APPL to APPL sessions.)
4. A TPF/APPC session is activated with an ALLOCATE request. See "TPF Advanced Program-to-Program Communications" on page 59 for more information about TPF/APPC.



---

## TPF Application Programming Considerations

**Note:** This chapter is not applicable for TPF/APPC. See “TPF Advanced Program-to-Program Communications” on page 59 for this information.

TPF provides the user with a standard interface for communicating with TPF applications. The input interface consists of:

- The routing control parameter list (RCPL)
- A main storage block attached to the entry control block (ECB).

The RCPL contains the origin and destination for a message. It is located in the ECB work area. Indicators in the RCPL delineate the type of input device, the environment (SNA or non-SNA), and the presence of a resubmitted message. When an application responds to a message, the origin field becomes the destination and the destination field becomes the origin. The main storage block contains the input message.

TPF assembles a segmented message before presenting it to the application. Some messages do not fit into a single main storage block. These are stored on file and forward-chained to the block. Applications use the ROUTC macro to send a message to an SNA logical unit. If output messages use more than 1 block, they too are chained together on file.

**Note:** In a TPF system, a reply is sent for every input message. Input messages are resubmitted to an application (TPF records every input message) if a reply is not received. Users define the time-out period during network definition. Applications **must** release the input message, indicating processing is complete, or the message is resubmitted.

When an application is started to process an input message, it receives:

- The RCPL, starting at field EBW000 in the ECB work area. The RCPL format is specified during system initialization. See *TPF System Generation* for more information on the system initialization program (SIP).
- The message is attached to level 0 of the ECB.

The RCPL contains the following fields:

- Origin
- Destination
- Indicators.

The origin field of an RCPL contains:

- An ordinal number that identifies the logical unit that sent the message. This sub-field is referred to as the resource identifier (RID) or session control block (SCB) identifier.
  - With the ALC support under SNA, the LN/IA/TA is now the Logical End-Point Identifier (LEID).
- A 1 character CPU identifier. The logical unit is directly attached to the named CPU.

Non-SNA application programs use a line number, interchange address, and terminal address (LN/IA/TA). The LN/IA/TA replaces the RID.

The destination field contains the 4-character name of the application that receives the input message. Before sending a message, the logical unit must **log on** to this application. Additionally, the application must be included in the routing control application table (RCAT).

The RCPL contains indicator fields that describe the input or output message. RCPL fields are defined via the RC0PL data macro. Application programs should set any unused RCPL indicators to zero. This practice prevents errors and eases installation of new applications.

**Note:** SNA output messages that cannot be sent because of telecommunication failures are also passed to the application as input. In this situation the returned message indicator (RCPL0RET) is set. Applications should be designed with the capability of processing returned output messages.

The original messages sent by the application may have been altered by the ROUTC exit or a process selection vector (PSV) routine. Therefore, when using these facilities, regardless of the Data Flow Control (DFC) layer implementation or the use of PSV routines, TPF **must** return messages to the originator (application/PSV) via the same path (ROUTC Exit/PSV) rather than directly to the application via the RCAT enter expansion.

See “Process Selection Vector (PSV)” on page 253 and “Data Flow Control (DFC) Considerations” on page 260 for additional information. Also, see the user exit information in the *TPF System Installation Support Reference*.

---

## Function Management Header (FMH)

Both host and remote applications can include an FMH when transmitting a message. An FMH contains information about the text of the message. For example, an FMH might be used to indicate the results of some host processing. The FMH contains 2 fields: the header length (HL) field and the control information (CI) field. The HL field is 1 byte long. The CI field is 1 – 255 bytes long. Content and usage for the CI field is determined by the user when designing the host and remote application programs. When included, the FMH is the first part of the message text. The presence of an FMH is indicated in the RCPL. When the FMH length field (HL) is set to zero, the output message is not sent. This procedure is sometimes used to comply with TPFs requirement that all input messages receive a reply. Only the first segment in a multiple segment input message can include an FMH; the remaining segments must not include one. Messages that include an FMH are often referred to as **data and control** messages.

---

## Programming Considerations for Session Initiation

When a logical unit requests a session, the following sequence occurs:

- The logical unit sends an SNA Initiate-Self command to its control point to request a session with the application.
- The remote control point communicates with the TPF control point to request the session.
- TPF returns a positive response, if requested.
- TPF sends an SNA BIND command built from the CDINIT to initiate the session.
- The logical unit returns a positive response. At this point, when necessary, message synchronization is performed.

- TPF sends an SNA Start Data Traffic (SDT) command, except for an LU 6.2 session.

At this point, data may now flow between LU session partners.

## Synchronizing Messages with Sequence Numbers

A sequence number is an arbitrarily assigned message number. Both TPF and the 3601 control program assign sequence numbers to output messages. Additionally, each maintains a record of these numbers for both input and output messages. Messages are synchronized via sequence numbers to ensure their integrity. TPF and the 3601 control program share responsibility for sequence number management.

The 3601 adds 1 to the sequence number each time a logical unit transmits data. The 3601 appends this new number to the data and sends the message. On receipt of the message, TPF adds 1 to the last sequence number it recorded. TPF compares this number with the number of the message just received. If the sequence numbers match, TPF processes the message. Otherwise, TPF returns a negative response indicating a sequence number error. The 3601 performs the same function for messages received from TPF. When TPF detects the error, the logical unit should request resynchronization of messages via the SNA Request-Recovery (RQR) command. When the 3601 detects the error, TPF automatically starts message resynchronization via the SNA Set-and-Test Sequence Numbers (STSN) command.

The STSN command contains a 5-byte data field. Byte zero (1) is called the action code and is defined in Figure 6.

Bytes 1 and 2 contain the SNA sequence number of the last message received by

Byte	The STSN Action Code Byte		
	Bits	Is set to	To
0	0	00	(TPF does not use - do not alter)
	&		
	1	01	Set the appropriate sequence number to the value in the corresponding sequence number field
	2	00	(TPF does not use - do not alter)
	&		
	3	01	Set the appropriate sequence number to the value in the corresponding sequence number field and test its validity

Figure 6. TPF Sent Set-and-Test Sequence Numbers Action Code

TPF. Bytes 3 and 4 contain the sequence number for the last message TPF sent.

If the sequence numbers are correct, the 3601 simply returns a positive response. Otherwise, the response also includes a 5-byte data field. Figure 7 on page 24 defines the 3601 action code (byte 0) response.

Byte	The 3601 Action Code Response		
0	Bits	Set to	Indicates
	0	00	The sequence number TPF sent in the STSN command is not acceptable. The number should be set to its previous value.
	&		
	1	01	The sequence number is acceptable.
	2	10	The 3601 has detected a sequence number error.
	&		
	3	11	The 3601 does not agree with the sequence number presented via the SET and TEST option.

Figure 7. 3601 Set-and-Test Sequence Numbers Action Code Response

The 3601 places the sequence number of the last message it sent to TPF in bytes 1 and 2, and the number of the last message it received in bytes 3 and 4.

The 3601 response to a TPF STSN command is summarized in Figure 8. The ensuing TPF processing options are summarized in Figure 9 on page 25.

Action Code (Byte 0)		Host input sequence number (Byte 1-2)	Host output sequence number (Byte 3-4)	3601 action when received
Host input #	Host output #	Last message received by TPF		Set SMSCWS field to value indicated in host input field of STSN
01 Set				
	01 Set		Last message sent by TPF and responded to by the logical unit	Set SMSCWS field to value indicated in host output field of STSN
	11 Set and test		Last message sent by TPF; no response from the logical unit received	

Figure 8. 3601 STSN Response

If action code was		Response action code must be *		TPF Action when response received
Host input #	Host output #	Host input #	Host output #	
01 Set		01 Test Positive		Send SDT to logical unit
	01 Set		01 Test Positive	Send SDT to logical unit
	11 Set and Test		01 Test Positive	Discard all messages previously sent and send SDT to logical unit
			11 Test Negative	Send SDT to logical unit and retransmit all unreceived messages

\* TPF terminates the session for any other response

Figure 9. TPF Action Resulting from 3601 STSN Response

## 3600 Application Considerations

### Data Transmission from 3600 to a TPF Application

Data is sent to TPF via the LWRITE instruction. 3601 applications must set indicators in the write-type (SMSCWT) and write-flags (SMSCWF) fields before issuing the LWRITE instruction. The write-type field indicates whether a function management header (FMH) is included with the data. The write-flags field indicates:

- The requested response type
- The position of the message in the message chain
- If brackets are present
- The presence of a change direction request.

Logical units must request *exception response* for all data sent to TPF. All other responses result in an error. See “SNA Message Protocol” on page 247 for an explanation of the various SNA response types.

The format of input data sent to a host application is not fixed; the designers of the 3601 and host applications should establish a format suitable to the needs of the installation.

### Data Transmission from a TPF Application to a 3600

TPF output messages (those messages TPF transmits to a logical unit) are classified as: 1) application messages, or 2) system messages. These messages are further defined as recoverable or non-recoverable (See “Recoverable and Non-Recoverable Messages” on page 247).

**Note:** 3600 multithread devices cannot be defined as recoverable.

Most output messages are sent in reply to an input message. Some, however, are not associated with an input message. These messages are referred to as unsolicited messages (see “Unsolicited Messages” on page 250). TPF prefixes all system messages with a function management header (FMH). The user defines the contents of the FMH during system initialization (SIP).

TPF output messages are sent in a single transmission. They are segmented only when: 1) the host application requests segmented output, or 2) the logical units' input buffer is too small to contain the entire message.

Data is received from TPF via the LREAD instruction. Then, the read-type field (SMSCRT) and the read-flags field (SMSCRF) are examined. A *definite response* is requested for single-segment recoverable messages. An *exception response* is requested for non-recoverable messages.

Chained messages request exception response for the first and middle messages in the chain. For recoverable chained messages, the last message in the chain requests a definite response. For non-recoverable chained messages, the last message in the chain requests an exception response.

---

## 3270 Application Considerations

### Non-SNA 3270 Application Considerations

- Non-SNA applications (RCB/AAA dependent) can use synchronous data link (SDLC) devices without being changed. An LEID is defined for each 3270 SDLC device in the offline ACF/SNA table generation process.
- Devices attached to 3x74/3276 cluster controllers have different terminal-type values than those attached to 3271 control units. See *TPF General Macros* for information on the TRMEQ macro. Terminal-type values are generated with this macro. However, issuing a WRITE with the START PRINT bit set to 1 simulates the COPY function.

### SNA 3270 Application Considerations

TPF and the devices that interact with it use the following SNA protocols:

- For terminals attached to 3271 control units:
  - Brackets are not used.
  - Change direction indication is not required.
  - Function management headers are not required.
- For terminals attached to 3x74/3276 control units:
  - Brackets are used.
  - Change direction indication is required.
  - Function management headers are not required.

**Note:** It is recommended that you use the 3x74/3276 protocol for applications that interact with both 3271 and 3x74/3276 units. TPF ignores end bracket and change direction settings in output messages sent to 3271 attached devices.

---

## Host Node Application Considerations

TPFs' host node support allows TPF applications to communicate with IMS and CICS applications. One of the uses of this support is to enable SNA or non-SNA (EP) terminals to access IMS or CICS databases. This is performed by writing a TPF application that forwards terminal requests to, IMS for example. When IMS

subsequently replies, the TPF application will forward the reply to the originating terminal. Normally, the TPF application reformats the data stream from the terminal to one acceptable to IMS and conversely reformats the IBM reply to a form appropriate to the terminal.

Because TPFs host node support requires the message recovery package, a key consideration for writing TPF host node applications is the operation of the system recovery table (SRT). The SRT is used to track each message from its receipt by TPF until the corresponding reply is successfully returned. For example, assume a TPF application that will forward all terminal input to IMS and subsequently return all IMS replies to the terminal.

In this example, an SRT entry will be made to record the receipt of each input message from the terminal. In addition, if the terminal and TPF application were defined as recoverable, a safe store copy of the input will be created in pool records on DASD. The SRT entry is active until the TPF application signals a Release Input to indicate that the entry was successfully processed. To signal a Release Input, the TPF application must issue a ROUTC macro with the RCPL2REL bit set in the RCPLCTL2 field of the routing control parameter list (RCPL). If the TPF application fails to signal Release Input in a user-defined time limit, TPF will consider the original input lost and resubmit a copy of the message to the application.

Therefore, a TPF application relaying messages between a terminal and IMS has several choices of when to signal Release Input. For example, the application can do any of the following:

- Release Input can be signaled after the reply from IMS is received. In this case the terminal keyboard is locked from the time it enters the message until it receives the IMS reply. If IMS is slow to reply or fails, TPF will time out the original input and resubmit it to the TPF application. The application has to decide to either (1) release the original input and unlock the terminal's keyboard or (2) wait for the reply from IMS. In general, three cases exist relative to the reply from IMS:
  1. The reply from IMS was delayed and will be received.
  2. The session failed after the message was sent, or the message was sent with the return message indicator RCPL0RET on. In this case the message will be (eventually) processed by IMS and a reply returned.
  3. The message was sent with the return message indicator RCPL0RET and the session failed before the message was actually transmitted. In this case the message will be returned to the TPF application as not deliverable.
- The application can signal Release Input before forwarding the message. In this case TPF will consider the terminal's input as processed to completion. This means that the application must implement its own technique for correlating the IMS reply with the requesting terminal and handling possible returned messages or delayed IMS replies.

The reader should understand that the decision to either unlock or leave locked the terminal's keyboard is left to the application. Leaving the keyboard locked until the IMS reply is delivered prevents the terminal from sending another message, but requires the application to unlock the terminal if the IMS reply is not returned. Conversely, unlocking the keyboard before sending the IMS reply can make the correlation of the IMS reply to the original input difficult.

When the reply is received from IMS, the TPF application must indicate receipt of the message by sending a *null RU* via a ROUTC to IMS. This will cause the system



to delete the SRT slot for the message from IMS (which is recoverable) and send a definite response to IMS. The TPF application must also send a message to the originating terminal. Because the TPF application will EXIT after sending the initial message to IMS and then be reactivated on the receipt of the reply message from IMS, it must **remember** the originating terminal. This can be done upon agreement of the interface between TPF and the IMS application that the RCPL be passed as part of the data stream and returned intact.

The null RU can also be used when an input message is resubmitted to the application because of a time-out when no reply is received to a recoverable input message. The application, if it decides that no more processing is required, can simply cause the SRT slot to be released and a response sent to the terminal by sending a null RU. The null RU in this case and the response to IMS described previously consists of an RCPL with the Release Input present and Function Management Header set, and the first byte of the message (AM0TXT) set to a X'00'.

The previous discussion assumes that both the TPF application and IMS have been defined to TPF as recoverable LUs so that message transmission between TPF and IMS is to be guaranteed. If either IMS or the TPF application were defined as non-recoverable, TPF will bypass the system overhead of ensuring messages are either successfully delivered or returned to the sender. Message recovery considerations (such as null RUs and SRT slots) still apply. The only difference is that TPF will not safe store messages on disk and will not request definite response for output messages.

RCPL control byte 2 (RCPLCTL2) also allows the application programmer to control transmission protocol on the Host Node LU session. If both session partners are defined as being recoverable, a half duplex flip-flop protocol is used by indicating Change Direction in the RCPL (RCPL2CD). If the SLU threads that are to be used for the Host Node LU session are defined as non-recoverable, a contention mode protocol may be used by indicating **End Brackets** in the RCPL (RCPL2EBK). Defining the SLU threads as non-recoverable allows the SLU to send **End Brackets**, which is necessary for full contention mode support. **Change Direction and End Bracket** are mutually exclusive and if both are indicated, **End Bracket** will be assumed. If **End Bracket** is indicated and the SLU threads are defined as being recoverable, half-duplex protocols are enforced.

In the previous examples, IMS was used, but the discussion is also applicable to CICS.

## IMS Relay Application Considerations

TPF's support of IMS uses the type P SLU support present in IMS. The TPF application appears to IMS as a type P SLU. As such, all IMS conventions for this session type must be adhered to by TPF.

One such convention used by IMS is sending an unsolicited IMS error message (DFSxxxx) immediately following an IMS exception DR1/DR2 to input (-RSP). For example: IMS will send a -RSP to input with system/user sense code 08260041 if the IMS transaction is stopped. IMS expects to remain in send state regardless of the current bracket state. IMS will then send the complete error message: DFS0065 TRAN/LTERM STOPPED (the hex '41' becomes the decimal 65 of the error message). TPF system code will stop output message transmission (OMT) on receiving the -RSP and pass both the -RSP and the following IMS error message to



the TPF application. The TPF application must ROUTC a null RU to notify TPF to restart OMT. The null RU for this case is defined as:

1. Unsolicited-msg indicator set on (RCPL0MTY)
2. FMH indicator set on (RCPL2FMO)
3. Null RU (RCPLDLEN = zero)
4. No SRT (RCPLDESS = zero).

TPF will function in this manner for system sense codes 0800, 0819, and 0826.

## CICS Relay Application Considerations

TPF support of CICS uses the 3790 Full Function LU support in CICS. The TPF application appears to CICS as a 3790. As such, all CICS conventions for this session type must be adhered to by TPF.

---

## X.25 NPSI Support

TPF supports connections to packet switched data networks (PSDNs) through the use of the NCP Packet Switching Interface (NPSI).

SNA devices that attach to TPF through NPSI are not defined as NPSI device types; instead, they are defined using other supported TPF RSC device types.

Defining a network containing X.25 resources using NPSI is described in the NPSI publications for the single-domain case. This task requires understanding of VTAM and NCP as well as NPSI and X.25, and consists of many considerations not relevant to TPF X.25 support or even a TPF application using the X.25 network. Such topics, mostly concerning X.25 link and physical level protocols, or X.25 vendor options, are not covered here because they have no influence on the TPF X.25 support, or on the definition of NPSI LUs to TPF.

TPF provides connectivity support for the following NPSI facilities defined by Logical Link Control (LLC) type:

- |                |  |
|----------------|--|
| <b>LLC=0 :</b> | Both Switched Virtual Circuits (SVCs) and Permanent Virtual Circuits (PVCs)  |
| <b>LLC=3 :</b> | Both Intermediate Network Node (INN) and Boundary Network Node (BNN) function forms  |
| <b>LLC=4 :</b> | General Access to X.25 Transport Extension (GATE) option with integrated Fast Connect support  |
| <b>LLC=5 :</b> | Packet Assembler/Disassembler (PAD) Options: <ul style="list-style-type: none"><li>• Integrated PAD</li><li>• Transparent PAD.</li></ul> |

TPF uses the ROUTC interface for TPF applications to communicate with logical end points reached through a PSDN. This is achieved by extending TPF SNA support to the LU\_T1 interface with the NPSI product in a local or remote NCP node. Additionally, user-written system modules, which collectively form the basis of a user Communication and Transmission Control Program (CTCP), provide the user with the capabilities offered by NPSI GATE support. Process selection vector (PSV) routines can be used for this purpose. See "Process Selection Vector (PSV)" on page 253 for additional information.

The NPSI link, PU, and LU resources are owned by VTAM. TPF needs network awareness of only the NPSI LU resources. Therefore, all LU sessions with NPSI

LUs, or in the case of LLC=3 with the LU residing across the PSDN, have cross-domain/network session characteristics.

Non-SNA devices are supported as follows:

- On Permanent Virtual Circuits: using NPSI LLC=0 or LLC=5.
- On Switched Virtual Circuits: using NPSI LLC=4(GATE) with the NPSI Fast Connect support **and** NPSI LLC=0 or LLC=5 for incoming calls without Fast Connect using switched SNA device protocols.

Non-SNA devices appear to TPF as LU\_T1 LU\_LU sessions and use the following session characteristics:

- Half-duplex contention
- No brackets
- Exception response only.

SNA devices are supported using NPSI LLC=3.

The following functions are not supported:

- TPF SNA Message Recovery for sessions with NPSI LUs
- X.25 D-bit delivery confirmation.

## NPSI Message Length Considerations

An “X.25 message” is a sequence of one or more X.25 data packets ending with a packet with the “more data” bit in its packet header turned off (indicating no more data). During a RECEIVE, for example, NPSI combines related data packets in one X.25 message and places it in 1 inbound PIU for application processing. During a SEND, an outbound PIU contains 1 X.25 message that NPSI transforms into 1 or more X.25 data packets for network transmission.

TPF applications, therefore, send or receive X.25 messages only. Furthermore, TPFs message length support allows up to a 4K block maximum. Because TPF message blocks contain headers and control bytes, user applications must adhere to the following data length constraints:

- An inbound X.25 message must be less than 4027 bytes of user data.  
Input messages will continue to be given to a TPF application in the smallest core block possible. However, a complete X.25 message must be placed in 1 core block. If an inbound message exceeds 1020 bytes of user data (size of text area in a 1055-byte core block), that message is placed in a 4K block for the application.
- An outbound X.25 message must be less than 4027 bytes of user data.  
Each use of the ROUTC macro sends 1 and only 1 X.25 message. For resources accessed through NSPI, OMT creates a single SNA PIU with an RU maximum length of 4027 bytes. Messages longer than 4027 bytes are discarded and a system error is taken. An application may present outbound messages using the existing interfaces (127, 381 byte core blocks or file chained 381 pool records) or as a single 1055-byte or 4K block.

## NPSI LU-LU Session Characteristics

The NPSI session protocols that are used by TPF X.25 support are:

- Half-duplex contention.
- No brackets.
- Multiple RU chains allowed both in and out but only single RU chains used.
- Exception response only.
- Immediate request mode.

- FMHs are not used.
- 4096 Max RU receive size (largest inbound X.25 message).
- 4096 Max RU send size (largest outbound X.25 message).

These options are given in the BIND image detailed in *TPF ACF/SNA Network Generation*.

Because different LLC types use different message formats, the TPF application must be aware of the LLC type used on each NPSI session. This information is not maintained by TPF X.25 support and must be supplied and tracked by the-user in user defined tables. (See the user exits information in *TPF System Installation Support Reference*.)

The SNA commands shown in the Table 1 are used for the TPF NPSI support:

Table 1. SNA Commands Supported for NPSI

Session Control	Data Flow Control
BIND CLEAR SDT UNBIND	CHASE LU_STAT RSHUT RSHUTD SHUTC SHUTD SIG (LLC=5 Integrated PAD)

## Request Unit Chaining

TPFs support of the NPSI request unit (RU) chaining allows messages of up to 32K in length to be transmitted and received through an X.25 network. Input arriving in 4K blocks is deblocked into a chain of 1055-byte blocks to insulate the application from the usage of 4K blocks. Messages chained in 4K blocks can be queued and dequeued. The NPSI RU chaining function allows chaining of 1055-byte blocks and 4K blocks to virtual circuits. This RU chaining function is optional on a multi-channel link basis and is specified in the NPSI generation by using the MBITCHN=YES/NO keyword. You can use the CHAIN parameter in OSTG to specify the NPSI resources using the RU chaining function. (For additional information on the CHAIN parameter, see the *TPF ACF/SNA Network Generation*.)

**Note:** RU chaining is only supported by TPF resource types VC and MCH. All other TPF resource types that represent NPSI devices do not support RU chaining. In addition, not all NPSI LLC types support the MBITCHN=YES keyword. See the *NPSI Planning and Installation* for complete details of supported LLC types.

There is a restriction of chained 381-byte or single 1055-byte or 4K blocks for NPSI resources that OSTG has defined as not supporting RU chaining.

## RU Sizes

Support of the NPSI RU chaining function also affects TPF's management of maximum RU sizes. Maximum RU sizes, or maximum input buffer sizes, are negotiated at session initiation time. The maximum RU size dictates the size of PIU segments in an RU chained message.

## Bind Command Processing and Input Buffer Size

Bind processing uses the SESINIT and CDCINIT values to fill RV1MAXIP in the RVT. These values are used by bind processing to specify TPF's maximum input buffer size. This value is the least of:

- The value in the bind image table for the resource
- The value in CTK2 that specifies TPF's maximum physical input buffer size (the field filled in by SNA Restart)
- The value in the SESINIT/CDCINIT.

To fill in RV1MAXDT, the remote resources maximum input buffer size, bind processing uses the SESINIT and CDCINIT values. This value is less than:

1. The value in the bind image table for the resource.
2. The value in the SESINIT/CDCINIT

OMT creates PIU segments equal to the remote resources maximum RU size (RV1MAXDT) whenever possible. Because of core block size restrictions, values greater than or less than 4027 bytes may not be honored. OMT constructs segments of 4027 bytes when a value greater than this is found. This allows applications that do not have the capability to chain 1055-byte or 4K blocks to take advantage of NPSI's ability to handle 4K PIUs. For instance, if an application issues ROUTC with an output message in chained 381-byte blocks, OMT packs the output into a 4K block and issues SOUTC of the PIU segment. This significantly reduces the number of PIUs flowing in the SNA network.

You can tune the SNA network performance by updating the VTAM LOGMODE table. TPF uses its bind image table as absolute maximum values to prevent you from specifying a value that is too high for TPF.

---

## NPSI GATE/Fast Transaction Processing Interface (GATE/FTPI)

The Fast Transaction Processing Interface (FTPI) facility of the X.25 NCP Packet Switching Interface (NPSI) enhances the GATE facility. (GATE is General Access to X.25 Transport Extension.) GATE allows communications between non-SNA devices and a Communication and Transmission Control Program (CTCP) running under TPF or VTAM. A CTCP assumes control of the X.25 DTE/DCE interface protocol. The communications between a CTCP and the NCP is established through SNA sessions. With FTPI, all virtual circuit traffic is multiplexed on a single SNA session between the CTCP and NPSI.

Without the FTPI option, NPSI requires the traffic for each virtual circuit to flow on its own unique SNA session. By reducing the multiple sessions required for each virtual circuit to a single session for each NCP, the time to activate the network is greatly reduced, and less main storage is required by NPSI. FTPI also improves NCP performance by blocking several packets into a single PIU for transfer to the host. TPF similarly blocks output to the NCP.

The essential parts of FTPI support include:

- Removing input traffic in order for each message to be associated with an ECB rather than associating each block of messages or each PIU with an ECB.
- Multiplexing output traffic for output message traffic to be blocked before transfer to the NCP.
- Defining the network components needed to support FTPI. The definitions affect TPF, VTAM, Logon Manager, NCP, and NPSI.

For more information regarding coding of a GATE/FTPI CTCP, as well as more detailed information regarding NPSI and its command flows, see the *NPSI Host Programming* and the *NPSI Planning and Installation*.

## Message Traffic Multiplexing

The format of the traffic sent between NCP and TPF is shown in Figure 10 on page 33. The data transferred is always complete messages and is prefixed by a command header. The header provides the data length, X.25 link identifier (MCH\_ID), the virtual circuit identifier (VC\_ID), and the correlation number. The correlation number is assigned to the virtual circuit by NPSI at CALL REQUEST time and is used for all control and data exchanges for the life of the virtual circuit. Figure 11 on page 34 shows the data content of each of the FTPI commands.

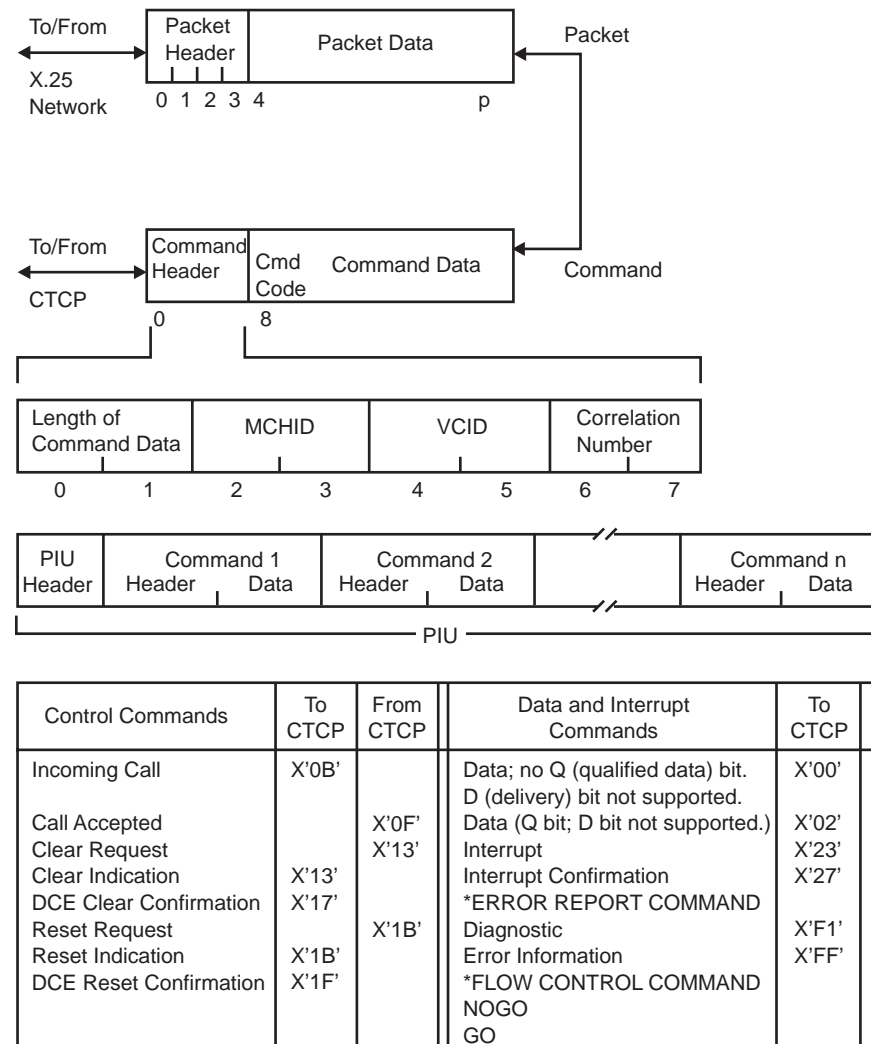


Figure 10. FTPI Message Blocking Format. For additional information on the Q and D bits, see the *NPSI Planning and Installation*.

Command Name	Header	Cmd Code	Command Data								
			Byte 1	2	3	4	5	6	7	8	9
Data	MCH=m VC=n CN=p	00	user data to/from network variable user data . . .								
Data Qualified	MCH=m VC=n CN=p	02	user data to/from network variable user data . . .								
Interrupt	MCH=m VC=n CN=p	23	user data	optional user data							
Interrupt Confirm.	MCH=m VC=n CN=p	27									
Incoming Call (IC)	MCH=m VC=n CN=p	08	vc_id	addr . . . block		from network facility . . . block		cud** . . .			
Call Accepted (CA)	MCH=m VC=n CN=p	0F	vc_id	pkt wndow	packet size		address block, facility block, cud**				
Clear Request (CR)	MCH=m VC=n CN=p	13	vc_id	clear cause		diag code		CR packet (to network) address block, facility block, cud**			
Clear Indicat. (CI)	MCH=m VC=n CN=p	13	vc_id	clear cause		diag code		CI packet (from network) address block, facility block, cud**			
DCE Clear Confirm (CF)	MCH=m VC=n CN=p	17	vc_id	clear cause		diag code		CF packet (from network) address block, facility block, cud**			
Reset Req/Ind. (RRQ/RI)	MCH=m VC=n CN=p	1B	RRQ/I reset cause	pkt diag code	** cud contains billing information (connect time packets sent/rcvd)						
Reset Confirm.	MCH=m VC=n CN=p	1F									
Diag.	MCH=m VC=0 CN=p	F1	code	expanation . . .		Diag. packet					
Error Info.	MCH=m VC=n CN=p	FF	vc_id	code error	error code	Billing info					
NOGO	MCH=0 VC=0 CN=0	F2									
GO	MCH=0 VC=0 CN=0	F3									

Figure 11. FTPI Command Content. As represented in this figure, the X.25 link identifier is MCH, the virtual circuit identifier is VC, and the correlation number is CN. Additional information on the FTPI command content is found in the NPSI Host Programming.

## Input Processing

The input logic is located in OPZERO and provides demultiplexing support for message traffic received over an FTPI session. The flow of input processing follows:

1. Each message in a PIU received from NPSI FTPI is copied to a core block and associated with an ECB.

2. The data collection counters for total message length and number of messages received reflect the number and size of the messages received rather than the number and size of PIUs received.
3. The demultiplexing of messages honors the input shutdown levels. When an FTPI PIU is processed by OPZER0, that is, the message is copied to a core block and the block attached to an ECB, OPZER0 only schedules 1 ECB before returning the PIU to the top of the Input\_List. This technique insures that the Input\_List shutdown levels for ECBs and core blocks continue to control the addition of new work to the system.

## Multiplexing Output Traffic

The logic in the SNA Output (SOUTC) routine accumulates output destined for an FTPI session. The general SOUTC processing recognizes traffic destined for an FTPI LU and:

1. Accumulates output messages using as the control structure the LU message blocking table. Messages are blocked into 4K blocks before transmission. This technique centralizes multiplexing in SOUTC.
2. Sends the accumulated data when the 4K block is full or when a user-definable timer expires.
3. Data collection counters for the number and size of messages sent reflect the actual message count, not the PIU count.

## Application Program Interface

FTPI message traffic is presented to the application using the standard TPF API:

- ECB work area EBW000 contains the routing control parameter list (RCPL).
- ECB core level 0 points to the input message.

Similarly, the application sends output messages by issuing a ROUTC macro with the data in a core block and a register pointing to the RCPL. On input to the application, the message text starts with an FTPI command header as shown in Figure 10 on page 33. Output sent by the application must also prefix the message text with an FTPI command header. Optionally, the PSV routine that handles the call setup and takedown for the X.25 resources supported by the FTPI multiplexed session can insulate the application from processing FTPI headers by also performing the FTPI header management. The PSV routine defined for the FTPI LU can remove the FTPI header from input messages and store for recall and appending to the corresponding output message.

## Network Definition

Define the following to NPSI:

- X.25 links and virtual circuits.
- CTCP pseudo LU (CPLU). The CPLU is the access port of the TPF system to the X.25 links serviced by the NCP. The CPLU is the NPSI LU that performs the GATE/FTPI virtual circuit concentration and multiplexing services on behalf of the X.25 virtual circuits. This LU appears to the NCP as a line and is placed in session with the CTCP in the host.

Define the following to the TPF system:

- The application or CTCP that processes the FTPI messages
- The names of the NPSI FTPI LU resources
- The process selection vector (PSV) routines.



For more information about defining SNA resources to the TPF system, see “Defining SNA Resources to the TPF System” on page 193.

### TPF CTCP Definition

CTCP is the host LU that handles X.25 traffic. In VTAM, the CTCP is viewed as an intermediate program that translates between X.25 and SNA message formats. In TPF, you have different implementation choices for a CTCP. A TPF LU that serves the role of a CTCP can be any of the following:

- An application that sends and receives message traffic in the format required by NPSI. The LU is defined via the SIP MSGRTA macro. As part of the SIP stage 1 process, an application name table definition (ANTDEF) statement is created to serve as input the OSTG process.
- An application that does not understand the message format used by NPSI. For these applications, you must provide a process selection vector (PSV) routine to translate between the message format used by NPSI and the format required by the TPF application.

The use of a PSV routine to intercept message traffic and modify TPF output processing is supported for the following resource types:

- 3270
- 3600
- MCH\_LU, VC\_LU, FTPI, XALCI, AX001, AX002
- NEF.

Regardless of the method used to create a CTCP LU, your application or PSV routine is responsible for translating between the data formats used by NPSI and those required by the TPF application.

### CTCP Definition

The CTCP pseudo LU is an LU that serves as a port between FTPI and the CTCP. There is a difference between operating with or without FTPI. Without FTPI, NPSI requires an LU and a session for each virtual circuit; with FTPI, there is a single LU and a single session for each CTCP or host. The format and parameters used to define a CPLU are shown in Figure 12.

lu_name	RSC	LUTYPE=FTPI [,PACING=0 n] [,AWARE=YES NO] [,PSV=name] [,LEID=nnnnnn]
---------	-----	--

Figure 12. RSC Definition Statement for an FTPI LU

The description of the FTPI values are:

#### lu\_name

Specifies the name of the pseudo CTCP LU.

#### LUTYPE=FTPI

Specifies the LU is a Fast Transaction Processing Interface (FTPI) LU.

#### PACING=0|n

Specifies the pacing value to use for message traffic sent by TPF. A value of zero (0) means that pacing is not used.

#### AWARE=YES|NO

Specifies whether the session awareness exit should be started whenever a



session with the LU has been established or terminated. For additional information on session awareness, see “TPF SNA Session Awareness” on page 327.

**PSV=name**

Specifies the name of the PSV routine to receive control when a message is received or sent to the LU.

**LEID=nnnnnn**

Specifies from 1 to 3 bytes of data to be passed to the PSV routine when an input message is received from the LU.

An example of the coding of the RSC statement for a CPLU is shown in Figure 13 on page 37.

```
CPLU1      RSC      LUTYPE=FTPI,
                  PACING=0,
                  AWARE=NO,
                  PSV=FTPI
```

Figure 13. RSC Coding Example Statement for an FTPI LU

See *TPF ACF/SNA Network Generation* for more information about the OSTG RSC statement.

**VTAM Considerations**

The only VTAM consideration for FTPI support is to provide a default VTAM log mode definition for the FTPI LU, as shown in Figure 14.

FTPI	MODENT	FM PROF=X'03',	FM Profile 3
		TS PROF=X'03',	TS Profile 3
		PRIPROT=X'90',	Exception response
		SECPROT=X'90',	Exception response
		COMPROT=X'0040',	Half-duplex contention
		RUSIZES=X'8989',	Max 3860 in, 3860 out
		SSNDPAC=X'00'	No send pacing
		SRCVPAC=X'00'	No receive pacing
		PSERVIC=X'01000000000000000000000000000000	LU type 1

Figure 14. VTAM Log Mode Entry for FTPI

For more information about the VTAM log mode definition, see *VTAM Resource Definition Reference*.

**NPSI Considerations**

NPSI stage 1 user macros describe the X.25 network. The overall structure of the macro is shown in Figure 15 on page 38. For additional information on the NPSI macros, see *NPSI Planning and Installation*.

X25.CPL	One for each CTCP
X25.NET	One for each network
X25.VCCPT	
X25.OUFT	
X25.MCH	One per multi-channel link
X25.LCG	One per logical channel group
X25.LINE	One per virtual circuit
X25.PU	One per virtual circuit
X25.END	

Figure 15. Organization of NPSI Macros

There are two macro changes to NPSI definition for FTPI:

1. X25.CPL - CTCP pseudo LU
2. X25.MCH - X.25 link.

The X25.CPL macro creates the NCP system generation statements to define a virtual link, PU, and LU. The virtual LU provides a port between NPSI and the CTCP for exchanging message traffic. One X25.CPL macro is required for each CTCP or TPF host. The macro format is shown in Figure 16 on page 38.

```
X25.CPL CTCPNO=n,
        LOGAPPL=ctcp_name,
        MINDATA=mm,
        MAXTIME=time,
        MAXDATA=size,
        DLOGMODE=mode_name,
        MODTAB=mode_table
```

Figure 16. Organization of X25.CPL Macro

The description of the X25.CPL macro values are:

**CTCPNO=n**

This is a 1-byte value associated with the CTCP or TPF application.

**LOGAPPL=ctcp\_name,**

Requests VTAM to automatically establish a session between the CTCP pseudo LU and the CTCP when both:

- The NPSI CTCP pseudo LU is active
- The TPF CTCP LU is active.

**MINDATA=mm,**

Specifies the number of X.25 packets that FTPI should accumulate before considering the message buffer full. When the message buffer is full, it is sent to the CTCP.

**MAXTIME=time,**

Specifies the time in tenths of a second that FTPI should wait before sending a partial message buffer to the CTCP. This value avoids the need to fill the message buffer before sending the data to the CTCP.

**MAXDATA=size,**

Specifies the maximum message size that can be transmitted. If the message sent by the CTCP is larger than MAXDATA, NPSI sends it as several messages.

**DLOGMODE=mode\_name,**

Specifies the name of the VTAM mode table entry associated with the CTCP pseudo LU. See Figure 14 on page 37 for an example of the entry.

**MODTAB=mode\_table**

Specifies the name of the VTAM mode table that contains the mode table entry for the CTCP pseudo LU. If MODENT is not coded, the first entry in this table is the mode table entry for the LU.

The coding of X25.CPL card is shown in Figure 17 on page 39.

	X25.CPL CTCPNO=0,LOGAPPL=NEFA,MINDATA=120,MAXTIME=10,MAXDATA=35665	X
*	X25.CPL CTCPNO=1,LOGAPPL=NEFB,MINDATA=121,MAXTIME=20	

Figure 17. CTCP Pseudo LU Coding Example

**X25.MCH**

The parameter to indicate that the Fast Transaction Processing Interface (FTPI) option of NPSI is to be used is on the X25.MCH macro. An MCH macro is required for every X.25 physical link. The macro, with the operands specific to FTPI support, is shown in Figure 18 on page 39.

	X25.MCH FTPI=YES CUD0 SUBD [,CTCP=(m1,m2,...mn)] [,CUD0=(n1,n2,...nn)] [,SUBD=(p1,p2,...pn)]
--	---

Figure 18. X.25 MCH Macro

The description of the X25.MCH values are:

**FTPI=YES|CUD0|SUBD**

Specifies that FTPI processing is to be used:

- YES specifies that all virtual circuits on the link are processed by the same CTCP.
- CUD0 or SUBD indicates that X.25 CALL REQUEST selects the specific CTCP to handle the virtual circuit. The CTCP chosen to handle a virtual circuit is based upon either:
  - CUD0 or the first byte of the CALL USER DATA in the CALL REQUEST packet.
  - or:
  - SUBD of the last digit of the called DTE address in the CALL REQUEST packet. This technique is called subaddressing.

**CTCP=(m1,m2,...mn)**

Used with either the CUD0 or SUBD operand to select a specific CTCP number. A CTCP's number is defined on the X25.CPL macro.

**CUD0=(n1,n2,...nn)**

Used with the CTCP= operand to select a CTCP based upon the CUD0 value.

**SUBD=(p1,p2,...pn)**

Used with the CTCP= operand to select a CTCP based on the subaddressing.

An example of the coding of X25.MCH card is shown in Figure 19 on page 40. For additional information on coding the X25.MCH card, see the *NPSI Planning and Installation*.

X25.MCH ADDRESS=1,	X
FTPI=CUD0,	X
CUD0=(00,09,01),	X
CTCP=(01,02,03),	X
LCGDEF=(1,10),	X
PKTMODL=8,	X
FRMLGTH=3500,	X
TDTIMER=3,	X
LCN0=NOTUSED,	X
LLCLIST=(LLC4),	X
GATE=GENERAL,	X
MWINDOW=2	X

Figure 19. X.25 Link Definition

## Operations

To allow traffic to flow between TPF and the network requires activating:

- VTAM and the VTAM Logon Manager
- NCPs that connect TPF to the network
- TPF and its connections to the NCP.

With FTPI there are two additional steps to the activation procedure:

1. Establishing a session between the FTPI LU in the NCP and the TPF application
2. Activating the X.25 multichannel links.

## Activating the TPF CTCP

Figure 20 on page 41 summarizes the line flows needed to establish a session between the TPF CTCP and the FTPI pseudo LU. NPSI FTPI support uses a pseudo link, PU, and LU to provide a port or connection to the CTCP. Typically, the session between the CTCP and the FTPI LU is automatically activated when the VTAM operator requests activation of the pseudo link, PU, and LU.

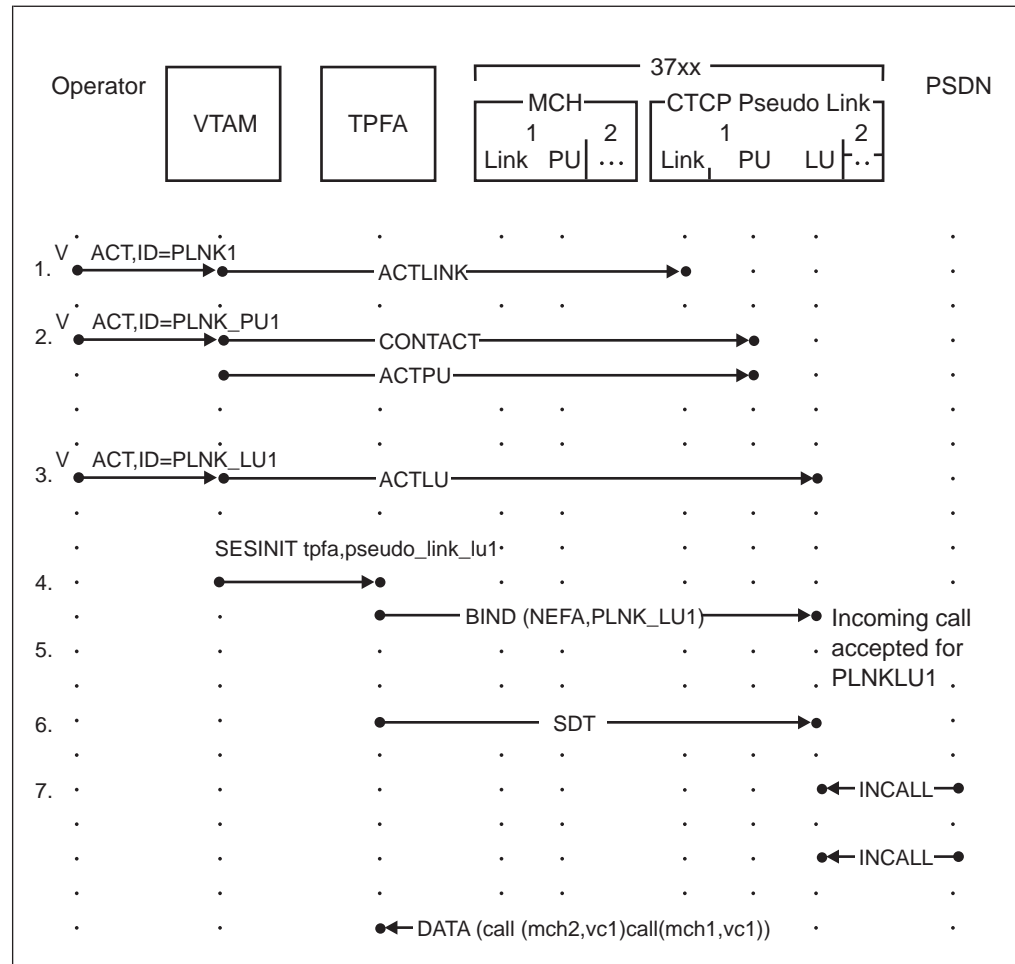


Figure 20. Pseudo Link Activation

1. The pseudo link is activated by the VTAM operator.
2. The pseudo PU is activated by the VTAM operator. Normally, the PU is defined as ISTATUS=ACTIVE, and the PU is automatically started when the link is started.
3. The pseudo LU is activated by the VTAM operator. As is the PU, the LU is usually defined as ISTATUS=ACTIVE, and the LU is automatically started when the PU is started. The LU is also defined as autolog, and the session with the CTCP is requested when the LU is active.
4. As part of the automatic logon process, the VTAM Logon Manager sends a SESINIT message to TPF requesting a session between the FTPI LU and the TPF CTCP.
5. TPF sends a bind command to establish the FTPI LU-CTCP session.
6. TPF allows data flow by sending a Start Data Traffic (SDT) command.
7. Incoming CALL REQUESTS are blocked into a PIU and sent to the CTCP.

## Activating Multi-Channel Links

Figure 21 on page 42 shows the activation of an X.25 link. The description is for a single link; however, in practice, a single operator command activates both the CTCP session and the X.25 links.

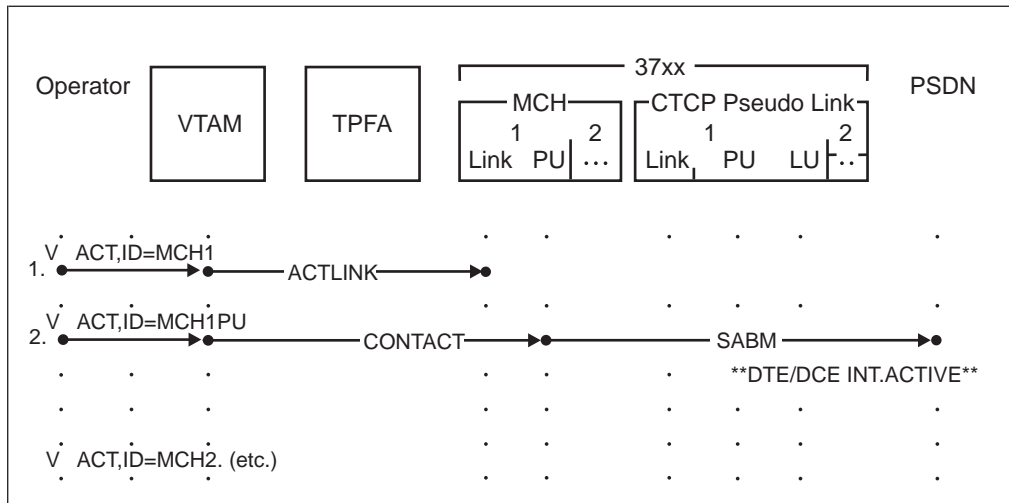


Figure 21. Multi-Channel Link Activation

1. The VTAM operator activates the X.25 link. Normally the links would be initially defined as active; the links would be automatically activated when the NCP is started.
2. The VTAM operator requests activation of the X.25 link station or PU. Normally, the PUs would be initially defined as active; the PUs would be automatically activated when the NCP is started.

## Traces

The 3 traces provided for FTPI are:

1. **PIU Trace:** TPF traces each PIU sent and received between an FTPI LU and TPF. The PIU can contain 1 or more blocked messages. The PIU trace collection and reporting programs continue to trace PIUs and do not attempt to trace each message in the PIU. The TPF message collection facility should be used to trace messages.
2. **Message Collection:** Captures a copy of each message sent and received from an FTPI LU. Message collection is provided by the Comm Source and ROUTC function.
3. **Real Time Trace (RTT):** RTT traces system and application activity for a specific terminal.

## TPF Mapping Support

TPF mapping support builds and formats data streams from application to terminal and from terminal to application. Input messages from 3270 devices contain both control characters and text. Mapping support deletes the control characters and arranges the text in an application compatible format. Conversely, as application-created output messages consist only of text, mapping support inserts the control characters necessary to display the message on a particular 3270 device. Applications receive input messages in AMSG format.

**Users are responsible for coding applications that interface properly with TPF mapping support.** For input messages, mapping support is used to delete control characters; for output messages it is used to insert control characters. Output messages need control characters inserted before the ROUTC macro is issued.

In bracket communication mode, operators of 3x74/3276-attached terminals may send a message only after receiving an output message with the change direction indicator set. In contention mode, messages may be sent at any time. Therefore, host applications are expected to set the change direction indicator (in the RCPL), thereby controlling bracket communication. TPF ensures the change direction indicator is set when applications release saved input messages.

See *3270 Data Stream Programmer's Reference* for more information on the operation of 3270 devices.





---

## Airlines Line Control (ALC) Support through SNA

When accessing non-SNA terminals via an SNA interface, TPF maintains the Application Program Interface (API) for application programs that address physical devices. This is done using either:

- TPF NEF support

### **Note:**

TPF Network Extension Facility (NEF) support depends on either:

- The Network Extension Facility (NEF2 PRPQ P85025), or
- Airlines Line Control Interface (ALCI) feature of ACF/NCP.

All subsequent occurrences of NEF in the text of this book are superseded by the previous information.

- TPF-provided Airline X.25 (AX.25) support with the NCP Packet Switching Interface (NPSI)
- TPF-provided XALCI support with the NPSI GATE/Fast Transaction Processing Interface (NPSI GATE/FTPI).

Most often the physical device is a terminal addressed via a Logical End Point-Identifier (LEID). These devices also use an Agent Assembly Area (AAA). The NEF/ALCI, AX.25, and XALCI enhancements permit the use of ALC protocols for a select group of non-SNA devices across an SNA interface.

See the *ACF, NCP, ALCI General Information* for more information about devices supported across these interfaces. These devices are also supported across the AX.25 interface.

Throughout the remainder of this publication, NEF implies either the NEF PRPQ or ALCI feature environments.

---

## TPF Host Interfaces

Using TPF multiple host extension support, NEF\_supported terminals can directly communicate with one or more TPF hosts via cross\_domain support. AX.25's use of Permanent Virtual Circuits (PVCs) requires direct attachment to a specific host in the PU\_5 environment. This is not required for PU 2.1. XALCI's use of GATE/FTPI allows the use of switched virtual circuits (SVCs) as well as PVCs. Because all 3 protocols are SNA-supported, either a block or byte multiplexer channel may be used.

Combining SNA and ALC support requires generation of both SNA and ALC data records. Further, it is necessary to cross-reference the basic SNA record (the resource vector table (RVT)) with the definition of each ALC supported terminal. The terminal address table (WGTA) is used for this purpose.

SNA (TPF) maintains the real network addresses of all logical units (LUs) in the NCP after they were discovered from the VTAM Communications Management Configuration (CMC) when the LU-LU session was set up. However, SNA (TPF) is not aware of the attached ALC terminals, lines, or interchanges. Each LU has a resource identifier (RID), network address, and node name. To maintain the ALC device address format, each terminal is given a 3-byte address known as Logical

End-Point Identifier (LEID). The LEID is used to create the AAA initialization table (UAT) entries for the ALC terminals. Each NEF/AX.25/XALCI terminal address table (WGTA) entry contains the RID of the associated LU. Each NEF terminal on a 37x5 references to the same LU; each NEF LU references to all of the NEF terminals on the 37x5. The same is not true for AX.25 LUs. There may be several AX.25 LUs in the 37x5, each represent 1 or many terminal controllers. GATE/FTPis' use of virtual circuit concentration allows a single LU type 1 to support multiple X.25 virtual circuits. Each of these virtual circuits can, in turn, support multiple terminals. At least 1 FTPI LU is required per TPF host. Multiple GATE/FTPI sessions to a single TPF host are supported.

**Note:** WGTA entries are created from user-supplied UAT records. VTAM Communications Management Configuration (CMC) must own NEF, AX.25, and XALCI resources. For NEF, the terminal interchange (TI) is identified to VTAM as a physical unit (PU). One LU per NCP with NEF and one or more LUs per NCP with NPSI/AX.25 or NPSI GATE/FTPI needs to be defined to a VTAM CMC. All terminals attached to the 37x5 through links and terminal interchanges communicate through the NEF, AX.25, and FTPI LUs. These LUs are also defined to TPF as cross\_domain resources (CDRSCs). This implies that TPF is not aware of the ALC lines or terminal interchanges. All terminals accessed using NEF/AX.25/XALCI LUs are identified to TPF as LEIDs through an entry in the terminal address table (WGTA). The LEID need not have any correlation with the physical address of the terminal. However, the first (most significant) byte of the LEID must be above the range of valid symbolic line numbers.

Input or output data messages pass through 4 message formats when sent over the network. An input/output message is received or sent from the terminal interchange in ALC format. This format includes the real interchange address (IA), the synchronization check characters, the cyclic check characters, and the data. NCP (NEF/NPSI) converts this to or from a path information unit (PIU). The request unit (RU) is in the PIU. The RU contains terminal address information that is used to obtain the LEID followed by the data. TPF converts the input message to TPF application message format. On input, the communications source program activates the application in the same format (AM0SG) or in the input message format (MI0MI). The user specifies the format during system initialization. Output message processing is based on the output message format AM0SG or UI0OM.

## Session Control via VTAM SSCP

The VTAM SSCP views NEF communication lines as SNA lines, and ALC terminal interchange units as physical units (PU). A pseudo\_NEF application logical unit is created in the TPF host. The NCP NEF logical unit is then bound in a session with the NEF application logical unit. The NEF application logical unit is known as NEFx where x is the CPU ID of the host. All NCP NEF logical units that communicate with the host must be bound to NEFx. This is not an application that a terminal operator can **log into**. Further, it does not appear in the routing control application table (RCAT). In the RVT, the NEF logical unit is defined as permanently logged. For AX.25, the TPF CCP code performs, on behalf of the application, the functions provided by the host NEF LU. FTPI LUs that are defined to TPF as using the XALCI protocol can be logged into any application. This dummy application is simply used for session startup and takedown. The XALCI data streams are used, in a manner similar to NEF, to allow access to the Log Processor. Terminals supported by XALCI are independent of the application that the FTPI LU is logged into.

The VTAM CMC system operator controls the NEF/AX.25/XALCI network using SNA commands. One exception exists: the operator does not control or communicate with ALC terminals. The TPF operator addresses the terminals via its LEID. The terminals can receive unsolicited messages sent through the ZSNDU command. The TPF operator can perform terminal diagnostics functions and terminal display functions via the ZTERM command. The ZTERM command displays the RCB file address, WGTA entry address, the application name, and the logical unit with which the terminal is associated.

## Data Flow

**Input Messages:** When a terminal operator enters a data message, it is sent from the terminal to TPF. Before being transmitted to the host, if required by the user, NEF provides the ability to translate the data from ALC to EBCDIC. TPF XALCI support assumes that the data has already been converted to EBCDIC. The RU portion of the PIU received by TPF contains terminal address information and input data, including the end-of-message character. The terminal address information is as follows:

1. NEF/XALCI - 3-byte LEID
2. AX.25 - 1 or 2-byte address information
  - One byte - Terminal Address (TA) (third byte of the LEID)
  - Two bytes - Interchange Address / Terminal Address (IATA) (second and third bytes of the LEID).

**Note:** During NCP generation the user specifies if translation/editing should be performed by NEF.

On receipt of the PIU, TPF determines the origin and the content, data, or response. If it is a data message, segment CNE1 is activated for all three classes of traffic. Otherwise, SNA OPZERO processes it like any other response.

**Note:** NEF/AX.25 resources are defined as unrecoverable. Therefore, error recovery does not exist for these messages.

The ALCI Comm Source program, CNE1, reformats input messages from a PIU to the TPF application message format. This includes setting up the LEID in the ECB from the input PIU. The message is also converted from an SNA format to a high speed message format. When the message is reformatted, an interface is provided to real-time trace (RTT) to allow *agent* trace as well as *nodename* trace. See *TPF Program Development Support Reference* for more information on these trace facilities.

The resource identifier (RID) field in the WGTA entry is updated to reflect the current association of the terminal with the NEF, AX.25, or FTPI/XALCI logical unit.

When CNE1 has completed successfully, the input message is processed as any other high-speed input message. Messages can be passed to either of the following:

- Log processor
- System message processor
- Locally resident application (as indicated in the RCAT entry)
- Message router.

Therefore, on exiting the input interface program, the input takes a parallel processing path to that of other high-speed ALC input messages.

**Output Messages:** When an application program, including the system message processor, sends an output message to a NEF/AX.25/XALCI terminal (in response to an input message), it uses either ROUT- or SEND-type macros. The ROUT or SEND macro service routines, or both, view this terminal as a high-speed ALC device. One exception exists: the first byte of the LEID is above the range of valid symbolic line numbers. Real symbolic line numbers are calculated during system generation with input supplied by the user. This data is stored in the system communication configuration table (CK6KE).

Support is included in the communication control program (CCP) to avoid accessing the system communication keypoint records. This logic intercepts the CCP when the symbolic line number is out of the valid range where a CCP error condition would have occurred without NEF, AX.25, or XALCI support. The LEID is then used to locate the terminals entry in the terminal address table (WGTA). The WGTA contains the RID that points to the resource vector table (RVT) entry. If the RVT entry cannot be used, CCP error logic is followed. Otherwise, the RVT entry is used to locate the device.

The technique described is used in the common CCP macro routine (segment CLXC) to determine when it is necessary to reformat the output message to a PIU format. In addition, only the SENDA, SENDC, SENDL, and SLMTC macros are supported. The ROUTC macro is only supported if it generates a SENDA, SENDC, or SENDL macro. In other words, ROUTC starts the appropriate macro for the appropriate device.

If an error occurs when locating the WGTA entry, a system error is issued.

The ALCI using the CLXV SNA output routine performs the reverse of the ALCI Comm Source interface program. When the message is reformatted, the SNA SOUTC macro service routine is activated. This routine performs the scheduling and transmission of the message to the 37x5. For XALCI, SOUTC also performs blocking of output messages according to the GATE/FTPI interface before transmission. The SOUTC routine also ensures that an SNA session is established with the receiving logical unit if the request was made via SLMTC. If required, it then translates the message. Lastly, the SOUTC routine informs the LU if the message is multisegmented and if a response is expected from the receiving terminal. For NEF, AX.25, and XALCI messages, pacing is not performed.

---

## Command Support

SNA commands (VTAM VARY NET) are used to control the network. In SNA processing, this is accomplished via the nodename of the NCP, the SNA line, and the physical and logical units. As SNA does not have a node name for ALC terminals, they must be addressed via an LEID.

The user controls the assignment of the LEID to each ALC terminal.

## Addressing an ALC Device

Normal ALC command support is not provided for NEF/AX.25 devices except for terminals. This portion of the network is not controlled or displayed via the ZL—type command. It is not considered a true high speed line.

Command support for terminals falls into two categories:

1. Terminals that require specific interfaces to provide functionally equivalent support

2. Terminals that use existing 3270 SDLC support (which also use an LEID addressing scheme for SNA resources).

The convert node name (ZNCVT), alter/display CRAS (ZACRS/ZDCRS), and 1052 fallback functions fall into the category of 3270 SDLC support. Here, LEID is used to locate the resource vector table (RVT) entry. See *TPF Operations* for more information on each of these commands.

An NEF/AX.25/XALCI terminal operator can operate from either a prime or alternate CRAS. Therefore, an entry is placed in the computer room agent table (CRAT). As a result of this, a terminal operator can **log into** application SMPx (where x is the symbolic CPU ID) and enter Z-type commands to the system message processor (CSMP).

The Terminal Reset function (ZLRST), applicable only to NEF support, requires special interfaces. Here, it is necessary to use the pseudo ALC address (LEID) to locate the SNA resource vector table, rather than the system communication keypoints. Normally this is done when the package detects what appears to be an invalid symbolic line number. The terminal reset function allows a system operator to reset a 1980-24 printer's buffer (see CVTR). NEF support does not provide for the terminal interchange (TI) reset function. This function can only be performed from the owning VTAM CMC by activating the physical unit (ACTPU) of that terminal interchange. ACTPU, in turn, causes NEF in the 37x5 to generate a configuration report. This configuration report gives TPF the information about the correlation between the NEF logical unit (represented by an entry in the RVT) and the NEF terminals (LEIDs, entries in WGTA).

---

## TPF/NEF SNA Multiple Host Support

NEF support must be present in any TPF host that communicates with a NEF terminal. A VTAM CMC host owns all NEF resources (lines, physical units, logical units). TPF hosts cannot own any NEF resources, but a given terminal can communicate with any one of the hosts provided that the terminal is defined (in WGTA) to that host.

Multiple host support requires two basic items:

1. Multiple logical units
2. Transaction analysis.

To accommodate an SNA multiple host environment, a NEF LU is defined for each host. Figure 22 on page 50 illustrates a multiple host network.

Each NEF terminal has a single address (LEID) that is known in each of the hosts. Therefore, user-assigned LEIDs for NEF terminals must define the first byte of the LEID outside the range of the symbolic line numbers in all of the hosts. Because the terminal has a direct link to each host, it is known to each host via its address, LEID, and the CPU ID of that host.

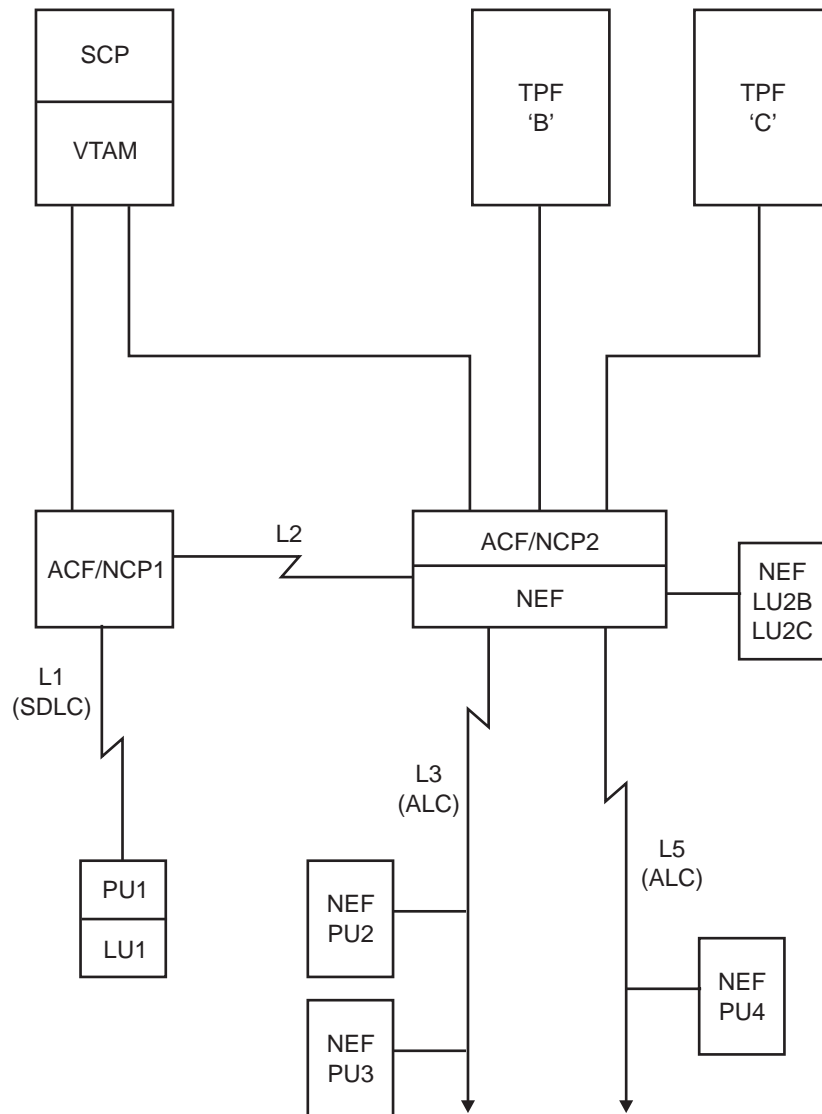


Figure 22. Multiple Host Network in NEF

## X.25 ALC Interface for Switched or Permanent Virtual Circuits (XALCI)

XALCI support allows ALC-type terminals attached to a packet switched data network (PSDN) to communicate with existing TPF applications. XALCI is similar to TPF NEF/ALCI and AX.25 support of the AX001 and AX002 protocols, but differs in the following ways:

- NPSI with the FTPI option is used in place of NPSI with the GATE option or the ALCI feature of NCP. This effect of this change is limited to the handling of the NPSI header.
- Terminal registration, XALCI header management, and correlation of terminals to an LEID are **not** performed by NPSI. These functions are implemented in a remote intelligent workstation (IWS) or gateway and are **not** supplied by NCP, NPSI, or TPF.
- XALCI supports switched virtual circuits (SVCs). The effect of using SVCs is that the correlation between LEID value and virtual circuit must be delayed until the

virtual circuit has been established and the registration report has been received. With AX001/AX002, the correlation between virtual circuit and LEID is made at network definition time. TPF provides a user-replaceable module to process virtual circuit setup requests. The replaceable module is implemented as a process selection vector (PSV) routine. Users can either use the sample PSV routine or substitute their own.

## User-Replaceable PSV Routine

The sample PSV routine bases its processing on the NPSI/FTPI command type. The valid command types are:

- Clear - User Data
- Qualified Data - DCE Reset Confirmation
- Diagnostic - Error Information
- Incoming Call - Interrupt
- Interrupt Confirm - Reset Confirm

The sample PSV routine classifies the input based on the FTPI header. The general processing is:

- CALL REQUEST commands are accepted by issuing a ROUTC to return a CALL ACCEPTED command and the ECB EXITCed. When the virtual circuit is established, the next message received should be an ALCI registration report.
- Data is passed through to the ALCI logic. The ALCI processing is described in “Airlines Line Control (ALC) Support through SNA” on page 45.
- All other types of NPSI commands are discarded by EXITCing the ECB.

### Tables

TPF's XALCI support code uses the WGTA as its control structure. Based on the LEID in the data stream, the WGTA slot is used to store the MCH\_ID, VC\_ID, and the correlation number of the input message for recall and insertion into the corresponding output message. The WGTA is expanded by 4 bytes to hold this additional information.

## Network Definition

### TPF Considerations:

The only network definition requirement for XALCI support is the need to:

- Define the NPSI LUs that connect TPF to the packet switch data network (PSDN).

**Note:** Use the XALCI resource type for all FTPI LUs that use the XALCI interface.

- Include the process selection vector routine to handle X.25 message traffic that complies with the ALCI standards.

**Defining the NPSI LU:** Define to the TPF system the NPSI or NPSI FTPI LU that connects the TPF system to the network and non-SNA terminals using ALCI support. The format and parameters used to define the NPSI or NPSI FTPI LU are shown in Figure 23 on page 52.



lu_name	RSC	LUTYPE=XALCI ,PSV=XALCI [,PACING=0] [,AWARE=NO] [,LEID=FFFFFF]
---------	-----	--

Figure 23. RSC Statement Definition Statement for an FTPI LU

The description of the XALCI values are:

**lu\_name**

Specifies the name of the NPSI LU.

**LUTYPE=XALCI**

Specifies if the LU uses the GATE/FTPI interface of NPSI and uses XALCI data streams to perform the FTPI header management.

**PACING=0**

Specifies the pacing value to use for message traffic sent by TPF.

**Note:** A value of zero (0) must be specified and means that pacing is not used.

**AWARE=NO**

Specifies whether the session awareness exit should be started whenever a session with the LU has been established or terminated. NO must be specified.

**LEID=FFFFFF**

Specifies from 1 to 3 bytes of data to be passed to the PSV routine when an input message is received from the LU. This parameter is optional and is neither required or used by the sample PSV routine supplied with TPF.

An example of the coding of the RSC statement for XALCI support is shown in Figure 24 on page 52.

CPLU1	RSC	LUTYPE=XALCI, PACING=0, AWARE=NO, PSV=XALCI
-------	-----	--

Figure 24. RSC Coding Example Statement for an XALCI LU

For more information about defining SNA resources to the TPF system, see “Defining SNA Resources to the TPF System” on page 193.

**Defining the XALCI PSV Routine:** TPF provides XALCI support by bridging its support of the NPSI FTPI option with the support for ALCI. The bridge between FTPI and ALCI support is implemented as a user-replaceable PSV routine. To include the PSV routine in the TPF system, you must:

- Assemble program segment COBU. COBU is the PSV table and includes and entry for XALCI. COBU must be allocated as a file resident program.
- Allocate CNEA as either main storage of file resident, but not defined as corefast. The PSV routine itself is a program segment named CNEA.

The source code for the PSV table, COBU, includes the default PSV routine for XALCI. Figure 25 on page 53 shows the coded to be added to segment COBU.



IPSVE	NAME=XALCI, PGM=CNEA, RESERVE=NO, DFC=NO
-------	---

Figure 25. XALCI PSV Entry Shipped with TPF

The macro to define a PSV table is shown in Figure 26 on page 53.

IPSVT	BEGIN END
-------	-----------

Figure 26. Create PSV Table Macro

The macro to define a PSV entry is shown in Figure 27 on page 53.

IPSVE	NAME=XALCI ,PGM=CNEA [,RESERVE=NO] [,DFC=NO]
-------	---

Figure 27. Create PSV Entry Macro

**NAME=XALCI,**

Specifies the 1 to 6 character PSV name. The character set is restricted to the letters A through Z and the numbers 0 through 9. The name XALCI was chosen for this example, but any valid name is acceptable.

**PGM=CNEA,**

Specifies the user-replaceable program that bridges FTPI messages to the ALCI interface. The program can be file or main storage resident but cannot be corefast.

This program segment must reside in the BSS, but may also be resident in other subsystems in an MDBF environment.

**RESERVE=NO,**

Specifies whether the entry is reserved for future use. If reserved, a SERRC is generated in place of the ENTNC. The default is NO.

**DFC=NO**

Specifies whether the user PSV provides the SNA-architected DFC layer. The default is NO.

Figure 28 on page 53 shows a sample of PSV definition for XALCI support.

IPSVT	BEGIN
IPSVE	NAME=XALCI,PGM=CNEA
IPSVT	END

Figure 28. Sample XALCI PSV Definition Statements

For more information about the IPSVT and IPSVE macros, see *TPF General Macros*.

## X.25 Gateway Considerations

TPF's XALCI support is based on a coordinated effort between TPF base code and a remote intelligent workstation (IWS) or gateway. The TPF code performs the

blocking and deblocking of data to and from GATE/FTPI, as well as the FTPI header management and ALCI functions. For additional information, see “Airlines Line Control (ALC) Support through SNA” on page 45. The FTPI header management is performed using WGTA and is based on architected XALCI headers discussed in “XALCI Header Formats”. The gateway is responsible for:

- Managing the virtual circuit to NPSI and TPF. This involves, for example, handling the various X.25 commands and error handling.
- Building and sending a registration report that lists the LEID of all terminals supported by the gateway. This should be done when the virtual circuit to TPF is first established, as well as when supported terminals are powered on after the gateway has established the virtual circuit to TPF.
- Inserting of the proper XALCI header information. This consists of 3 bytes (the X.25 Q byte and 2 bytes that are similar to NEF/ALCI headers that describe the data in “XALCI Header Formats”), as well as a 3-byte LEID or, in the case of a registration report, a list of LEIDs.
- Ensuring that the data that is sent to TPF is in EBCDIC. TPF XALCI code assumes that all headers *and* data are in EBCDIC.
- Removing of XALCI header information from responses received from TPF and routing to the correct destination based on the LEID in the XALCI header.
- Ensuring that the data received from TPF is translated to the correct character set. Headers and data transmitted from TPF are in EBCDIC.

***XALCI Header Formats:*** Figure 29 on page 55 shows an example of the various XALCI headers.

#### User Data (Q=0)

0 – 3	X'00'	X'05'	X'01'	24 bit
4 – n	LEID		User Data	

#### User Data (Q=1)

##### Registration Report

0 – 4	X'02'	X'02'	X'02'	24 bit
5 – n	LEID	24 bit LEID		...

#### Terminal Reset

0 – 4	X'02'	X'05'	X'06'	24 bit
5 – n	LEID	Reset Message		

#### Error Report

0 – 4	X'02'	X'0A'	X'07'	Error Code
5 – 9	Error Code (cont.)		X'80'	24 bit
10 – n	LEID			

#### Valid Error Codes

X'10086021' Unknown LEID  
X'084B6031' Inactive LEID

Note: The first byte off the XALCI header corresponds to the X.25 Q byte. All commands with the exception of User Data should be transmitted on the virtual circuit as qualified data.

Figure 29. XALCI Header Formats

## XALCI Configuration

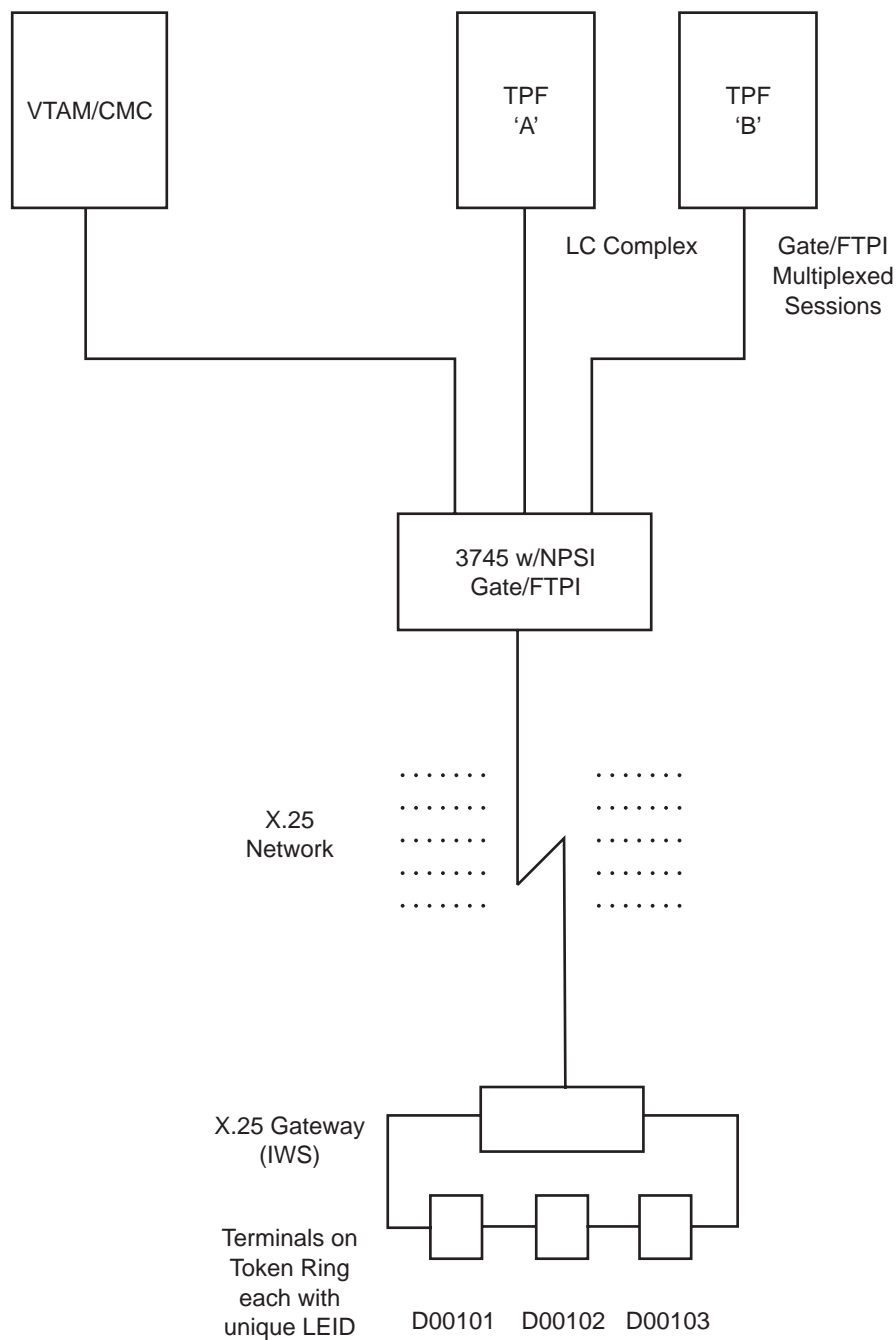


Figure 30. XALCI Configuration

Figure 30 on page 56 shows a typical XALCI configuration. GATE/FTPI multiplexed sessions are used to connect each host in the TPF loosely coupled complex to NPSI in the 3745. The 3745 is owned and loaded by the VTAM CMC. On the remote side of the X.25 network is an intelligent workstation (IWS) that supports multiple workstations attached via token ring. The IWS, or X.25 gateway, manages the addition to and removal of XALCI header information for all traffic originating

from or destined for its supported workstations. The correlation of data to workstation is performed using a unique LEID that is associated with each workstation on the token ring.

On initial activation, the X.25 gateway is responsible for establishing a virtual circuit with the TPF complex. This is achieved by issuing a call to NPSI. Through the use of call user data or subaddressing, the gateway can specify a particular processor in the complex. Also, depending on NPSI generation options, the GATE/FTPI function can perform alternate routing of the CALL if the primary target host is unavailable. NPSI then forwards the call to the specified TPF CTCP.

The CTCP may analyze the call and either accept the call with a CALL ACCEPTED or reject it with a CLEAR. If CALL ACCEPTED is sent, this is then forwarded by NPSI across the newly established virtual circuit to the originating gateway. When the gateway receives the CALL ACCEPTED, the XALCI registration report is then built and transmitted on the virtual circuit. The registration report specifies the LEID of every workstation that the gateway supports. This report is primarily used by TPF to activate the LMT queues of hardcopy devices. The CTCP also uses the registration report to store routing information, such as the device's multichannel ID, virtual circuit ID, and current correlation number that are all required for GATE/FTPI support. When the registration report has been transmitted to TPF, terminal traffic may begin to flow between the gateway and the TPF complex.

To trace the flow of a message in an XALCI configuration, assume that LOGI RES0 is entered by one of the workstations on the token ring. The input message is transmitted on the ring to the gateway. The gateway then inserts the XALCI header information that describes this as user data, as well as the LEID that identifies the terminal. This data is then forwarded across the X.25 network on the virtual circuit. NPSI receives the input message and blocks it with other traffic that is destined for the same TPF CTCP. The data is accumulated until either a user-definable timer expires or the user-definable number of messages has been blocked into the PIU. In either case, the PIU is then forwarded across the multiplexed session to the specified TPF host.

OPZERO manages the deblocking of the deblocking of the messages from the GATE/FTPI blocked PIU based upon header information that was appended by NPSI on each message. Messages are copied by OPZERO to a core block and attached to an ECB. Upon inspection of the origin SNA resource type, SNA COMMSOURCE passes control to ALCI COMMSOURCE. ALCI COMMSOURCE uses the LEID in the data stream to access the WGTA slot for the device, and stores the multi-channel ID, virtual circuit ID and correlation number for this input message in the WGTA. The FTPI and XALCI headers are stripped out, and the message is put in AMMSG format.

Because this is a LOGI message, control is passed to the log processor. The log processor logs the terminal into RES0 and responds with a LOGI COMPLETE data stream that it routes to the originating terminal. The message router determines that the LOGI COMPLETE message is destined for an XALCI resource and passes control to ALCI output transformation code in CCCCCP1. Routines access the WGTA slot for the device based on the LEID and retrieve the FTPI header information that was saved on input. FTPI and XALCI headers are inserted into the output message and control is passed to SOUTC. SOUTC determines that this PIU is destined for an FTPI LU, and control is passed to the FTPI message blocking process. The FTPI message blocking process accumulates PIUs in a 4K core block until either the block is full or a timer expires. (See the *TPF ACF/SNA Network Generation* for

detailed information on LUBLKT in SNAKEY/CTK2.) In either case, the blocked PIU is then transmitted to NPSI across the multiplexed session.

The GATE/FTPI process in NPSI then deblocks the messages from the blocked PIU, removes the FTPI header from the message, and transmits the message on the virtual circuit that was specified in the FTPI header. When the gateway receives the message, it removes the XALCI header information and LEID and sends the message on the token ring destined for the terminal that corresponds with the LEID. LOGI COMPLETE is displayed on the originating terminal.

For more information regarding coding of a GATE/FTPI CTCP as well as more detailed information regarding NPSI and its command flows, see the *NPSI Host Programming* and the *NPSI Planning and Installation*.

---

## TPF Advanced Program-to-Program Communications

The function provided by the TPF Advanced Program-to-Program Communications (TPF/APPC) interface is an implementation of IBM's Advanced Program-to-Program Communications (APPC) architecture. TPF/APPC is an interface that allows TPF transaction programs to communicate with remote SNA nodes that have implemented the APPC interface using LU 6.2 protocols.

Before using TPF's LU 6.2 support, you must be familiar with the SNA protocol as defined in the following publications:

- *IBM Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*
- *IBM Systems Network Architecture LU 6.2 Reference: Peer Protocols*
- *IBM Systems Network Architecture Transaction Programmer's Reference Manual for LU Type 6.2.*

---

### TPF/APPC Components

TPF/APPC provides TPF's implementation of the LU 6.2 architecture. TPF/APPC support consists of the following components:

#### **Presentation services**

Processes the TPF/APPC verbs.

#### **Resource manager**

Services requests relating to conversations and activates the functions required of the half-session components.

#### **Session manager**

Is responsible for insuring that the underlying LU-LU session needed for a conversation is available.

#### **Half-session**

Is primarily responsible for controlling the data traffic flow for a session. This component includes the:

- Data flow control component, which controls the flow of data between half-sessions,
- Transmission control component, which controls the flow of data between the half-session and path control.

Figure 31 on page 60 shows how the components relate to 1 another and what types of interfaces are used between them.

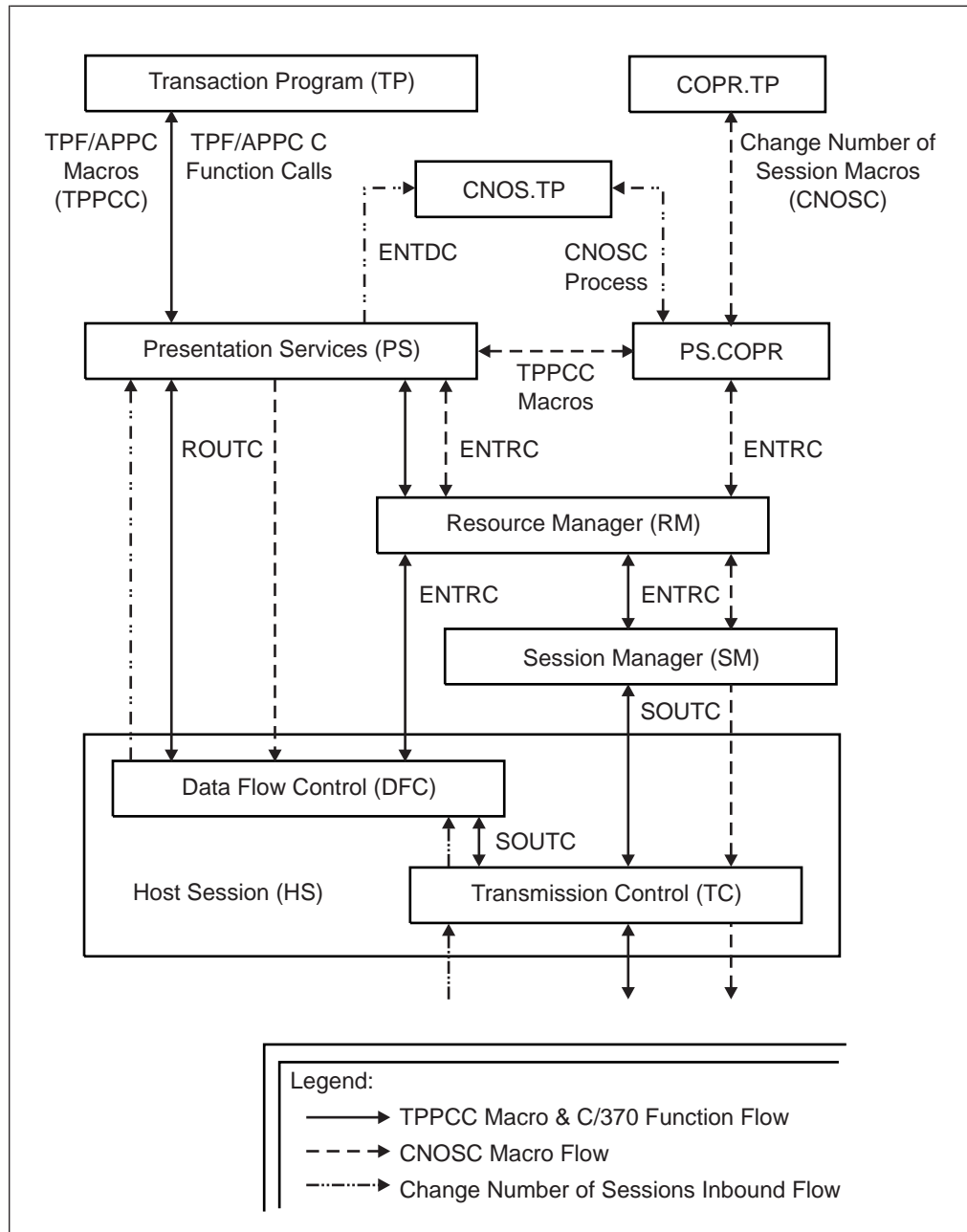


Figure 31. TPF/APPC Component Flow Diagram

## TPF/APPC Installation Checklist

The following describes the tasks you must perform to use the TPF/APPC support.

### General Installation Tasks

The following list contains the tasks that you must perform regardless of your network environment:

- Define the local TPF application programs that represent the TPF/APPC LUs.



There can be many TPF/APPC local LUs in a TPF system. 1 of the LUs defined is considered the default LU and is used by the TPF system for the following conditions:

- When the TPF system issues a change number of sessions (CNOS) INITIALIZE request for a remote LU without specifying a local LU, and this is the first mode name initialized with the specified remote LU.
- When the TPF system issues a single session allocate request and a CNOS INITIALIZE was not previously done for the LU specified on the ALLOCATE verb.

The definition is done with the SIP MSGRTA macro and the offline ACF/SNA table generation (OSTG) program. The output from the MSGRTA macro consists of an ANT deck, which is then used as input to the OSTG process. To define the default TPF/APPC LU, code the MSGRTA macro as follows:

**MSGRTA APLIC=*appl*,EDIT=CHDD,ASNA=APPC**

To define additional TPF/APPC LUs, code the MSGRTA macro as follows:

**MSGRTA APLIC=*appl*,EDIT=CHDD,ASNA=LU62**

In both macro statements, *appl* is the local LU name.

**Note:** Unlike other MSGRTA statements, the MSGRTA statement for TPF/APPC represents the actual SNA logical unit (LU) rather than the end-user application program. For TPF/APPC, the end-user application is referred to as a *transaction program*. The MSGRTA statement defines the LU that is placed in session with remote LUs. This LU, whose EDIT parameter must specify program segment CHDD, then provides the LU services on behalf of the transaction program. See *TPF System Generation* for more information about the MSGRTA macro and *TPF ACF/SNA Network Generation* for more information about OSTG.

See “Defining SNA Resources to the TPF System” on page 193 for more information about defining local TPF applications to the TPF system.

- Define to the TPF system the remote LUs that will communicate with the local TPF LU.

See “Defining SNA Resources to the TPF System” on page 193 for more information about defining remote LU resources to the TPF system.

- If you are using parallel sessions, make sure that the TPF system is connected to the SNA network either as a T2.1 node or as a T5 node through an SNA channel-to-channel (CTC) connection. T5 parallel sessions are not supported through an NCP PU 5 connection. If you are using parallel sessions across a CTC link, the remote LU **must** reside in the adjacent host connected by the CTC link. Single sessions are supported through all links (T2.1 and T5).
- Define or change your remote application programs to request session initiation with the appropriate TPF/APPC LU name.
- Define the remote LUs to VTAM or NCP.  
See the *VTAM Resource Definition Reference* and *NCP/SSP/EP Resource Definition Reference* for more information.
- Define the #CCBRU, #SC1RU, #SC2RU, and #CMSIT fixed file records.

This is done with the RAMFIL statement in the system initialization program (SIP) deck. #CCBRU records are 4-KB records that are used to hold the checkpointed

copies of the main storage conversation control blocks (CCBs). Enough records must be allocated to hold the number of CCBs allocated with the MAXCCB parameter of the SNAKEY macro in the SNA communications keypoint (CTK2, defined by the CK2SN data macro).

#SC1RU and #SC2RU records are 4-KB records that are used to hold the checkpointed copies of the main storage session control blocks (SCBs). Enough records must be allocated to hold the number of SCBs allocated with the MAXSCB parameter of the SNAKEY macro in CTK2.

#CMSIT records are 4-KB records that are used to hold the side information table needed for C language mapped conversations.

See *TPF System Generation* for more information about the SIP RAMFIL statement, and see *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro and the number of records to define.

- Code the following parameters on the SNAKEY macro:

#### **MAXCCB**

Defines the maximum number of conversation control blocks (CCBs) required in your TPF system. See “TPF/APPC Conversation Control Block” on page 67 for more information about CCBs.

#### **MAXSCB**

Defines the maximum number of session control blocks (SCBs) required in your TPF system. See “TPF/APPC Session Control Block” on page 68 for more information about SCBs.

**Note:** There is an interdependency between CCBs and SCBs; if you want to use the TPF/APPC support, you must code both MAXCCB and MAXSCB greater than 0.

#### **TPALLOC**

Defines the time-out value for the TPF/APPC ALLOCATE verb.

#### **TPRECV**

Defines the time-out value for the TPF/APPC RECEIVE verb and any verb that implies a CONFIRM.

#### **TPWAIT**

Defines the time-out value for the TPF/APPC WAIT verb.

#### **PARACOS**

Defines the class of service (COS) name that the TPF system uses when building a PU 5 CDINIT response for a TPF/APPC parallel session.

#### **SINGMODE**

Defines the mode name used for TPF/APPC single sessions initiated by the TPF system.

See *TPF ACF/SNA Network Generation* for details about the SNAKEY macro.

- Define the local transaction programs that are activated by remote transaction programs or by the TPF/APPC ACTIVATE\_ON\_CONFIRMATION or ACTIVATE\_ON\_RECEIPT verb in the transaction program name table (TPNT). The TPNT contains the local transaction program name and the associated TPF E-type program segment that provides the application function. Use the ITPNT macro to define these programs. See *TPF General Macros* for details on this macro.
- If you plan to use the IBM C language mapped conversation interface, you may need to generate the side information table.

The side information table is used to identify a partner LU name, remote transaction program name, and mode name with a symbolic destination name. Use the side information table offline program (CHQI) to generate the table offline. See “Generating the Side Information Table for Mapped Conversations” on page 83 for details on how to use this program. Use the ZNSID command to build or change the table online. See *TPF Operations* for details about this command.

- If you are using TPF/APPC in a loosely coupled environment, you may need to perform the following additional tasks (based on your particular environment):
  - Define 1 TPF/APPC service LU for every processor in the loosely coupled complex.
  - Define the TPF service transaction program to TPNT.
  - Initialize session limits for the TPF/APPC service LUs.

See “Considerations for a Loosely Coupled Environment” on page 103 for more information.

## Installation Tasks for Low-Entry Networking (LEN)

The following tasks must be performed when the TPF system is operating in LEN mode; however, if the TPF system is operating in APPN mode, skip this section.

- Install IBM ACF/VTAM Version 3 Release 4 or higher, which contains enhancements to Logon Manager that are required for the TPF system to be able to start TPF/APPC sessions.
- Define the qualifier number (QN) and the control LU (CLU) name on the channel-to-channel (CTC) statements for connections to VTAM as well as on the ALS statements.

This is done in OSTG when there will be TPF/APPC sessions across a CTC connection managed by the logon manager. Sessions across an ALS connection are always managed by the logon manager. There must be a CLU-logon manager session over each CTC connection if the logon manager will be used. See *TPF ACF/SNA Network Generation* for details about coding the CTC and ALS statements.

See “Defining SNA Resources to the TPF System” on page 193 for more information about defining SNA resources to the TPF system.

- Define TPF/APPC LU alias names to VTAM as CDRSCs under the TPF CTC CDRMs. The alias name is the LU name with the TPF CPU ID suffix and the qualifier number.

See *VTAM Resource Definition Reference* for more information.

- Define to the NCP alias names for the TPF applications and define them as independent LUs.

See *NCP/SSP/EP Resource Definition Reference* for more information.

- Define to the NCP the CLU names for sessions across ALS connections.

See *NCP/SSP/EP Resource Definition Reference* for more information.

## Defining TPF/APPC LUs to the Network

This section describes how to define a TPF/APPC LU to the network based on your configuration.

### Defining LUs across a PU 5 Connection

To define TPF/APPC LUs across a PU 5 NCP connection, define the LUs to VTAM under the owning cross-domain resource manager (CDRM).

To define TPF/APPC LUs across a PU 5 CTC connection for a TPF system operating in APPN mode, also define the LUs to VTAM under the owning CDRM.

Figure 32 shows sample definitions for TPF/APPC primary LUs (PLUs) and secondary LU (SLU) threads. The real name of a TPF/APPC PLU is used.

TPFA	CDRM
LU62	CDRSC
LU62A001	CDRSC
LU62A002	CDRSC
LU62A003	CDRSC
LU62A004	CDRSC
LALA	CDRSC
MONY	CDRSC
MONYA001	CDRSC
MONYA002	CDRSC
MONYA003	CDRSC

Figure 32. Defining TPF/APPC LUs to VTAM across a PU 5 Connection

### Defining Alias LU Names across a PU 5 CTC Connection to VTAM

When the TPF system is operating in LEN mode, you must define an alias name to VTAM for each TPF/APPC PLU; however, when the TPF system is operating in APPN mode, do not define an alias name.

To define TPF/APPC alias LU names across a PU 5 CTC connection, define the alias LU names to VTAM under the owning CDRM. Define the LU alias name for a TPF/APPC PLU as a cross domain resource with the CDRSC statement. The format is: `xxxxyz CDRSC`

Where:

`xxxx`

is the 4-character LU name.

`y` is the 1-character CPU ID.

`zz` is the 2-character qualifier number defined with the QN parameter on the OSTG CTC statement.

Figure 33 shows sample definitions for TPF/APPC PLUs and SLU threads. The alias name for a TPF/APPC PLU is used.

TPFA	CDRM
APPCA01	CDRSC
APPCA001	CDRSC
APPCA002	CDRSC
APPCA003	CDRSC
APPCA004	CDRSC
DEADA01	CDRSC
TPARA01	CDRSC

Figure 33. Defining TPF/APPC Alias LU Names to VTAM across a PU 5 CTC Connection

### Defining LUs across a PU 2.1 LEN Connection

To define a TPF/APPC LU across a PU 2.1 LEN connection, do the following:

- Define the LUs to the NCP under the NCP PU defined for TPF. Define the LU alias name for a TPF/APPC PLU with the NCP LU statement. The format is:  
**xxxxyzz LU**

Where:

**xxxx**

is the 4-character LU name.

**y** is the 1-character CPU ID.

**zz** is the 2-character qualifier number defined with the QN parameter or ALSQN parameter on the OSTG ALS statement.

Figure 34 shows sample definitions for TPF/APPC PLUs and SLU threads in 2 ALSs. The alias name for a TPF/APPC PLU is used.

P30CA4A	PU	PUTYPE=2
APPCA30	LU	LOCADDR=0
APPCA005	LU	LOCADDR=0
APPCA006	LU	LOCADDR=0
APPCA007	LU	LOCADDR=0
TPARA30	LU	LOCADDR=0
AMFMA30	LU	LOCADDR=0
P42CA4A	PU	PUTYPE=2
APPCA42	LU	LOCADDR=0
APPCA008	LU	LOCADDR=0
APPCA009	LU	LOCADDR=0
APPCA010	LU	LOCADDR=0
TPARA42	LU	LOCADDR=0
AMFMA42	LU	LOCADDR=0

Figure 34. Defining TPF/APPC LUs across PU 2.1 LEN Connections

- Define the real name of a TPF/APPC PLU to VTAM as an APPL. For information about coding APPL statements for TPF applications, see the *VTAM Network Implementation Guide*.

### Defining LUs across a PU 2.1 APPN Connection

In an APPN network, LUs do not need to be predefined to the NCP or VTAM because an APPN node has the ability to register its LUs dynamically when links are activated. LUs can be predefined or registered dynamically in an APPN network. See “LU-LU Sessions in an APPN Network” on page 211 for more information.

To define a TPF/APPC LU across a PU 2.1 APPN connection, do the following:

- Define the LUs to the NCP under the NCP PU defined for the TPF system. Define the real LU name for a TPF/APPC LU with the NCP LU statement.

Figure 35 on page 66 shows sample definitions for TPF/APPC PLUs and SLU threads in 2 ALSs.

P30CA4A	PU	PUTYPE=2
APPC	LU	LOCADDR=0
APPCA005	LU	LOCADDR=0
APPCA006	LU	LOCADDR=0
APPCA007	LU	LOCADDR=0
TPAR	LU	LOCADDR=0
AMFM	LU	LOCADDR=0
P42CA4A	PU	PUTYPE=2
APPC	LU	LOCADDR=0
APPCA008	LU	LOCADDR=0
APPCA009	LU	LOCADDR=0
APPCA010	LU	LOCADDR=0
TPAR	LU	LOCADDR=0
AMFM	LU	LOCADDR=0

Figure 35. Defining TPF/APPC LUs across PU 2.1 APPN Connections

- Define the real name of a TPF/APPC LU to VTAM as a CDRSC and specify the name of the TPF control point (CP) on the CPNAME parameter. For information about coding CDRSC statements for TPF applications, see the *VTAM Network Implementation Guide*.

## Defining TPF/APPC LUs to a VTAM Subsystem

The definition of an LU to a subsystem, such as CICS, depends on the convention used by the subsystem. TPF's LU 6.2 support follows the SNA protocol that is comparable to the LUTYPE62 as supported by CICS (see *CICS/DOS/VS Intercommunication Facilities Guide*). Figure 36 provides an example of the CICS definition for a TPF 6.2 logical unit.

DFHTCT	TYPE=SYSTEM,	REQUIRED	X
	ACCMETH=VTAM,	REQUIRED	X
	TRMTYPE=LUTYPE62,	REQUIRED	X
	FEATURE=SINGLE,	REQUIRED	X
	SYSIDNT=APPC	TPF LU NAME	X
	CONNECT=AUTO,	OPTIONAL	X
	TRMSTAT=TRANSCIVE,	DEFAULT	X
	RUSIZE=256	DEFAULT	

Figure 36. Defining a TPF/APPC LU to CICS

## Defining a TPF/APPC LU to a PS/2 Workstation (Using a Generic Name)

See the following publications for information about defining LU 6.2 communications to OS/2.

For OS/2 1.30.1 or earlier, see:

- *OS/2 Extended Edition APPC Programming Reference*
- *OS/2 Extended Edition System Administration Guide for Communications*.

For OS/2 1.30.2 or later, see:

- *Communications Manager/2 Workstation Installation and Configuration Guide*
- *Communications Manager/2 Network Administration and Subsystem Management Guide*.

---

## TPF/APPC Control Blocks

The concept of a transaction program instance is incorporated in the TPF entry control block (ECB) by storing the transaction control block identifier (TCB ID) in the reserved ECB field EBTCBID. TPF creates a new TCB ID when a remote transaction program initiates a conversation (TPF receives an ATTACH functional management header type 5 [FMH5] record).

TPF checks for an existing TCB ID when a local TPF transaction program initiates a conversation. If a TCB ID does not exist (the ECB field is zero), TPF assigns a new TCB ID to the ECB. If a TCB ID exists, the ALLOCATE request represents an additional conversation controlled by the same transaction program instance.

In addition to the TPF entry control block (ECB), TPF/APPC uses the following control blocks:

- The conversation control block (CCB), defined by data macro ICCB. (There is also a C language equivalent to ICCB, C\$ICCB, which is used for mapped conversation support.)
- The session control block (SCB), defined by data macro ISCB.

## TPF/APPC Conversation Control Block

Conversations are controlled and identified by an entry in a conversation control block (CCB). Conversation control blocks are retrieved from and returned to a pool of available CCBs. Entries are created and destroyed as conversations are allocated or deallocated: 1 entry for each conversation. The life of a CCB entry begins when a conversation starts and ends when the conversation ends.

Conversation control block entries are identified to the transaction programs by a unique conversation identifier (CCB ID). The CCB ID is the value returned by the ALLOCATE verb, an INITIALIZE verb, or assigned when an ATTACH is received. The ID is used as the value on the RESID parameter on all subsequent verb requests for this conversation.

Conversation control blocks are used to:

- Provide the anchor point for conversation input message queues
- Maintain the finite state machine status of a conversation
- Provide the area to define and control the event block necessary to suspend an ECB and wait for the posting of an event
- Identify the TPF transaction program instance (TCB ID) that controls the conversation.

Main storage for the conversation control block is allocated by the TPF initialization component (CTIN) from information provided in the SNA main storage allocation table (MSAT), which is contained in the SNA communications keypoint (CTK2, defined by data macro CK2SN). You specify the information in CTK2 with the SNAKEY macro. Conversation control blocks are written to DASD file storage by cycle-down and soft-IPL processing, and are read back into main storage by the TPF restart components.



## TPF/APPC Session Control Block

Sessions are controlled by session control blocks (SCBs). An SCB has 2 parts, SCB1 and SCB2, which are extensions of the 2 areas of the resource vector table, RVT1 and RVT2. SCBs are available only for TPF/APPC sessions (both single and parallel).

Main storage for the session control block is allocated by CTIN during TPF IPL based on the information provided in the SNA main storage allocation table.

1 SCB is initialized and allocated for each remote LU to save session limit information when:

- There is a change number of sessions (CNOS) initialize exchange
- An SCB does not already exist for that (LU,mode name) pair.

An additional SCB is allocated for each session that is brought up. All parallel session SCBs that belong to the same remote LU are chained together and anchored off the resource vector table (RVT) of the remote LU. SCBs are returned to the SCB list when sessions are deactivated.

**Note:** The SCB previously initialized to save session limit information is not returned until a CNOS RESET is done.

---

## TPF/APPC Work Blocks

TPF/APPC also uses the following work blocks:

- The TPF/APPC work block, defined by data macro IWBL
- The change number of sessions (CNOS) work block, defined by data macro ICNOS
- The partner work block, defined in data macro ICCB
- The half-session to presentation services record, defined by data macro IHPR.

## TPF/APPC Work Block

The TPF/APPC work block is a working storage block attached to data level 15 (DF) of an ECB that issued a TPF/APPC verb request. If there is a block attached to a data level required by the TPF/APPC support code, the system detaches the block from the ECB with the DETAC macro and then reattaches it with the ATTAC macro when the data level is no longer needed by the support code.

The work block is used to:

- Save the issuer's registers
- Preserve the contents of the issuing ECB and the verb parameter list
- Provide working storage for the TPF/APPC support programs.

The work block is created by the TPF/APPC support programs and released before returning control to the verb requester.

## TPF/APPC Change Number of Sessions Work Block

The TPF/APPC change number of sessions (CNOS) work block is a working storage block attached to data level 5 of an ECB that issued a CNOS verb request.

The CNOS work block is used to:

- Save the issuer's registers
- Preserve the contents of the issuing ECB and the verb parameter list
- Provide working storage for the CNOS support programs.



The CNOS work block is created by the CNOS support programs and released before returning control to the verb requester.

## **TPF/APPC Partner Work Block**

The TPF/APPC partner work block is a working storage block used for the TPF/APPC mapped conversation support. The partner work block is used to store information from the side information table between the time a conversation is initialized and actually allocated. (See “Mapped Conversations” on page 71 for a definition and explanation of the side information table.) Once the conversation is allocated, the block is released and the conversation gets the information from the LU. The information that is stored in the partner work block includes:

- A symbolic destination name
- The partner LU name
- The transaction program name
- The mode name.

## **TPF/APPC Half-Session to Presentation Services Record**

The TPF/APPC half-session to presentation services records (HPR) are 4K short-term file pool records used to buffer input messages. The TPF/APPC support package receives incoming messages from the communications source program in AMOSG format. These messages are then put on an inbound queue in HPR records. For more information about queuing, see “Inbound Message Queuing” on page 100.

---

## **Presentation Services**

Presentation services supports the TPF/APPC verb interface for both assembler language and C language. It provides the interface between the TPF transaction programs issuing the verbs and the rest of the components. The main functions include:

- Initialization of the transaction program
- Conversation verb processing
- Change number of sessions (CNOS) processing
- Finite state machine checking and control
- Activation of other components.

## **TPF Transaction Program ATTACH Interface**

Figure 37 on page 70 shows the TPF transaction program ATTACH interface.

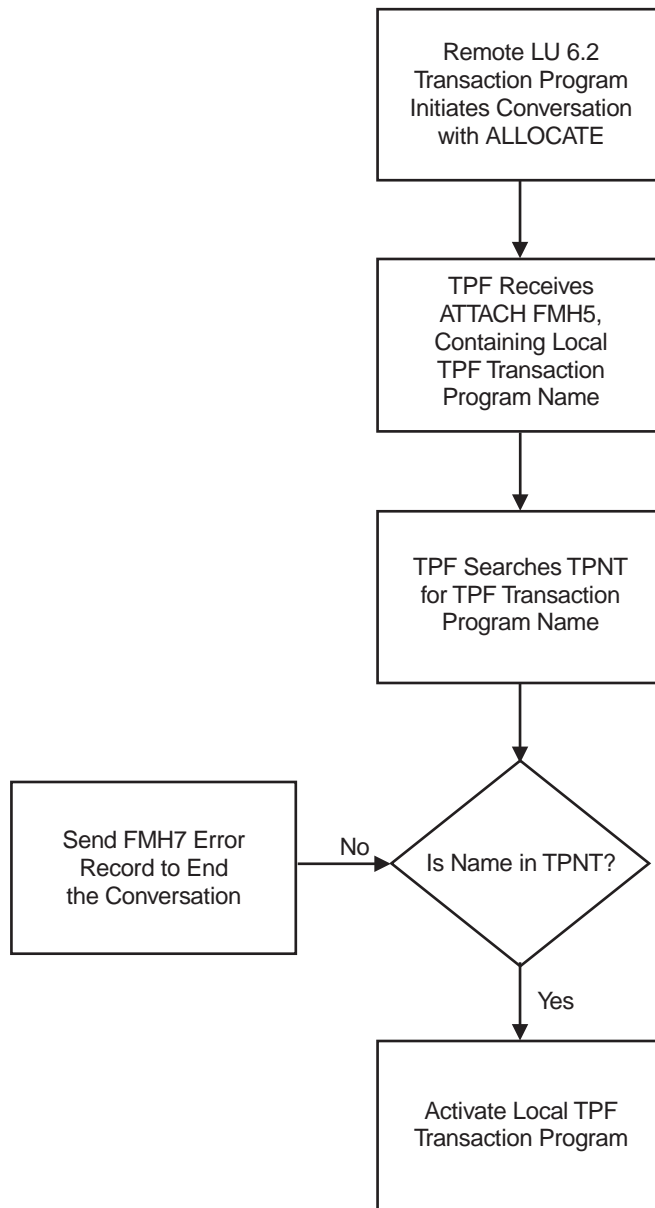


Figure 37. TPF Transaction Program ATTACH Interface

When a remote LU 6.2 transaction program initiates a conversation with a TPF transaction program by issuing an ALLOCATE verb, TPF receives an ATTACH functional management header type 5 (FMH5) record. The ATTACH FMH5 record contains the local TPF transaction program name, which is used as a search argument against the user-defined TPF transaction program name table (TPNT). If the transaction program name is **not** found in the TPNT, an FMH7 error record is sent to end the conversation. If the transaction program name is found in the TPNT, control is transferred to the TPF E-type program associated with the transaction program name. When this program is activated, the following interface is set:

- The activated transaction program is in conversation with the remote transaction program and is in receive state.
- All application registers (R0–R7) are available to the activated program. The program cannot rely on the contents of any of the registers when the transaction program is activated.

- All TPF data levels are available for use by the transaction program.
- The transaction program name received on the ATTACH is placed in the ECB beginning at EBW001. (EBW000 contains the length of the transaction program name.)

Other information carried in the ATTACH FMH5 received by TPF is **not** passed to the local TPF transaction program.

**Note:** The transaction program can issue a GET\_ATTRIBUTES or GET\_TYPE verb for additional information.

- The ECB field EBROUT contains the session control block identifier (SCB ID) of the SCB entry for the session with the remote LU that sent the ATTACH.
- The transaction control block identifier (TCB ID) is stored in the ECB field EBTCCBID.
- The conversation control block identifier (CCB ID) is stored in the ECB field EBCCBID. Use this identifier on the RESID parameter of all subsequent TPF/APPC verb macros for the conversation with the remote transaction program.

## Conversation Verbs

The conversation verbs provide program-to-program communication by means of conversations between programs. The verbs defining the conversation protocol boundary are divided into subcategories based on conversation type, as follows:

- Mapped conversation verbs
- Type-independent conversation verbs
- Basic conversation verbs.

### Mapped Conversations

The mapped conversation verbs are intended for use by application transaction programs. These verbs provide functions that are suitable for application programs written in high-level programming languages.

The TPF/APPC mapped interface is based on the communication element of SAA's Common Programming Interface (CPI). Although TPF/APPC does not fully conform to CPI Communications, standard CPI Communications programs can be converted easily, provided the programs do not use the features that TPF does not support.

Mapped conversation support is provided only through the C language interface. For additional information and details on the C language functions, see *TPF C/C++ Language Support User's Guide*. For more information about CPI Communications, see the *Systems Application Architecture Common Programming Interface Communications Reference*.

**Side Information:** For a program to establish a conversation with a partner program, TPF/APPC requires a certain amount of initialization information: the name of the partner program, the name of the LU at the partner's node, and the mode name. The TPF/APPC mapped interface provides a way to use predefined values for these required fields; these predefined values are called *side information*. The side information is accessed by a symbolic destination name and is maintained in an online database called the *side information table*. Each symbolic destination name corresponds to an entry in the side information table and contains the following 3 pieces of information:

#### Transaction program (TP) name

Specifies the name of the remote transaction program

**Partner LU name**

Is the name of the LU where the remote transaction program is located

**Mode name**

Is used by TPF/APPC to designate the properties for the session that will be allocated for the conversation.

When a transaction program requests the initiation of a conversation and specifies a symbolic destination name, the TPF/APPC mapped interface retrieves the corresponding entry from the side information table and uses the data to initialize the conversation characteristics *mode\_name*, *mode\_name\_length*, *partner\_LU\_name*, *partner\_LU\_name\_length*, *TP\_name*, and *TP\_name\_length*.

The side information table offline program (CHQI) provides a way of generating the side information table, verifying the syntax of the entries before bringing them online, and managing the data contained in the online side information table. You can also modify the online side information table using the ZNSID command. For details on how to run CHQI, see “Generating the Side Information Table for Mapped Conversations” on page 83. For details on the format of the ZNSID command, see *TPF Operations*.

**Type-Independent Conversation Verbs**

The type-independent conversation verbs are intended for use with both mapped and basic conversations. These verbs provide functions that span both conversation types.

TPF supports the following type-independent conversation verbs:

- GET\_TYPE
- WAIT.

**Basic Conversation Verbs**

The basic conversation verbs are intended for use by application transaction programs and LU service programs.

The TPPCC macro provides the assembler language interface for basic conversations. For more information, see *TPF General Macros*. The *tppc\_* function calls provide the C language interface for basic conversations. For more information, see the *TPF C/C++ Language Support User's Guide*.

TPF supports the following basic conversation verbs:

- ACTIVATE\_ON\_CONFIRMATION<sup>2</sup>
- ACTIVATE\_ON\_RECEIPT<sup>2</sup>
- ALLOCATE
- CONFIRM
- CONFIRMED
- DEALLOCATE
- FLUSH
- GET\_ATTRIBUTES
- POST\_ON\_RECEIPT
- PREPARE\_TO\_RECEIVE
- RECEIVE
- REQUEST\_TO\_SEND
- SEND\_DATA
- SEND\_ERROR

---

2. This verb is a TPF extension to the LU 6.2 architecture.

- TEST.

### Architecture and TPPCC Macro Comparison

The following figures compare the interface defined by the LU 6.2 architecture with the interface provided through the TPPCC macro. The C language interface is not shown but it is similar to the TPPCC macro except that all C language function calls use positional parameters instead of keyword parameters. (See *TPF General Macros* and the *TPF C/C++ Language Support User's Guide* for details about the TPF/APPC macros and functions.)

**Note:** In each figure, there is a line that separates the parameters that are passed to the processing components from the parameters that are returned. The passed parameters are shown above the line and the returned parameters are shown below the line. Parameters shown between 2 lines are both passed and returned.

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
ALLOCATE	LU_NAME MODE_NAME TPN TYPE RETURN_CONTROL SYNCH_LEVEL SECURITY PIP	ALLOCATE,	LUNAME= MODE= TPN= TYPE= RCONTROL= SYNC= SECURITY=NONE PIP=NO
	RESOURCE RETURN_CODE		RESID= RCODE=

Figure 38. ALLOCATE Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
CONFIRM	RESOURCE	CONFIRM,	RESID=
	RETURN_CODE REQUEST_TO_SEND_RECEIVED		RCODE= RTSRCVD=

Figure 39. CONFIRM Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
CONFIRMED	RESOURCE	CONFIRMED,	RESID=
			RCODE=

**Note:** RCODE returns state checks and parameter checks.

Figure 40. CONFIRMED Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
DEALLOCATE	RESOURCE TYPE LOG_DATA	DEALLOCATE,	RESID= TYPE= LOGDATA=NO
	RETURN_CODE		RCODE=

Figure 41. DEALLOCATE Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
FLUSH	RESOURCE	FLUSH,	RESID=
			RCODE=

**Note:** RCODE returns state errors and parameter checks.

Figure 42. FLUSH Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
GET_ATTRIBUTES	RESOURCE	GET_ATTRIBUTES,	RESID=
	OWN_LU_NAME PARTNER_LU_NAME MODE_NAME SYNCH_LEVEL SECURITY_USER_ID SECURITY_PROFILE LUW_IDENTIFIER CONVERSATION_CORRELATOR		OWNAME= PLUNAME= MODE= SYNC=  RCODE=

**Note:** RCODE returns state checks and parameter checks.

Figure 43. GET\_ATTRIBUTES Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
GET_TYPE	RESOURCE	GET_TYPE,	RESID=
	TYPE RETURN_CODE		TYPE= RCODE=

Figure 44. GET\_TYPE Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
POST_ON_RECEIPT	RESOURCE FILL LENGTH	POST_ON_RECEIPT,	RESID= FILL=LL LENGTH=
	RETURN_CODE		RCODE=

**Note:** RCODE returns state checks and parameter checks.

Figure 45. POST\_ON\_RECEIPT Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
PREPARE_TO_RECEIVE	RESOURCE TYPE LOCKS	PREPARE_TO_RECEIVE,	RESID= TYPE= LOCKS=SHORT
	RETURN_CODE		RCODE=

Figure 46. PREPARE\_TO\_RECEIVE Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
RECEIVE_AND_WAIT	RESOURCE FILL	RECEIVE,	RESID= FILL=LL WAIT=YES
	LENGTH		LENGTH=
	WHAT_RECEIVED DATA REQUEST_TO_SEND_RECEIVED RETURN_CODE		WHATRCV= DATA= RTSRCVD= RCODE=

Figure 47. RECEIVE Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
REQUEST_TO_SEND	RESOURCE	REQUEST_TO_SEND,	RESID=
			RCODE=

Figure 48. REQUEST\_TO\_SEND Verb



LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
SEND_DATA	RESOURCE DATA LENGTH	SEND_DATA,	RESID= DATA= LENGTH=
	RETURN_CODE REQUEST_TO_SEND_RECEIVED		RCODE= RTSRCVD=

Figure 49. SEND\_DATA Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
SEND_ERROR	RESOURCE TYPE LOG_DATA	SEND_ERROR,	RESID= TYPE= LOGDATA=NO
	RETURN_CODE REQUEST_TO_SEND_RECEIVED		RCODE= RTSRCVD=

Figure 50. SEND\_ERROR Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
TEST	RESOURCE TEST	TEST,	RESID= TEST=
	RETURN_CODE		RCODE=

Figure 51. TEST Verb

LU 6.2 Architecture		TPPCC Macro	
Verb	Parameter	Verb	Keyword
WAIT	RESOURCE_LIST	WAIT,	RESIDL=
	RESOURCE_POSTED RETURN_CODE		RESPSTD= RCODE=

Figure 52. WAIT Verb

## Change Number of Sessions

The change number of sessions (CNOS) verbs change the (LU,mode) session limit, which controls the number of LU-LU sessions per mode name that are available between 2 LUs for allocation to conversations.

The 2 LUs may cooperate in the execution of the CNOS verbs by means of a CNOS request and CNOS reply. The LU executing the control operator transaction program sends a CNOS request to the partner LU. The partner LU starts an SNA service transaction program (the CNOS service transaction program), which causes the partner LU to process the CNOS request and send back a CNOS reply.

### Change Number of Sessions Components

The following is a description of the individual components for TPF's change number of sessions (CNOS) support.

#### TPF control operator transaction program (COPR.TP)

Serves as an interface between the TPF change number of sessions command (ZNCNS) and the presentation services for control operator (PS.COPR).

TPF COPR.TP is activated when the operator issues a ZNCNS command. Based on the parameters passed by ZNCNS, TPF COPR.TP issues a CNOSC macro to communicate with the remote LU's CNOS service transaction program (CNOS.TP). The macros that TPF COPR.TP can issue are:

- CNOSC RESET
- CNOSC INITIALIZE
- CNOSC CHANGE.

**Note:** You can create your own COPR.TP to issue CNOS requests. There can be as many COPR.TPs in the system as you need.

#### CNOS service transaction program (CNOS.TP)

Is a reserved SNA-defined transaction program in TPF. This program serves as an interface between the remote control operator transaction program and the TPF PS.COPR. When the remote LU makes a CNOS request to TPF, the TPF CNOS.TP is activated, and in turn, activates the PS.COPR to perform the PROCESS\_SESSION\_LIMIT function.

When control returns from PS.COPR, the conversation with the source LU (in this case, the remote LU) has already been deallocated and the session limit parameters updated for the target LU (in this case, the local LU). A

message that notifies the operator that the process is complete and displays the new session limits is then sent to the TPF operator console.

### **Presentation service for the control operator (PS.COPR)**

Serves as an interface between (1) the TPF COPR.TP and the remote CNOS.TP or (2) the remote COPR.TP and the TPF CNOS.TP. PS.COPR provides services for each TPF-supported change number of sessions verb.

Control is passed to PS.COPR when COPR.TP issues CNOSC macros or from CNOS.TP to perform the PROCESS\_SESSION\_LIMIT function. When receiving change number of sessions requests, TPF PS.COPR issues TPF/APPC application verbs to communicate with the target LU. After processing those verbs, TPF PS.COPR interacts with the resource manager to request session changes. Control returns to the calling COPR.TP or CNOS.TP after the services are complete.

### **Managing Session Limits**

To understand the applications' options in managing session limits, you need an overview of how TPF processes change number of sessions (CNOS) requests.

The process of changing session limits involves the exchange of session limits between the 2 control operators. TPF does not negotiate the session limits; however, if TPF initiates the CNOS exchange, the remote LU can negotiate the limits. TPF accepts the remote control operator's session limits and processes them. The control operator that initiated the CNOS request is informed of the new limits in the reply to the CNOS request.

You can write your own COPR.TP application that issues CNOS requests with the CNOSC macros. There can be only 1 CNOSC macro outstanding at a time for a particular partner LU and mode name combination. A CNOSC macro must complete before another application can issue a CNOS request for the same partner and mode name combination. An application can have multiple outstanding CNOSC macros specifying different LUs or the same LU with a different mode name.

**Setting Session Limits to Zero:** The CNOSC RESET macro and ZNCNS RESET command both provide the ability to set the session limit to zero. Setting session limits to zero deactivates sessions using the specified mode name and LU. Sessions being used by a conversation are not deactivated until the conversation finishes.

Setting session limits to zero is part of the orderly termination of an LU 6.2 application, or consider setting the session limits to zero to deactivate sessions that are active but not presently needed.

If the application requests a session limit of zero, the session limit cannot be negotiated; the partner LU must accept the zero limit. When the application sets session limits to zero, it can specify that the change is to apply to 1 of the following:

- 1 specific mode name
- All mode names with the partner LU except SNASVCMG
- The SNASVCMG mode name only.

When making a CNOSC RESET request, the application can also specify whether to drain (or honor) all the queued ALLOCATE requests (those waiting for a session) before deactivating sessions or whether to reject the requests immediately.

The partner LU cannot negotiate the draining capability of the application making the CNOS request; it must accept whatever the application specifies. The partner LU can negotiate the draining capability for its side to indicate that it will not allow draining on the target side.

**Managing Session Activation and Deactivation:** Change number of sessions requests frequently cause sessions to be activated or deactivated. TPF activates additional sessions when a transaction program requests an ALLOCATE, there are no sessions free, and the session limit has not been reached. If the CNOS request lowers the session limits, TPF deactivates sessions to meet the new limits. TPF only deactivates sessions when conversations are no longer using those sessions and TPF contains the LU responsible for deactivating sessions.

## Finite State Machine

The LU 6.2 architecture uses a finite state machine as a method of defining a limited (finite) number of states (current status) so that only certain actions can take place. These actions can then change the current state. In general, actions can take place only when the finite state machine is in certain states, and the execution of the requested action can then cause the state of the finite state machine to change.

The TPF/APPC support implements a logical conversation finite state machine (FSM) to control the sequence in which verbs are issued for a conversation.

**Note:** TPF/APPC support implements 2 other types of finite state machines, which are not discussed in detail. 1 maintains the status of posting, and the other maintains the status of errors and failures.

The TPF/APPC support package defines the following conversation states:

State	Definition
Reset (0)	The state in which a new conversation can be allocated.
Send (1)	The state in which a conversation can send data, status, or a confirmation request.
Receive (2)	The state in which a conversation can receive data, status, or a confirmation request from the remote partner.
Received Confirm (3)	A conversation that was in receive state received a confirmation request.
Received Confirm Send (4)	A conversation received a send request with a confirmation request.
Received Confirm Deallocate (5)	A conversation received a deallocate request with a confirmation request.
Pending Deallocate (8)	A conversation issued a deallocate request with a confirmation request.
End Conversation (9)	The state in which a conversation cannot continue.
<b>Note:</b> TPF does not support the states Prepare to Receive Defer (6) and Deallocate Defer (7) defined in the LU 6.2 architecture.	

When the local TPF transaction program issues a TPF/APPC verb request or when the remote transaction program sends data, requests, or status messages to the local TPF transaction program, an action or an event results. The state of the

conversation controls the events or actions. For example, the local TPF transaction program is allowed to issue a TPF/APPC CONFIRMED verb only after it has received a confirmation request from the remote transaction program. The TPF/APPC FSM checks for the validity of all the requested actions and maintains the current state of each conversation. Invalid requests cause a state check. If the state check is due to the local TPF transaction program's issuing a TPF/APPC verb, the return code parameter (RCODE) contains the state check. If the state check is due to the remote transaction program's sending invalid data, status, or confirmation, the session is unbound, and the local transaction program is notified when it issues its next TPF/APPC verb.

The following table shows the definition of the individual actions or events. The first column contains an indication of whether the action was sent (S) or received (R) by the local TPF transaction program (TP).

Action or Event	Description
S, Allocate	The local TPF TP issues the ALLOCATE verb.
R, Attach	The local TPF TP is activated when it receives an ATTACH header from a remote TP that issued an ALLOCATE request for the local TPF TP.
S, Send_Data	The local TPF TP issues the SEND_DATA verb.
S, Prep_to_Rvc_Flush	The local TPF TP issues the PREPARE_TO_RECEIVE verb with the FLUSH option.
S, Prep_to_Rvc_Confirm	The local TPF TP issues the PREPARE_TO_RECEIVE verb with the CONFIRM option.
S, Flush	The local TPF TP issues the FLUSH verb.
S, Confirm	The local TPF TP issues the CONFIRM verb.
S, Send_Error	The local TPF TP issues the SEND_ERROR verb.
S, Receive_and_Wait	The local TPF TP issues the RECEIVE verb.
S, Post_on_Receipt	The local TPF TP issues the POST_ON_RECEIPT verb.
S, Wait	The local TPF TP issues the WAIT verb.
S, Test (posted)	The local TPF TP issues the TEST verb with the TEST=POSTED option.
S, Test (RTSR)	The local TPF TP issues the TEST verb with the TEST=RTSRCVD option.
S, Request_to_Send	The local TPF TP issues the REQUEST_TO_SEND verb.
R, Send_Indicator	The local TPF TP in <u>receive</u> state is placed in <u>send</u> state when the remote TP enters <u>receive</u> state.
R, Confirm_indicator	The local TPF TP receives a confirmation request when it is in <u>receive</u> state.
R, Confirm_Send_Ind	The local TPF TP receives a confirmation request with the SEND indication when it is in <u>receive</u> state.
R, Confirm_Dealloc_Ind	The local TP receives a confirmation request with the deallocation indication.
S, Confirmed	The local TPF TP issues the CONFIRMED verb.
R, Program_Error_RC	The local TPF TP receives an indication that the remote TP issued a SEND_ERROR verb because of a program failure.
R, Service_Error_RC	The local TPF TP receives an indication that the remote TP issued a SEND_ERROR verb because of a service failure.
R, Dealloc_Normal_RC	The local TPF TP receives an indication that the remote TP issued a DEALLOCATE verb with the FLUSH option.
R, Dealloc_Abend_RC	The local TPF TP receives an indication that the remote TP issued a DEALLOCATE verb with 1 of the ABEND options.
R, Resource_Failure_RC	The local TPF TP receives an indication that a conversation failure occurred.
R, Alloc_Error_RC	The local TPF TP receives an indication that a conversation allocation request failed.

Action or Event	Description
S, Dealloc_Flush	The local TPF TP issues the DEALLOCATE verb with the FLUSH option.
S, Dealloc_Confirm	The local TPF TP issues the DEALLOCATE verb with the CONFIRM option.
S, Dealloc_Abend	The local TPF TP issues the DEALLOCATE verb with 1 of the ABEND options.
S, Dealloc_Local	The local TPF TP issues the DEALLOCATE verb with the LOCAL option.
R, Dealloc_Pend_Confirm	The local TPF TP receives the confirmation reply while waiting in response to a DEALLOCATE verb with the CONFIRM option.
S, Get_Attribute	The local TPF TP issues the GET_ATTRIBUTES verb.
S, Get_Type	The local TPF TP issues the GET_TYPE verb.

The TPF/APPC conversation FSM is shown in Table 2. The columns of the matrix show the defined states, and the rows show the individual actions or events. The number in each box indicates the new state of the conversation following the completion of the event. An empty box shows that the condition is not allowed, and, if encountered, a state check results.

Table 2. TPF/APPC Conversation Finite State Machine Matrix

Actions or Events	States							
	Reset (0)	Send (1)	Receive (2)	Received Confirm (3)	Received Confirm Send (4)	Received Confirm Deallocate (5)	Pending Deallocate (8)	End Conversation (9)
S, Allocate	1							
R, Attach	2							
S, Send_Data		1						
S, Prep_to_Rvc_Flush		2						
S, Prep_to_Rvc_Confirm		2						
S, Flush		1						
S, Confirm		1						
S, Send_Error		1	1	1	1	1		
S, Receive_and_Wait		2	2					
S, Post_on_Receipt			2					
S, Wait			2					
S, Test (posted)			2					
S, Test (RTSR)		1	2					
S, Request_to_Send		1	2	3	4	5		
R, Send_Indicator			1					
R, Confirm_Indicator			3					
R, Confirm_Send_Ind			4					
R, Confirm_Dealloc_Ind			5					
S, Confirmed				2	1	9		
R, Progrm_Error_RC		2	2				2	
R, Service_Error_RC		2	2				2	
R, Dealloc_Normal_RC		9	9					
R, Dealloc_Abend_RC		9	9				9	
R, Resource_Failure_RC		9	9				9	
R, Alloc_Error_RC		9	9				9	
S, Dealloc_Flush		0						

Table 2. TPF/APPC Conversation Finite State Machine Matrix (continued)

Actions or Events	States							
	Reset (0)	Send (1)	Receive (2)	Received Confirm (3)	Received Confirm Send (4)	Received Confirm Deallocate (5)	Pending Deallocate (8)	End Conversation (9)
S, Dealloc_Confirm		8						
S, Dealloc_Abend		0	0	0	0	0		
S, Dealloc_Local								0
R, Dealloc_Pend_Confirm							0	
S, Get_Attribute		1	2	3	4	5		9
S, Get_Type		1	2	3	4	5		9
<b>Note:</b> TPF does not support the states Prepare to Receive Defer (6) and Deallocate Defer (7) defined in the LU 6.2 architecture for the SYNCH_POINT options.								

## Generating the Side Information Table for Mapped Conversations

The side information table offline program (CHQI) runs under MVS or CMS to generate either a tape or a general data set that can be loaded to a TPF system. The generated data set contains commands and data for modifying the side information table. Data from the side information table is retrieved whenever an application transaction program issues the mapped conversation initialize conversation (cminit) function.

**Note:** Before generating the side information table, be sure that you have defined the #CMSIT fixed file records at system generation time. These are the 4K records used to hold the side information table.

Figure 53 on page 84 shows the process for generating a side information table.

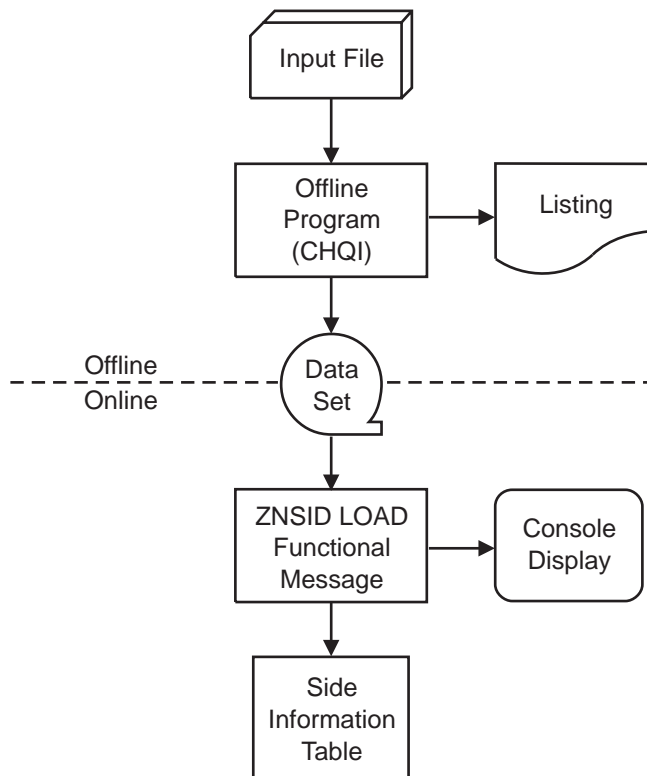


Figure 53. Generating the Side Information Table

The steps required for generating a side information table are as follows:

1. On MVS or CMS, create an input file for CHQI.

The input to CHQI is an EBCDIC file of fixed-length 80-byte records, consisting of statements that specify the function to be performed. You can also include comments and blank lines to improve readability. See “Creating the Input File” on page 85 for details.

2. On MVS or CMS, run CHQI against the input file.

CHQI creates 2 output files:

- A tape or general data set that can be loaded online by TPF
- A listing describing the results of the conversion on the input file to the tape or general data set.

See “Running CHQI” on page 92 and “Side Information Table Offline Program Output” on page 93 for details.

3. Use the listing file to correct any errors in the input file.
4. Repeat steps 2 and 3 until there are no errors in the input file.
5. Mount the tape or general data set on the TPF system.
6. If the side information table is not initialized, issue the ZNSID INITIALIZE command.

**Note:** Initializing the side information table erases any existing information in it.

7. Load the side information data set to the TPF system using the ZNSID LOAD command.

Steps 5, 6, and 7 are described in more detail in “Loading the Side Information Data Set to TPF” on page 96.



In the following discussion, the phrase *side information table* refers to the online side information table residing on TPF. The phrase *side information data set* refers to the data set that is generated offline by CHQI and contains the entries for the online side information table.

## Creating the Input File

The following describes the basic steps for creating the input file for CHQI. Additional details on the syntax requirements for the statements are described in the following section, "Statement Syntax" on page 86.

1. Create a file with a logical record length of 80 (LRECL=80) and a fixed record format (RECFM=FB).
2. You can code comments (lines with an asterisk in column 1) and blank lines (containing only spaces) anywhere in the file.
3. If the side information data set must be loaded to a specific subsystem, code:

**LOAD SS-*subsystem***

where *subsystem* is the 1- to 4-character name of the subsystem on which the data is to be loaded. If you specify a subsystem, the side information data set must be loaded to that subsystem. If you do not code a LOAD statement, the side information data set can be loaded to any subsystem. After the side information data set is loaded, the symbolic destination names defined by that data set are available to transaction programs running in the specified subsystem. You can specify only 1 subsystem for a side information data set; that is, only 1 LOAD statement can be included in the input file. This statement, if included, must precede all other statements.

4. If you want to display text on the TPF operator console while loading the side information data set, for each line of text to be displayed code:

**DESCR *text***

where *text* is the line of text to be displayed. The text is displayed as a response to the ZNSID LOAD command.

5. If you want to add an entry to the side information table, for each entry code:

**ADD NAME-*sdn* TP-*tpn* LU-*lun*MODE-*mode***

where *sdn* is the symbolic destination name of the new entry, *tpn* is the corresponding remote transaction program name, *lun* is the corresponding partner LU name, and *mode* is the corresponding mode. If an entry with the identical symbolic destination name already exists in the side information table, it is changed by the entry in the side information data set. If 2 or more entries with the same symbolic destination name are included in the same side information data set, the entries are loaded in the order that they are specified in the input file. After the load is complete, the side information table entry for the duplicated symbolic destination name contains the data specified for the last entry loaded.

6. If you want to remove entries from the side information table, for each entry to be removed code:

**REMOVE NAME-*sdn***

where *sdn* is the symbolic destination name of the entry to be removed. If no entry with the specified symbolic destination name exists in the table, the entry in the side information data set is ignored.

## Statement Syntax

The CHQI input syntax consists of the following elements:

### Comments

Any record beginning with an asterisk in column 1 is a comment. The text in columns 2–80 is not parsed. You can code a comment line anywhere in the file.

### Verbs

Each statement begins with a verb, and each type of statement requires a different set of parameters following the verb. There are no abbreviations for the verbs; you must code each 1 in full. CHQI recognizes 4 verb statements: LOAD, DESCR, ADD, and REMOVE.

### Parameters

The DESCR verb takes a positional parameter (the text that follows it); the other verbs all take keyword parameters. Keyword parameters consist of the keyword, followed immediately by either a single dash or by a single equal sign, followed by the value. There can be 1 or more spaces between the dash or equal sign and the value, as long as the value is in the same record as the keyword. There must not be any spaces between the keyword and the dash or equal sign.

The input file is free format in that you do not have to code verbs and parameters on a particular line or in a particular column. Blank lines can be inserted at any point to improve readability. Exceptions to this free format include the following:

- Comments must begin in column 1.
- Each parameter must be completed in the same record that it begins; that is, a keyword and its associated value must be on the same line.
- The DESCR verb takes 1 parameter that includes all of the text from the DESCR verb to the end of the record.
- The input file must be in uppercase, except for:
  - Text in comments
  - Text in DESCR parameters
  - TP parameter values; these must match the transaction program names defined in your network.

CHQI's parser scans the input file, strips out comments, and parses the remaining text into tokens. A token is any string of characters that is delimited by a space, null character, comma<sup>3</sup>, dash, equal sign, or logical end of record and that does not contain a space, null character, dash, equal sign or logical end of record<sup>4</sup>.

Tokens are classified as verbs, parameter keywords, and parameter values. Each verb begins a new statement, and each statement consists of the verb that initiates it and any following parameters up to the next verb or the end of file. The statements are checked for errors. Any statement containing no error messages or only attention messages is written to the side information data set in the proper

---

3. Commas have no meaning other than to separate tokens and are exactly equivalent to blank characters.

4. The parameter for the DESCR verb is an exception to this. DESCR's parameter begins with the first nonblank character following the verb and continues to the last nonblank character before the next logical end of record. If there is no nonblank character between the DESCR verb and the next end of record, the parameter is a null string.

format to be read by the TPF side information data loader. Statements containing 1 or more errors are not written to the side information data set.

The following shows the format of the statements and describes the verbs and parameters.

#### **LOAD SS-*subsystem***

##### **LOAD**

Must be the first statement in the file, if used. If the LOAD statement occurs following any other statement, the LOAD statement is flagged with an error. The LOAD statement requires the SS parameter; if SS is omitted, the LOAD statement is flagged with an error.

##### **SS-*subsystem***

Specifies the subsystem on which the generated side information data must be loaded. SS can be abbreviated to S.

The SS parameter value is any string from 1 to 4 characters long. The subsystem name is copied to the header of the first record in the side information data set.

#### **DESCR *text***

##### **DESCR**

Can be coded anywhere except before a LOAD statement.

##### ***text***

Represents a positional parameter that consists of the text between the DESCR verb and the next logical end of record, stripped of any leading or trailing blanks. There are no restrictions on the characters or strings that you can use, assuming the characters you use can be handled by your console.

#### **ADD NAME-*sdn* TP-*tpn* LU-*lun* MODE-*mode***

##### **ADD**

Can be coded anywhere except before a LOAD statement. Each ADD statement takes 4 keyword parameters: NAME, TP, LU, and MODE. You can code these parameters in any order. If the NAME parameter is omitted, the statement is flagged with an error. If the TP, LU, or MODE parameter is omitted, the statement is flagged as requiring attention and the corresponding field in the side information entry is set to a null value.

**Note:** Transaction programs referring to an entry containing a null field must set valid values for the *mode\_name*, *mode\_name\_length*, *partner\_LU\_name*, *partner\_LU\_name\_length*, *TP\_name*, and *TP\_name\_length* conversation characteristics before successfully allocating a conversation. See the *cmsmn*, *cmstp1n*, and *cmstp2n* functions in the *TPF C/C++ Language Support User's Guide* for additional information.

If any of these parameters is repeated in a single ADD statement, it is flagged as an error. (Successive TP parameters, which are concatenated, are treated as a single instance of the TP parameter.)

##### **NAME-*sdn***

Specifies the symbolic destination name of the new side information table entry. NAME can be abbreviated to NAM, NA, or N.

The NAME parameter value is any string from 1- to 8-characters long, consisting of any characters except token delimiters (dash, equal sign, comma, or blank) and lowercase letters. The symbolic name written to the side information data set is an 8-byte field consisting of the NAME parameter value padded to the right with space characters.

#### TP-tpn

Specifies the remote transaction program name associated with the symbolic destination name. TP can be abbreviated to T.

SNA allows transaction program names to include unprintable characters. To allow entry of such names, the TP parameter value can include substrings that are translated to unprintable characters in the output side information entry. The generated transaction program name is a variable-length field from 1 to 64 bytes long with no padding.

The dollar sign (\$, X'5B') is used to delimit substrings of TP parameter values that are to be interpreted as hexadecimal values, according to the following rules:

- Pairs of hexadecimal digits entered between 2 single dollar signs (\$) are converted to the corresponding binary values; for example, T-\$07F0\$ sets the transaction program name to X'07F0'.
- Hexadecimal substrings can be freely intermixed with literal substrings in the same TP specification, as in T-A\$01\$BCD\$020304\$E\$05\$, which sets the transaction program name to X'C101C2C3C4020304C505'.
- If a dollar sign is actually desired in the transaction program name, 2 consecutive dollar signs (\$\$) must be used for each dollar sign needed. For example, T-X\$\$Y\$\$Z sets the transaction program name to "X\$Y\$Z", and T-\$\$\$\$32\$F1\$\$F0\$\$\$ sets the transaction program name to "\$\$\$3210\$".
- The transaction program name cannot include blank characters (X'40'); therefore, a parameter of T-123\$40\$ABC is flagged as an error.
- The length of the *converted* TP parameter value must be between 1 and 64 bytes.

To accommodate the possibility of a TP parameter value being too long to fit into 1 80-byte record, the values of adjacent TP parameters are concatenated. Each of the parameter values must conform to the rules stated previously, and the total length of all the concatenated converted strings must be less than or equal to 64. No other parameters can be coded between 2 concatenated TP parameters.

**Note:** N1 of the other keyword parameters are concatenated in this way.

The following example shows adjacent TP parameters that are concatenated to produce 1 transaction program name value for the generated entry:

```
*****
* ADD STATEMENT WITH CONCATENATED TP PARAMETERS. *
* THE GENERATED TRANSACTION PROGRAM NAME IS:      *
*   HEX      : C2C5C7C9 D5010203 5BC5D5C4          *
*   EBCDIC   : B E G I N . . . $ E N D             *
*****
ADD      N-CONCAT
          T-BEGIN
          T-$010203$
          T-$$
          T-END
          ...
```

**LU-*lun***

Specifies the partner LU name associated with a symbolic destination name. LU can be abbreviated to L.

The LU parameter value is a string from 1 to 17 characters long, consisting of an LU name and an optional network ID. If the network ID is included, it must precede the LU name and be separated from it with a period. If you only include the LU name, a period is not required.<sup>5</sup> Both the network ID and the LU name must be from 1 to 8 characters long, must contain only capital letters (A–Z) and numeric digits (0–9), and must begin with a capital letter. Any LU parameter that does not satisfy these requirements is flagged as an error. The following are examples of valid LU parameter values:

```
NETID.LUNAME
LUNMONLY
N112358D.L1234567
N.L
LU62MAPD
```

The partner LU name field written to the side information data set is a variable-length field from 1 to 17 bytes long. The LU parameter value is copied to it without conversion or padding.

**MODE-*mode***

Specifies the mode associated with a symbolic destination name. MODE can be abbreviated to MOD, MO, or M.

The MODE parameter value is a string from 1 to 8 characters long. It must contain only capital letters (A–Z) and numeric digits (0–9), and must begin with a capital letter. Any MODE parameter value that does not satisfy these requirements is flagged as an error. The mode field written to the side information data set is a variable-length field from 1 to 8 bytes long. The MODE parameter value is copied to it without conversion or padding.

**REMOVE NAME-*sdn*****REMOVE**

Can be coded anywhere except before a LOAD statement. Each REMOVE statement requires the NAME parameter. If the NAME parameter is omitted or repeated in the REMOVE statement, the statement is flagged with an error. The TP, LU, and MODE parameters are not required for this statement. Each occurrence of 1 of these parameters is flagged as requiring attention, and the parameter value is ignored.

**NAME-*sdn***

Specifies the symbolic destination name of the side information table entry to be removed. Refer to the ADD statement parameter description for additional information about this parameter.

**CHQI Input File Example**

Figure 54 on page 90 shows an example of a CHQI input file.

5. For consistency with TPF commands, LU parameters specified as *.luname*; that is, no network ID but a leading period before the LU name are accepted. The leading period is not included in the side information data set or in the online side information table.

```

*****
* THIS CHQI INPUT FILE GENERATES A TPF-LOADABLE SIDE INFORMATION DATA SET THAT *
* REMOVES SOME OBSOLETE SIDE INFORMATION TABLE ENTRIES AND ADDS SOME NEW 1S. *
*****

LOAD S-BSS

*****
* IF CODED, THE LOAD STATEMENT MUST BE THE FIRST STATEMENT IN THE INPUT FILE. *
* COMMENTS AND BLANK LINES CAN PRECEDE THE LOAD STATEMENT. *
*****

DESCR =====
DESCR = REVISIONS TO SIDE INFORMATION TABLE      =
DESCR = CREATED 01/30/91                          =
DESCR =====

*****
* THE FOLLOWING ENTRIES ARE OBSOLETE AND WILL BE REMOVED *
*****
REMOVE N-OBSDEST1
REMOVE N-OBSDEST2
REMOVE N-OBSDEST3

DESCR =====
DESCR = OBSOLETE ENTRIES HAVE BEEN DELETED        =
DESCR =====

*****
* THE FOLLOWING NEW ENTRIES WILL BE ADDED *
*****
ADD   N-NEWDEST1  T-TPNAME1  L-SNANET.LU62MAP1  M-MODE1
ADD   N-NEWDEST2  T-TPNAME2  L-SNANET.LU62MAP2  M-MODE2
ADD   N-NEWDEST3  T-TPNAME3  L-SNANET.LU62MAP3  M-MODE3

DESCR =====
DESCR = NEW ENTRIES HAVE BEEN ADDED                =
DESCR =====

```

Figure 54. Example of an Input File for CHQI

Figure 55 on page 91 shows how the TPF operator console looks after loading the side information data set generated from Figure 54.

```

NSID0017I 01:23:45 USER TEXT FROM SIDE INFORMATION GDS
=====
= REVISIONS TO SIDE INFORMATION TABLE           =
= CREATED 01/30/91                               =
=====
--- END OF DESCRIPTION ---

NSID0017I 01:23:45 USER TEXT FROM SIDE INFORMATION GDS
=====
= OBSOLETE ENTRIES HAVE BEEN DELETED             =
=====
--- END OF DESCRIPTION ---

NSID0017I 01:23:46 USER TEXT FROM SIDE INFORMATION GDS
=====
= NEW ENTRIES HAVE BEEN ADDED                     =
=====
--- END OF DESCRIPTION ---

NSID0005I 01.23.46 NEW SIDE INFORMATION TABLE ENTRIES LOADED
ENTRIES PROCESSED - 6
ENTRIES ADDED     - 3
ENTRIES CHANGED   - 0
ENTRIES REMOVED   - 3
ERRORS DURING LOAD - 0

```

Figure 55. TPF Console Display. This shows the result of loading the side information data set generated using the input in Figure 54 on page 90.

## CHQI Input Statements and ZNSID Commands

The syntax of the CHQI input file is similar to the syntax of the ZNSID command. The main differences between CHQI input statements and ZNSID commands are as follows:

- There is no CHQI equivalent of the ZNSID DISPLAY, INITIALIZE, LOAD, or HELP commands.
- For entries in the side information table with the same symbolic destination name as an entry in a side information data set, the CHQI ADD statement works the same as the ZNSID CHANGE command.

If the NAME parameter of a ZNSID ADD command matches the symbolic destination name of an entry in the side information table, the command is rejected. If the NAME parameter of a CHQI ADD statement matches the symbolic destination name of an entry already in the side information table when the side information data set is loaded, the entry loaded from the side information data set **replaces** the online entry.

**Note:** Only those values that are coded in the input file are replaced.

Consider, for example, there is an entry in the table for a symbolic destination name of SYMDEST0. An input file contains the statement:

**ADD N-SYMDEST0 M-MODE0**

(the LU and TP parameters are not specified). When the input file is processed, attention messages are generated stating that the LU and TP parameters are missing. After the resulting data set is loaded to TPF, the entry for SYMDEST0 contains the value for MODE which was specified by the input file; the values for LU and TP remain as they were before the new data was loaded.

- The CHQI LOAD verb specifies a particular subsystem on which the side information data set must be loaded. There is no equivalent ZNSID command; the ZNSID LOAD command is used to specify which side information data set is to be loaded and to perform the loading of side information entries from that data set.
- CHQI does not require or accept the token ZNSID preceding its verbs.
- ZNSID allows abbreviation of some verbs, CHQI does not.
- The ZNSID ADD command requires the NAME parameter and at least 1 of the 3 parameters TP, LU, and MODE (this is done mainly to allow entering of values too long to fit on 1 input line using successive commands). The CHQI ADD statement **requires** only the NAME parameter.
- CHQI concatenates adjacent TP parameter values. ZNSID rejects more than 1 TP parameter per command.

## Running CHQI

CHQI consists of a single CSECT that can be assembled, link-edited, and run under either MVS or CMS. There are no runtime parameters for CHQI, but you must define the following 3 data sets:

### Input file

Is the file discussed previously in “Creating the Input File” on page 85. The DD name for this file must be **CHQIIN**.

### Side information data set

Is the standard labeled tape or general data set created by CHQI, which can be loaded online by TPF. The DD name for this file must be **SIDOUT**.

**Note:** If you are writing the data set to tape be sure to include IBM standard labels. For MVS, code the **LABEL = (,SL)** parameter on the SIDOUT DD card. For CMS, include the **SL** parameter on the FILEDEF SIDOUT command.

The side information data set is described further in “Side Information Data Set” on page 93.

### The output listing

is a printable file containing ANSI printer control characters. It includes a listing of the input file with attention messages and errors flagged, and a summary of the processing. The DD name for this file must be **CHQILIST**.

The contents of the output listing is described in “CHQI Output Listing” on page 93.

The following is a list of the possible return codes from CHQI.

### Code    Meaning

<b>0</b>	Processing completed normally; no error messages or attention messages are found.
<b>4</b>	Processing completed; attention messages were found, but no error messages.
<b>8</b>	Processing completed; errors were found.
<b>12</b>	CHQI was not able to open the input file.
<b>16</b>	CHQI was not able to open the output side information data set.
<b>20</b>	CHQI was not able to open the output listing file.



## Sample JCL for CHQI

The following is a sample of the JCL to create a tape under MVS:

```
//* Your JOB card here
/*****
/* Execute CHQI                *
/*****
//CHQIRUN EXEC PGM=CHQI41
//STEPLIB DD DSN=TPF.BASE.RLSE.LK,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//CHQIIN DD DSN=TPF.BASE.SIDINPUT(NEWSID),DISP=SHR
//CHQILIST DD SYSOUT=*
//SIDOUT DD UNIT=(TAPE,,DEFER),DISP=(NEW,KEEP),LABEL=(,SL),
// DSN=SIDTAPE,VOL=SER=xxxxx
/*
```

**Note:** Modify the data set names and volume serial number as appropriate.

The following is a sample of the commands to create a tape under CMS:

```
FILEDEF CHQIIN DISK SIDTAPE INPUT A
FILEDEF CHQILIST DISK SID LISTING A
FILEDEF SIDOUT TAP SL
CHQI
```

**Note:** Modify the file names as appropriate.

## Side Information Table Offline Program Output

As stated earlier, CHQI generates 2 output files: a side information data set that can be loaded online by TPF and an output listing describing the results of the conversion of the input file to the side information data set.

### Side Information Data Set

A side information data set consists of entries containing data to be added to or removed from the side information table, or descriptive text to be displayed during the load. The entries are grouped into TPF 4K (4095-byte) records.

Each record consists of an 80-byte header and forty 100-byte entries. The byte following the last entry is used as an end-of-data indicator. The 14 remaining bytes at the end of the record are not used and are set to binary zeros. The data format in the record header and the entries is defined by the ISIDE data macro.

### CHQI Output Listing

The CHQI output listing is a file of fixed-length 95-byte records. The first byte of each record is an ANSI printer control character. The listing echoes the input file with each line numbered and with attention or error messages interspersed among the input lines. The last page of the listing is a summary of the number of entries written to the side information data set and a summary of error and attention messages.

#### Attention messages

Attention messages are issued when valid but extraneous parameters are coded in REMOVE statements and when not all parameters are coded in ADD statements. When an attention message is issued, the ADD or REMOVE statement is still formatted into an entry and written to the side information data set (if there are no errors in the statement). Missing ADD parameters are set to null values in the generated entry; extraneous REMOVE parameters are ignored. If there are attention messages but no error messages, CHQI ends with a return code of 4.

### **Error messages**

When an error message is issued, CHQI does not format the error statement into an entry on the side information data set. When there are errors, CHQI ends with a return code of 8.

### **File error messages**

CHQI issues a file error message when a file (other than the listing) cannot be opened. Summary information is written to the listing file and processing ends without reading from the input file or writing to the side information output data set.

All attention and error messages begin with the prefix CHQI and are fully described in *Messages (System Error and Offline)* and *Messages (Online)*.

The last page of the listing is a short summary of the processing that took place. There are 3 sections to the summary.

The first section describes the entries that were written to the side information data set. It includes a message telling which subsystem the data must be loaded to, or that the data can be loaded to any subsystem, the number of ADD entries generated, the number of REMOVE entries generated, and the number of lines of descriptive text generated.

The second section summarizes any error or attention messages that were flagged during processing. If any error or attention messages were flagged, the lines that were flagged are also listed.

The third section contains the return code issued to the operating system when CHQI ended.

Figure 56 on page 95 shows an example of a listing with a number of error and attention messages.

```

LINE # | .....1.....2.....3.....4.....5.....6.....7.....8|
-----+-----
000001 | *****
000002 | *
000003 | *   SIDE INFORMATION INPUT -- A PLETHORA OF ERRORS.
000004 | *
000005 | *****
000006 |
000007 | LOAD      NAME=NEWINFO
-----+-----
A. CHQI0009E  INVALID LOAD PARAMETER
000008 |
000009 | DESCR  THIS TEXT WILL GET PRINTED ON THE OPERATOR'S CONSOLE
-----+-----
A. CHQI0015E  LOAD STATEMENT BEGINNING IN LINE 000007 ENDS WITHOUT SPECIFYING SS
000010 | DESCR  WHEN THE OFFLINE DATA SET GETS LOADED.
000011 |
000012 | MODE-BATCH
-----+-----
A. CHQI0030E  NO VERB WAS SPECIFIED FOR THIS PARAMETER
000013 |
000014 | ADD      LOAD      DATA
-----+-----
A. CHQI0007E  ADD STATEMENT BEGINNING IN LINE 000014 ENDS WITHOUT SPECIFYING NAME
A. CHQI0003W  ADD STATEMENT BEGINNING IN LINE 000014 ENDS WITHOUT SPECIFYING TP
A. CHQI0001W  ADD STATEMENT BEGINNING IN LINE 000014 ENDS WITHOUT SPECIFYING LU
A. CHQI0002W  ADD STATEMENT BEGINNING IN LINE 000014 ENDS WITHOUT SPECIFYING MODE
A. CHQI0014E  LOAD MUST BE THE FIRST STATEMENT
B. CHQI0015E  LOAD STATEMENT BEGINNING IN LINE 000014 ENDS WITHOUT SPECIFYING SS
B. CHQI0013E  INVALID VERB
000015 |
000016 | REMOVE ALL--EVERY
-----+-----
A. CHQI0012E  INVALID REMOVE PARAMETER
B. CHQI0011E  INVALID PARAMETER SYNTAX
000017 |
000018 | ADD      N=M=EXPEDITE L=LU.62 T=ABC$123$
-----+-----
A. CHQI0032E  REMOVE STATEMENT BEGINNING IN LINE 000016 ENDS WITHOUT SPECIFYING NAME
B. CHQI0031E  PARAMETER KEYWORD FOLLOWED BY PARAMETER KEYWORD
C. CHQI0019E  LU NAME MUST BEGIN WITH AN UPPERCASE LETTER
D. CHQI0039E  TP NAME CONTAINS HEX SUBSTRING WITH ODD NUMBER OF DIGITS
000019 |
000020 | REMOVE NAME-SAME MODE-SLOW LU-SLOWNET.LU62 TP-ZAPMAP
-----+-----
A. CHQI0005W  MODE PARAMETER IS IGNORED IN REMOVE STATEMENT
B. CHQI0004W  LU PARAMETER IS IGNORED IN REMOVE STATEMENT
C. CHQI0006W  TP PARAMETER IS IGNORED IN REMOVE STATEMENT
000021 |
000022 | ADD      N-3_DOG_NIGHT L-NET@NY.LU62MAPPED T-BLANK$40$SPACE M-1SECRESP
-----+-----
A. CHQI0035E  SYMBOLIC DESTINATION NAME IS LONGER THAN 8 CHARACTERS
B. CHQI0026E  Network ID CONTAINS INVALID CHARACTER

```

Figure 56. Example CHQI Output Listing (Part 1 of 2)

TPF SIDE INFORMATION TABLE OFFLINE PROGRAM						PAGE	2
LINE #	....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8						
-----+-----							
C.	CHQI0018E	LU NAME IS LONGER THAN 8 CHARACTERS					
D.	CHQI0038E	TP NAME CONTAINS AN EMBEDDED SPACE CHARACTER					
E.	CHQI0024E	MODE MUST BEGIN WITH AN UPPERCASE LETTER					
000023							
000024	ADD	N-MY	M-DOG	L-HAS	T-FLEAS		
000025		N-EVERY	M-GOOD	L-BOY.DOES	T-FINE		
-----A-----B-----C-----D-----							
A.	CHQI0025E	NAME IS ALREADY DEFINED FOR THIS STATEMENT					
B.	CHQI0022E	MODE IS ALREADY DEFINED FOR THIS STATEMENT					
C.	CHQI0016E	LU IS ALREADY DEFINED FOR THIS STATEMENT					
D.	CHQI0036E	TP IS ALREADY DEFINED FOR THIS STATEMENT					
000026							
000027	ADD	N-DEST0001	M-MODEMAP1	L-TPFNET1.MAPLU1	T-RECEIVER1		
000028	ADD	N-DEST0002	M-MODEMAP2	L-TPFNET2.MAPLU2	T-RECEIVER2		
000029	ADD	N-DEST0003	M-MODEMAP3	L-TPFNET3.MAPLU3	T-RECEIVER3		
000030	ADD	N-DEST0004	M-MODEMAP4	L-TPFNET4.MAPLU4	T-RECEIVER4		
000031	ADD	N-DEST0005	M-MODEMAP5	L-TPFNET5.MAPLU5	T-RECEIVER5		
000032							
000033	REMOVE	N-OBSDEST1					
000034	REMOVE	N-OBSDEST2					
000035	REMOVE	N-OBSDEST3					
000036							
000037	END						
-----A-----							
A.	CHQI0013E	INVALID VERB					

TPF SIDE INFORMATION TABLE OFFLINE PROGRAM						PAGE	3
SUMMARY							
SIDE INFORMATION DATA FOR ANY SUBSYSTEM							
5 ADD ENTRIES WERE GENERATED							
4 REMOVE ENTRIES WERE GENERATED							
2 LINES OF DESCRIPTIVE TEXT WERE GENERATED							
23 ERRORS WERE FLAGGED							
6 WARNINGS WERE FLAGGED							
THE FOLLOWING LINES WERE FLAGGED:							
7 9 12 14 16 18 20 22 25 37							
RETURN CODE = 8							

Figure 56. Example CHQI Output Listing (Part 2 of 2)

## Loading the Side Information Data Set to TPF

After you have the side information data set generated and free of errors, you can move on to steps 5, 6, and 7 listed in the beginning of “Generating the Side Information Table for Mapped Conversations” on page 83.

The following description refers to Figure 57, which shows a TPF console during the mounting of an offline generated side information tape, initialization of the online side information table, and loading of the tape to the side information table.

1. The ZTVAR command activates the subchannel address of the tape drive on which the tape is mounted.
2. The 2 ZTLBL commands create a TPF internal label (SID) matching the IBM standard label on the offline generated tape, and make the internal label usable so that the tape can be mounted.
3. The ZTMNT command defines the tape drive at subchannel address 580 as the active SID tape. At this point the tape is accessible to a TPF online process.
4. The side information table has not been initialized, so the ZNSID INITIALIZE function message is entered.

**Note:** The side information table must be initialized before any data can be added to it. Initialization sets up the structure of the side information table and clears any data that previously existed online. The side information table must be initialized only when a new TPF system is brought online, or when the entire table is to be replaced. If you attempt to load the side information data set when the online table is not initialized, the following response is returned to the operator console:

```
NSID0016E RECORD ID CHECK FAILURE, CHECK SIDE INFORMATION
TABLE INITIALIZATION
```

5. The ZNSID LOAD command loads the tape into the newly initialized side information table.
6. ZNSID D displays 1 of the newly loaded side information entries.

```
User:  ZTVAR A 580
System: CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
COTU0101I 11.48.39 TVAR BSS PROT ENTRY ACQUIRED FOR DEVICE 580
BY THIS PROCESSOR
CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
COTE0001I 11.48.39 TVAR - TAPE STATUS

ADDRESS  TPNAME  SSUNAME  STATUS  TPIND  VOLSER  DENS  #BLOCKS LDR
580      AVAIL

User:  ZTLBL SID C LSL F$SIDOUT$ I G
System: CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
COTL0001I 11.48.39 DTLB HPN - TAPE LABEL INFORMATION
SID - NOT USABLE I/O-INPUT L-SL D-ALL NOCOMP
NOBLK T - 000
F-SIDOUT G - S-0001
RETENTION PERIOD - 00007
LAST MOUNTED -
LAST FILE SERIAL NO -

User:  ZTLBL SID U
System: CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
COTJ0044I 11.48.39 TLBL HPN COMPLETE

User:  ZTMNT SID 580 AI
System: CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
COTM0046I 11.48.39 TMNT HPN TAPE SID MOUNTED ON DEVICE 580
VSN A00112 G S0001 D38K SL NOBLK
```

Figure 57. Loading a Side Information Offline Generated Tape (Part 1 of 2)

```

User:  ZNSID INITIALIZE
System: CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
NSID0004I 11.48.39 SIDE INFORMATION TABLE INITIALIZED

User:  ZNSID LOAD T-SID
System: CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
NSID0037I 11.48.39 LOADING SIDE INFORMATION DATA
CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
NSID0017I 11.48.39 USER TEXT FROM SIDE INFORMATION TAPE
ADDING 100 ENTRIES TO SIDE INFORMATION TABLE
--- END OF DESCRIPTION ---
CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
NSID0005I 11.48.39 NEW SIDE INFORMATION TABLE ENTRIES LOADED
ENTRIES PROCESSED -      100
ENTRIES ADDED     -      100
ENTRIES CHANGED   -       0
ENTRIES REMOVED   -       0
ERRORS DURING LOAD -       0
CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
COTC0087A 11.48.39 TCLS HPN      REMOVE SID FROM DEVICE 580
                      VSN A00112 NOBLK

User:  ZNSID D N-NAME100
System: CSMP0097I 11.48.39 CPU-B SS-BSS SSU-HPN IS-01
NSID0002I 11.48.39 SIDE INFORMATION ENTRY DISPLAY
NAME- NAME100
TP- TP100
LU- LU100
MODE- MODE100

```

Figure 57. Loading a Side Information Offline Generated Tape (Part 2 of 2)

## Resource Manager

The resource manager services requests relating to conversations. The requests are initiated through presentation services when a TPF/APPC verb is issued by a transaction program or through the presentation services for the control operator (PS.COPR). PS.COPR is activated when a COPR.TP issues a TPF/APPC change number of sessions verb (CNOSC macro) or control is passed from CNOS.TP.

The resource manager coordinates the following functions:

- Assigning conversation control blocks (CCBs) for conversations
- Assigning session control blocks (SCBs) for sessions
- Choosing sessions for a conversation and, if necessary, requesting use of the session through bidding
- Requesting the session manager to activate a new session or to deactivate an existing session
- Replying to requests (BIDs) that are received from remote resource managers for the use of a session
- Limiting resource sessions
- Queuing allocate requests.

The resource manager also activates the functions required of the half-session components.

---

## Session Manager

The session manager is responsible for insuring that the underlying LU-LU session needed for a conversation is available. When an ALLOCATE request is issued, the session manager sends out a LOCATE/CDINIT to initiate another session if there is no session active and bound or available for the conversation, and the session limit has not been reached.

**Note:** There is 1 exception to this: PU 2.1 secondary LU (SLU) thread sessions cannot be activated from the TPF side.

## Initiating a Session

If you want to initiate a session before issuing an ALLOCATE request, you can use 1 of the following ways:

- The VTAM operator issues a **VARY NET ACT LOGON** command.
- Through the use of autologon at network definition for VTAM or CICS. To define autologon in VTAM, code **LOGAPPL** on an LU. To define autologon in CICS, code **CONNECT=AUTO** on the DFHTCT statements.
- The end user (remote LU operator) can also initiate a session. The procedures for this depends on the type of remote LU. See the publications associated with the remote LU for details on these procedures.

### LU 6.2 BIND Command

See *IBM Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2* for information about the LU 6.2 BIND command.

## Session Termination

Sessions are terminated with an UNBIND command. Session termination can be requested in the following ways:

- The VTAM operator issues a **VARY NET INACT**.
- The remote LU sends a LOGOFF or Terminate-Self command to its SSCP.
- The TPF operator issues a **ZNETW INACT** command.

**Note:** If you inactivate a local TPF/APPC PLU that has SLU threads, only sessions with the PLU are ended; there is no affect on any sessions with the SLU threads.

- The TPF operator issues a **ZNCNS RESET** command.
- The TPF operator issues a **ZNCNS CHANGE** command to lower session limits.

For details on the ZNETW and ZNCNS commands, see the “ACF/SNA Communications Commands” in *TPF Operations*.

---

## Message Flow

The TPF/APPC support code buffers all data messages sent as a result of the TPF/APPC SEND\_DATA verb. The FMH5 built as a result of an ALLOCATE request is also buffered. The messages remain in the buffer until 1 of the following occurs:

- The buffer is full
- The TPF/APPC FLUSH verb is issued by the transaction program
- Some other TPF/APPC verb causes the buffer to be flushed.

When the buffer is flushed, the TPF/APPC support code issues the TPF ROUTC macro with the extended routing control parameter list (RCPL) to forward the contents of the buffer. The buffer size is set to the maximum request unit (RU) size of the session.

All inbound messages that are to be handled by the TPF/APPC support code arrive and are processed by the base TPF SNA support. The base TPF SNA support, recognizing that the remote LU sending the message is defined (using OSTG) as an LU 6.2 resource, passes the message to LU 6.2 OPZERO program CS2A. (CS2A is in control program CSECT CCSNAE.)

The LU 6.2 OPZERO program passes control to the communications source program (COA4), which then passes control to the message editor. To use TPF/APPC support, the remote LU sending the message must be in session with an LU defined by the SIP MSGRTA statement that specifies the input message processor program (CHDD) as the message editor.

## Inbound Message Queuing

As stated earlier, the TPF/APPC support package receives incoming messages from the communications source program in AM0SG format. The message may be in 381-byte, 1055-byte, or 4KB blocks. CHDD reblocks the incoming messages into 4K half-session to presentation services records (HPRs). These blocks are chained together so that individual logical records are identified and presented to presentation services. Logical records are identified by the logical record length (LL) included as the first 2 bytes of a record. The maximum logical record length supported by TPF/APPC is 32 767 (32KB). If a logical record is too large to fit in 1 4K block, it is segmented into multiple 4K pool files. These pool files are chained using the TPF forward chain field in the standard record header. The end of the chain is indicated by a value of zero in the forward chain field. Each logical record contains a pointer in its primary HPR to the next logical record.

**Note:** If an FMH7 record is received, a new logical record is created for the FMH7, even if the previous logical record is incomplete.

Once the message is blocked into HPR records, CHDD either queues the message or passes it directly to the transaction program. The message is passed directly to the transaction program (that is, there is no queuing) if the following conditions are true:

- The transaction program (TP) is waiting for data; that is, the TP issued a verb that caused the ECB to be suspended, or the TP issued an ACTIVATE\_ON\_RECEIPT or ACTIVATE\_ON\_CONFIRMATION verb.
- The message contains enough information or data to satisfy the request.

When the message is queued, the HPR records are filed in short-term pools.

The conversation control block contains the file address of the first and last HPRs on the inbound queue. Figure 58 on page 101 shows the HPRs as queued on the inbound queue.



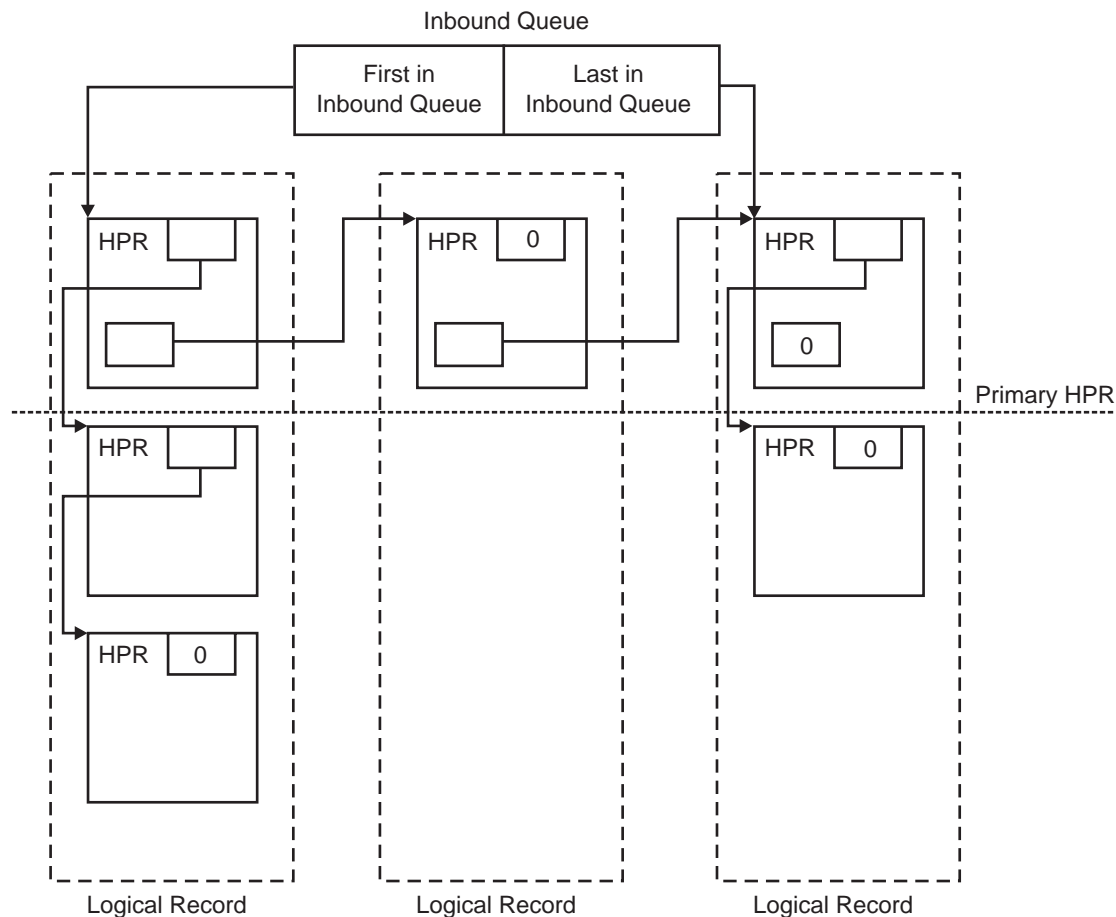


Figure 58. Inbound Queue

## System Restart

TPF restart is generally responsible for bringing the TPF system back to the state it was in at the time of a system outage. During software IPL, the CCBs and SCBs are considered critical records, and the entire CCB and SCB areas are filed to the DASD area to insure a current copy is available for the subsequent TPF restart programs.

The TPF restart programs recognize when the TPF SNA resources have been redefined across a system outage. TPF SNA resources are redefined by loading a new copy of the OSTG pilot tape, which is used to define the SNA resources as known to TPF. When this occurs, all sessions with the remote SNA resources are disrupted. When the TPF/APPC restart programs recognize that this fresh load has taken place, the conversation allocation process must begin again; no attempt is made to notify the remote transaction program of any conversation deallocation, because the underlying session required to send the notification has been unbound.

## Session Considerations

If a hardware IPL occurs, TPF unbinds all LU 6.2 sessions and does not attempt to bring these sessions back. TPF does retain the session limits previously initialized so you do not have to issue another CNOS INITIALIZE request.

If a software IPL occurs, the LU 6.2 sessions remain active; however, all conversations are deallocated.

## Conversation Considerations

Conversations that were active at the time of an outage are represented as a transaction program instance, identified in a TPF entry control block (ECB) by a unique TCB ID. Because all ECBs are lost across a system outage, any conversations controlled by the transaction program instance must be deallocated. The TPF restart segments activate the TPF/APPC support restart segments. TPF/APPC restart uses the keypointed copy of the CCB on the fixed file to find all conversations that were active at the time of the system outage. Then, TPF/APPC notifies the remote transaction program that the conversation was terminated by sending a DEALLOCATE request. The local TPF transaction program, represented by the ECB and associated programs, cannot be notified of the system outage implied by a TPF restart.

---

## Subsystem Considerations

A TPF transaction program can run in any subsystem. When a remote LU issues an ALLOCATE request, TPF activates the transaction program in the subsystem where the local LU for that session resides. Once a conversation is started, the ECB representing the transaction program cannot switch subsystems.

---

## Considerations for a Tightly Coupled Environment

The TPF/APPC support code allows the TPF user-written transaction programs to be run on any available I-stream in a TPF tightly coupled environment. In fact, the TPF transaction programs can be defined to run on any I-stream: the main I-stream or a specific I-stream. This definition is done with the ITPNT macro as described in *TPF General Macros*.

Because the TPF transaction programs are running on any I-stream in the tightly coupled environment, the TPF/APPC support code activated by a TPF/APPC verb request also runs on the same I-stream that initiated the verb request. With the various components of the TPF/APPC support code running on multiple I-streams concurrently, the potential for simultaneous updating exists. To ensure proper sequence of updates, TPF/APPC support provides a locking mechanism by means of the ILCKCB and IULKCB macros. (See *TPF System Macros* for details on these macros.)

The TPF/APPC support code is sometimes activated as a result of inbound messages received in TPF from the remote transaction programs. To insure that these messages are queued and presented to the transaction program in the same sequence as they arrive in TPF, this portion of the TPF/APPC support code must always run on a single specified I-stream. To accomplish this serialization, when the TPF communications source program (COA4) determines that the message is from a resource defined as an LU 6.2 resource, it uses the SWISC macro to assign the ECB to I-stream 2 and to activate the edit program specified in the routing control application table (RCAT). Specifying I-stream 2 instead of the main I-stream allows the queuing logic overhead to be moved off of the main I-stream. (Of course, in a uniprocessor system, there is only 1 I-stream, and the SWISC macro simply performs the program enter function without changing I-streams.)

---

## Considerations for a Loosely Coupled Environment

For sessions between a remote LU and a TPF/APPC LU (not an SLU thread), there is a restriction that forces all sessions with a particular remote LU 6.2 to be established with only 1 processor in the loosely coupled complex. It does not matter which processor is chosen, but once the first session is established, all sessions with this remote LU 6.2 must go to the same processor as the first session.

To overcome this restriction, TPF provides unique TPF/APPC service LUs for each processor and a TPF service transaction program to allow TPF/APPC transaction programs in all processors of a loosely coupled complex to have LU 6.2 conversations with a particular remote LU 6.2.

To fully utilize TPF/APPC in a loosely coupled environment, there are some installation tasks that you need to perform. These tasks are described in “Loosely Coupled Complex Example”; however, before setting up your system to run TPF/APPC in a loosely coupled environment, you must determine the kind of sessions that your applications require.

A remote LU can view the complex in 1 of 2 ways:

- Each host as a separate TPF image
- The entire loosely coupled complex as a single TPF image.

If a remote LU needs sessions with multiple TPF hosts (separate image view), the LU can do 1 of the following:

- Establish sessions with a unique TPF/APPC service LU in each TPF host required
- Establish sessions with SLU threads in each TPF host required.

If a remote LU only requires sessions with 1 TPF host (single image view), the LU can specify a local TPF/APPC LU name to establish a session with 1 TPF host.

1 factor to consider when determining whether an application needs sessions with multiple hosts or only 1 host is the number of messages it generates. Some applications may generate such a high volume of messages that funneling all the requests to a single TPF processor in the complex would create a problem with performance.

From the point of view of the TPF system, if all the TPF hosts need to communicate with the same remote, they can also establish sessions using the TPF/APPC service LUs in each host.

## Loosely Coupled Complex Example

Figure 59 on page 105 the many possible types of TPF/APPC sessions. In this example, there are 2 TPF loosely coupled complexes, each containing 2 hosts (LC1 contains TPFA and TPFB, and LC2 contains TPFC and TPFD), and several remote LU 6.2 nodes.

The local TPF/APPC LUs for the LC1 complex are **APPC**, **TOUR**, and **DEAD**. The local TPF/APPC LUs for the LC2 complex are **LU62** and **LVNV**. (The local TPF/APPC LUs could be the same for both complexes in this example.) The TPF/APPC service LU for: (1) TPFA is **SVCA**, (2) TPFB is **SVCB**, (3) TPFC is **SVCC**, and (4) TPFD is **SVCD**.

In each host, the diagram shows the local TPF/APPC LUs and the unique TPF/APPC service LU, the RVTs for the remote LUs, and any SCBs associated with the particular remote RVT. Each SCB represents 1 session. The LU name in the SCB box indicates the local partner LU with which the remote is in session.

In the example, remote LU **DB2** needs to communicate with multiple TPF hosts, so it is set up to view each TPF host in the complex as a separate image. Sessions are established using the service LU in each host; **DB2** currently has 2 sessions with each host.

Remote LU **PS2Y** also views LC1 as separate TPF images and has a session established with each service LU in the complex. If necessary, **PS2Y** could also establish sessions with 1 of the TPF/APPC LUs in TPFC or TPDF, or it could establish sessions with 1 or both service LUs in TPFC and TPDF.

Remote LU **PS2X** views the LC1 complex as a single TPF image. It currently has 3 sessions established with LU **APPC** in TPFA. Because TPFB is in the same loosely coupled complex as TPFA, no sessions between **PS2X** and TPFB can be established at this time. However, because TPFC and TPDF are in a different complex, **PS2X** can establish sessions with either host in the LC2 complex.

**Note:** Even though sessions between TPFB and **PS2X** cannot be established at this time, transaction programs in TPFB *can* establish conversations with transaction programs in **PS2X** using the TPF service transaction program.

Remote LUs **PS2K** and **PS2Q** both view LC2 as a single TPF image. **PS2K** has sessions with **LU62** in TPFC and cannot establish sessions with TPDF, but can establish sessions with the hosts in LC1. **PS2Q** has a session with **LVNV** in TPDF and cannot establish sessions with TPFC.

**PS2N** has a session with the service LU in TPFC, which allows **PS2N** to establish sessions with the service LU of TPDF if needed.

Although not shown with lines in this diagram, there are also sessions between the service LUs of the hosts in different loosely coupled complexes as well as in the same complex. For example, **SVCA** in TPFA has 2 sessions with **SVCC** in TPFC and 1 session with **SVCD** in TPDF. **SVCA** also has a session with **SVCB**.

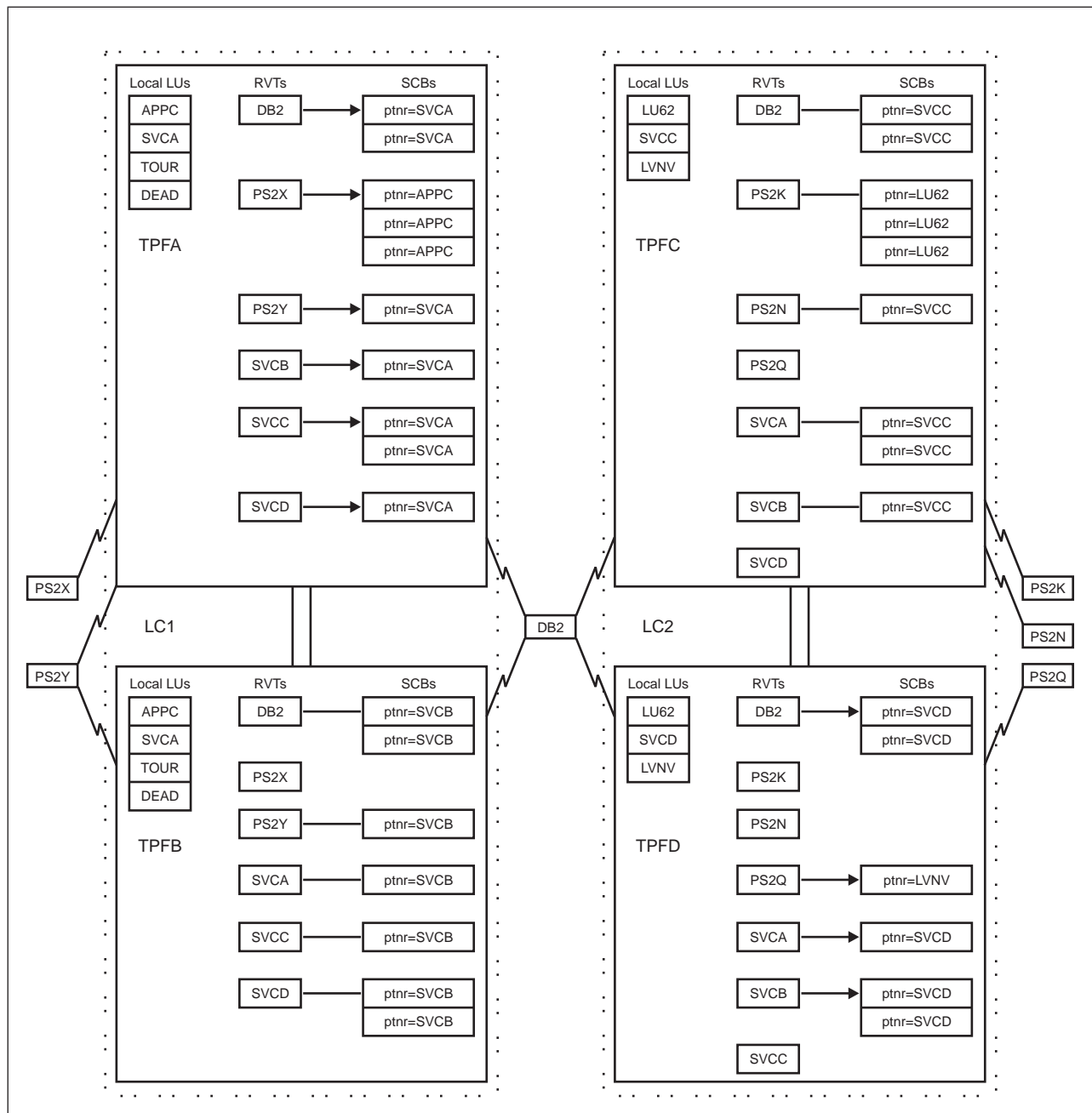


Figure 59. Loosely Coupled Complex Example for TPF/APPC Sessions

## Loosely Coupled Installation Checklist

This section describes the tasks you must perform to use the TPF/APPC support in a loosely coupled environment.

**Note:** These tasks are also listed in the “TPF/APPC Installation Checklist” on page 60. This section is provided to give more specific details for the definitions required for loosely coupled environments.

The last 3 tasks are not required if all remote LU 6.2s are to view the complex as a single TPF image and the TPF system does not initiate any conversations with these remote LUs (that is, the remote LUs start all conversations).

These tasks are as follows:

- Define the local TPF applications that represent the TPF/APPC LUs.
- Define 1 TPF/APPC service LU for every processor in the TPF loosely coupled complex, if necessary.
- Define the service transaction program in the transaction program name table (TPNT), if necessary.
- Initialize the session limits for the mode name used by the TPF/APPC service LU, if necessary.

### Defining Local TPF/APPC LUs

See “TPF/APPC Installation Checklist” on page 60 for information about how to define the local TPF/APPC LUs.

### Defining the TPF/APPC Service LU

Define 1 TPF/APPC service LU for every processor in the TPF loosely coupled complex with a MSGRTA statement as follows:

```
MSGRTA APLIC=SVCx,EDIT=CHDD,APROC=x,ASNA=APPC
```

where x is the processor ID.

For example, if there are 4 processors in the loosely coupled complex (TPFA, TPFB, TPFC, TPFD), code the following:

```
MSGRTA APLIC=SVCA,EDIT=CHDD,APROC=A,ASNA=APPC  
MSGRTA APLIC=SVCB,EDIT=CHDD,APROC=B,ASNA=APPC  
MSGRTA APLIC=SVCC,EDIT=CHDD,APROC=C,ASNA=APPC  
MSGRTA APLIC=SVCD,EDIT=CHDD,APROC=D,ASNA=APPC
```

### Defining the Service Transaction Program

If any remote LUs view the complex as a single TPF image and TPF is to initiate conversations with these remotes, define the service transaction program in the transaction program name table with the ITPNT macro as follows:

```
ITPNT TYPE=ATTACH,PGM=CHRP,TPN=TPF_SERVICE_TP  
ITPNT TYPE=AOR,PGM=CHRM,TPN=CHRM
```

### Initializing the Session Limits

When initialize the network, use the ZNCNS command to initialize the session limits for the mode name used by the TPF/APPC service LUs. (As with the ITPNT definitions, this is necessary only if any remote LUs view the complex as a single TPF image and TPF is to initiate conversations with these remotes.)

You must issue a ZNCNS INITIALIZE for *each* processor pair. For example, if there are 4 processors (TPFA, TPFB, TPFC, TPFD) and you decide that each processor needs 10 sessions with each of the remaining processors, issue the following commands:

- From TPFA:

```
ZNCNS INIT LU-SVCB MODE-SERVMODE LIMIT-10 LOCAL-SVCA
ZNCNS INIT LU-SVCC MODE-SERVMODE LIMIT-10 LOCAL-SVCA
ZNCNS INIT LU-SVCD MODE-SERVMODE LIMIT-10 LOCAL-SVCA
```

- From TPFB:

```
ZNCNS INIT LU-SVCC MODE-SERVMODE LIMIT-10 LOCAL-SVCB
ZNCNS INIT LU-SVCD MODE-SERVMODE LIMIT-10 LOCAL-SVCB
```

- From TPFC:

```
ZNCNS INIT LU-SVCD MODE-SERVMODE LIMIT-10 LOCAL-SVCC
```

**Notes:**

1. This is only an example; you can specify any limit that is required by your environment. You can also specify the number of contention winners and contention losers. See the ZNCNS command in *TPF Operations* for additional information.
2. It does not matter which host in the TPF to TPF pair initializes the session limits. For example, TPFB can issue the ZNCNS message to initialize the limits with TPFA.
3. The INITIALIZE only establishes the session limits; it does not activate sessions for the mode name SERVMODE. Sessions are brought up by TPF/APPC as needed.
4. Once the session limits are fine-tuned for a TPF complex, you can write a COPR.TP to issue CNOSC macros to initialize the session limits rather than issuing several ZNCNS commands.

---

## User Exits

The TPF/APPC support code activates the following user exits:

**CSXA** Activated when an LU-LU session is established.

**CSXB** Activated when a conversation allocation request has been successful or failed.

**CSXC** Activated when a session has ended.

---

## Transaction Program Design Considerations

When designing your application, referred to as a transaction program (TP) in LU 6.2 terminology, a major factor to consider is the volume of traffic that will flow between the TP in the TPF system and the TP in the remote LU. There are 3 classes of LU 6.2 conversations that can be used:

- Traditional LU 6.2 conversations
- Pipeline LU 6.2 conversations
- Shared LU 6.2 conversations.

Each class has advantages and disadvantages in the areas of application design and throughput. To help you understand these advantages and disadvantages, the following information compares the characteristics of each class performing a common task; processing 4 messages where the TPF TP sends a request (data) to the remote LU and receives a response (data).

### Traditional LU 6.2 Conversations

The majority of existing LU 6.2 applications send and receive data on the same session. This is known as traditional LU 6.2 conversations. From a TP design point

of view, this is the easiest method to use. If the number of transactions to be processed is low, then traditional LU 6.2 conversations are the best choice.

Figure 60 shows how a TP written to use traditional LU 6.2 conversations processes 4 messages. 1 ECB processes an entire message; the request and the response (req1 and rsp1, for example). Each message uses its own conversation, and the request and response flow on the same conversation.

Only 1 conversation can use a given LU 6.2 session at a time. When a conversation ends, another conversation can use that session. The main drawback to using traditional LU 6.2 conversations is that the session is not usable from the time that the TPF system sends out the request until the response is received. In the sample TP, the 4 messages are processed 1 at a time. Even though the TPF system has 4 requests to send out, the next request cannot be sent out until the response to the current request is received.

Using LU 6.2 parallel sessions helps the throughput situation because there are multiple sessions between the TPF system and the remote LU. For example, if you have 20 parallel sessions, you can process 20 messages at a time. If the TPF system is generating a consistently large amount of requests and processing each request in the remote LU takes a long time, then it is likely that all parallel sessions will be in use forcing requests to queue up in the TPF system. For high-volume TPs like these, traditional LU 6.2 conversations are not a good choice.

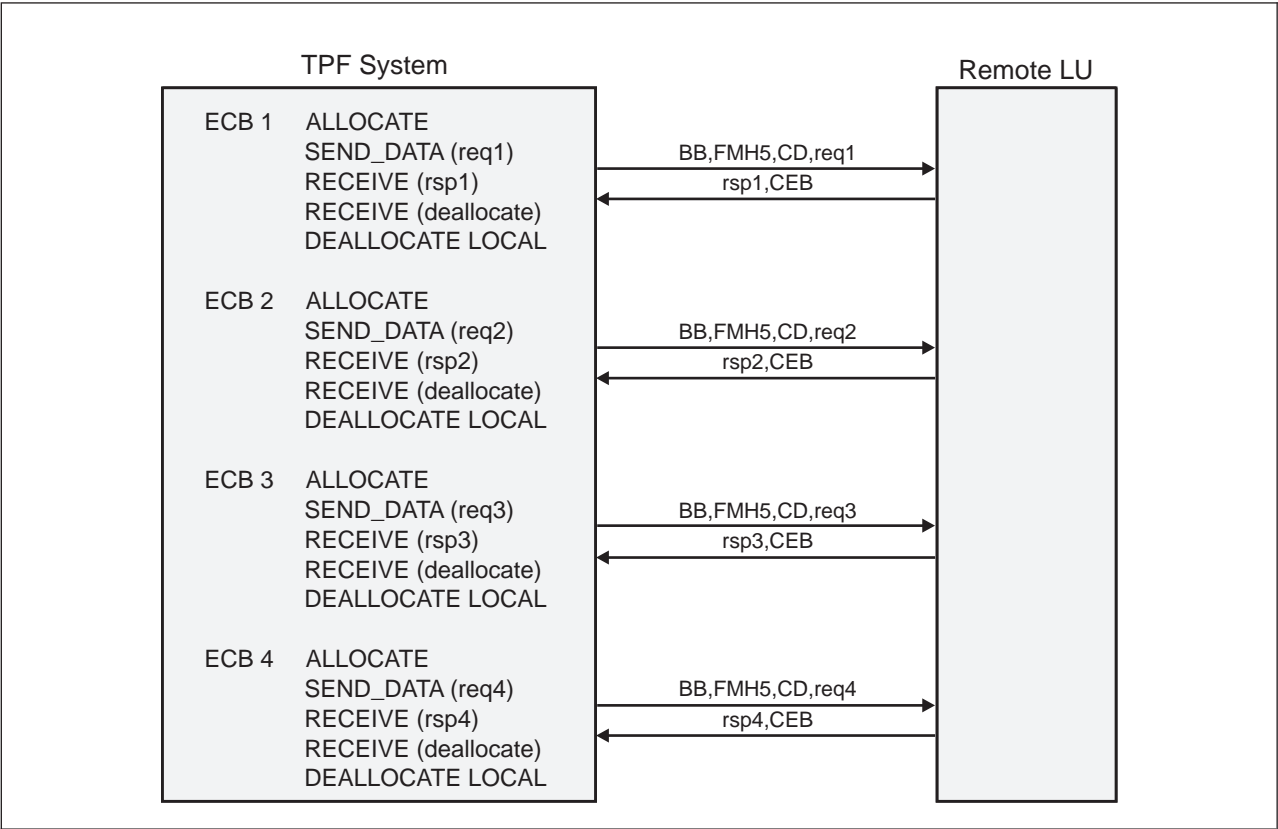


Figure 60. Traditional LU 6.2 Conversations

## Pipeline LU 6.2 Conversations

Pipeline LU 6.2 conversations are 1-way conversations. In most cases, 2 conversations are used; 1 for sending data and the other for receiving data. Using



pipeline conversations requires more work by the TP because a request is sent over 1 conversation, but the response comes back over a different conversation. It is up to the TP to correlate responses with requests.

Figure 61 on page 110 shows how a TP written to use pipeline LU 6.2 conversations processes 4 messages. 2 ECBs are needed to process a message; 1 ECB sends out the request and a different ECB is activated to process the response when that response is received. To process a message, 2 conversations are used; 1 for the request and 1 for the response. By ending the conversation as soon as a request is sent out, the session is immediately available and can be used to send out a second request before the response to the first request is received.

Pipeline conversations allow a large amount of messages to flow between the TPF system and a remote LU. An example of pipeline conversations can be found in any APPN network. A control point (CP) has special sessions, called CP-CP sessions, with each adjacent CP. These CP-CP sessions are a pair of LU 6.2 sessions used as 1-way pipes.

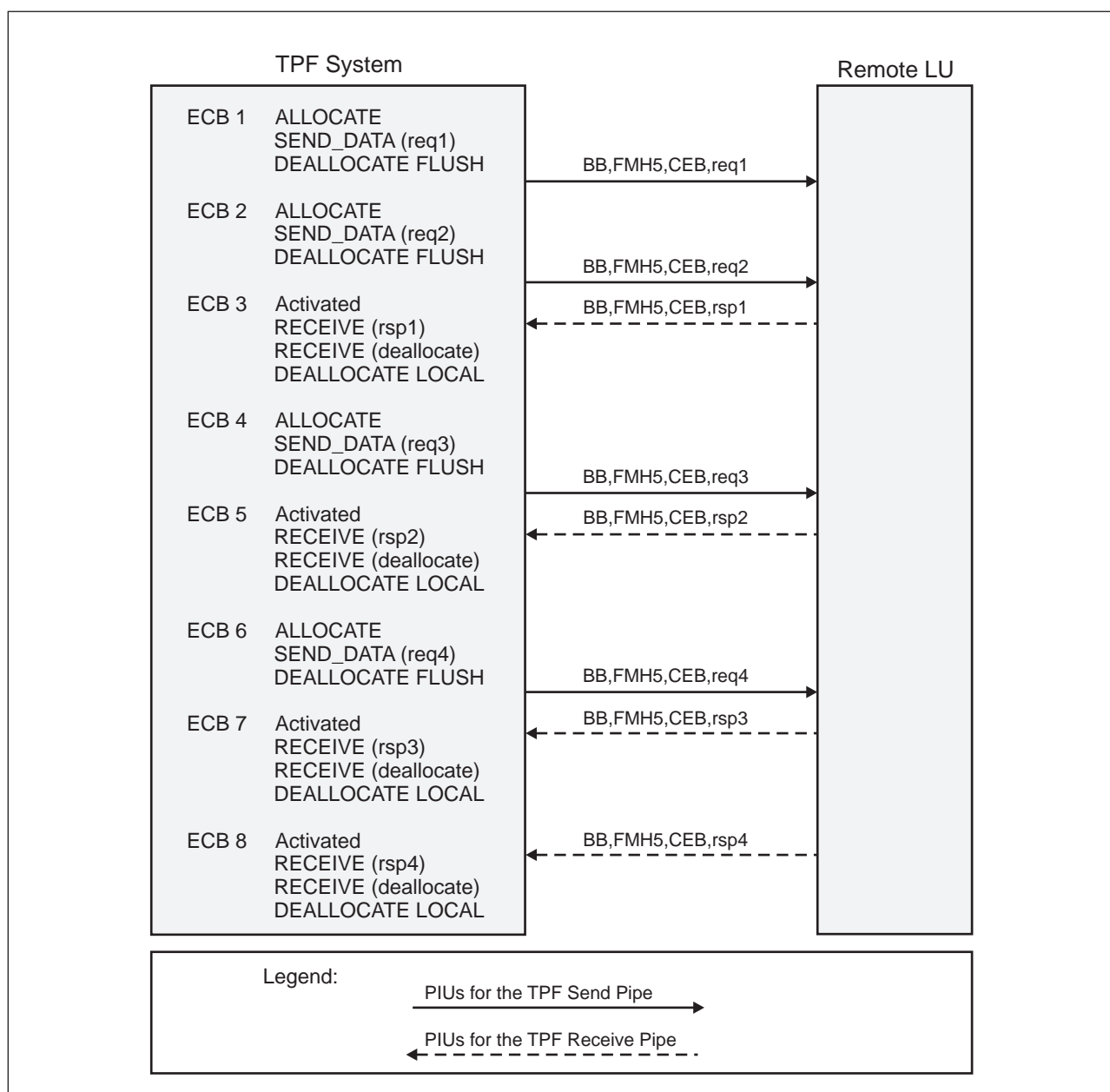


Figure 61. Pipeline LU 6.2 Conversations

## Shared LU 6.2 Conversations

Shared LU 6.2 conversations are similar to pipeline conversations in that a pair of 1-way pipes are used. The difference is that shared conversations allow multiple messages to be processed by 1 conversation, thereby eliminating the overhead involved with starting and ending the conversations associated with pipeline conversations. Shared conversations are more efficient from a network point of view because the number of path information units (PIUs) that flow are drastically reduced compared to traditional or pipeline LU 6.2 conversations.

When starting a conversation, a TPF TP codes a parameter on the TPPCC ALLOCATE macro (or `tppc_allocate` C language function) to indicate that the

conversation is shared. The shared option on the ALLOCATE verb starts the conversation that the TPF system will use as the 1-way pipe to send data.

Only certain LU 6.2 verbs are allowed for shared conversations because these conversations are 1-way outbound pipes. The valid verbs include SEND\_DATA, FLUSH, GET\_ATTRIBUTES, and DEALLOCATE (except when TYPE=CONFIRM is specified). Any ECB can issue 1 of these verbs for a shared conversation. For a conversation that is not shared, only 1 ECB (the ECB that creates or owns the conversation) can issue verbs for that conversation.

Multiple requests (from different ECBs) all flow on the same shared conversation. To increase network efficiency, multiple requests are packaged together and sent out in a single PIU. Multiple responses are also returned in a single PIU.

Figure 62 on page 112 shows how a TP written to use shared LU 6.2 conversations processes 4 messages. 1 ECB starts the shared conversation, which causes the TP in the remote LU to start the other shared conversation. Next, ECBs in the TPF system representing different TPs issue SEND\_DATA verbs. In this example, 4 requests fit in a PIU, so all 4 requests are sent out in a single PIU. However, only 2 responses fit in a PIU. To handle responses, the ECB created when the second conversation was started (ECB 2) issues an ACTIVATE\_ON\_RECEIPT verb (equivalent to the LU 6.2 RECEIVE verb except that the data received is passed to a new ECB, not to the ECB that issued the ACTIVATE\_ON\_RECEIPT verb). When the first response reaches the TPF system, a new ECB is created (ECB 7) and the response (rsp1) is passed to ECB 7. ECB 7 immediately issues an ACTIVATE\_ON\_RECEIPT verb, and then processes the response that arrived. ECB 8 is created right away because the second response (rsp2) has already been received by the TPF system. ECB 8 also issues ACTIVATE\_ON\_RECEIPT, and then processes the response that it was passed (rsp2).

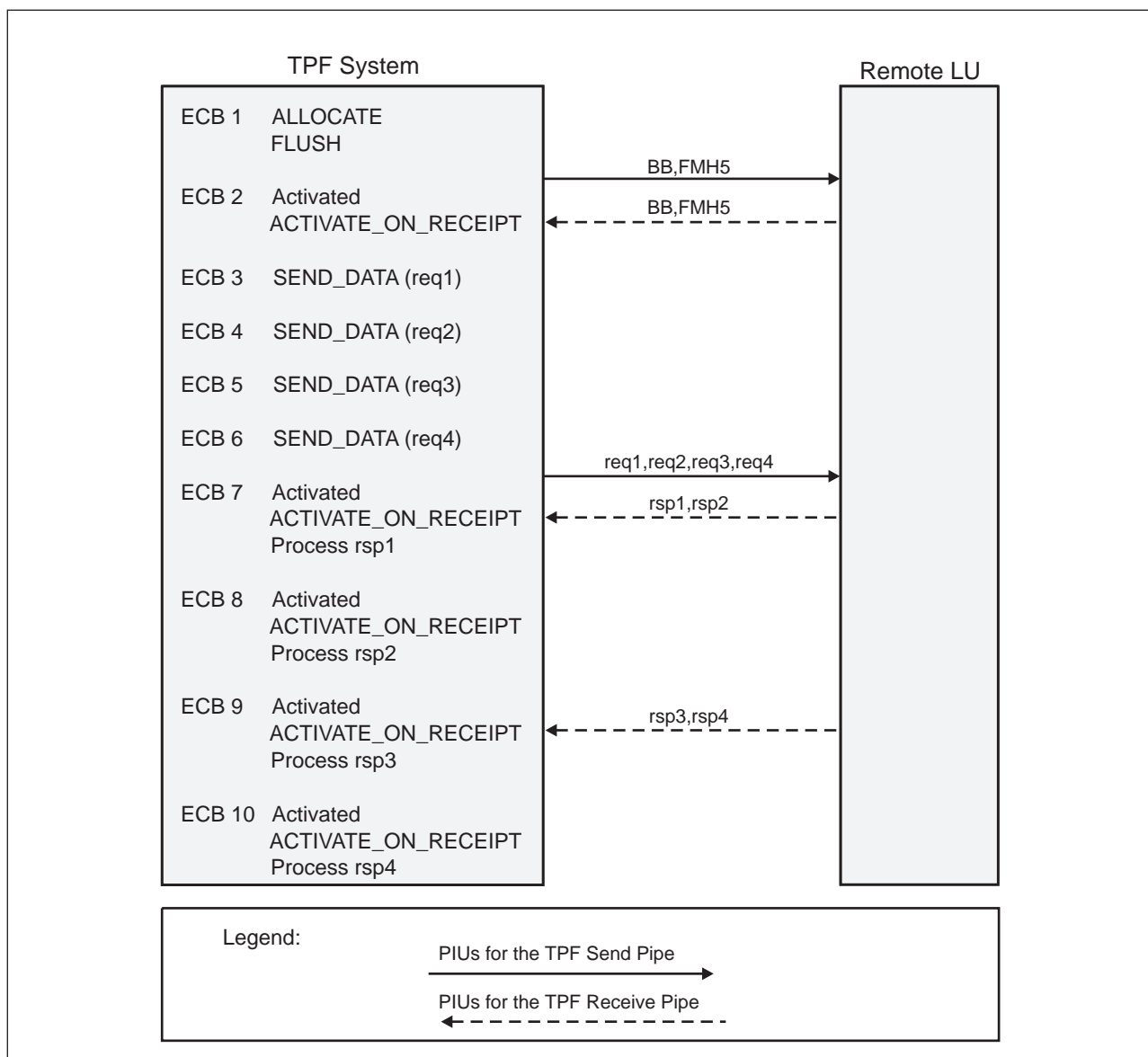


Figure 62. Shared LU 6.2 Conversations

### Why Use Shared Conversations

Table 3 on page 113 shows the statistics for processing 4 messages using the different LU 6.2 conversation methods. To do comparisons, a larger scale is needed, so the data in Table 4 on page 113 for processing 100 messages will be used. This table shows the value of using shared conversations to process high-volume messages.

In traditional and pipeline LU 6.2 conversations, 40-50% of the verbs issued by the TP (ALLOCATE and DEALLOCATE) is overhead used to start and end those conversations. For shared LU 6.2 conversations, less than 1% of the verbs is overhead.

An even larger savings is in the number of PIUs. Both traditional and pipeline LU 6.2 conversations require 2 PIUs to process each message. Table 4 on page 113 uses realistic values to say that 25 requests fit in 1 PIU and 10 responses fit in 1

PIU. Using these numbers, only 16 PIUs are needed to process 100 messages with shared LU 6.2 conversations. The other classes of conversations require 200 PIUs, therefore shared conversations reduce the number of PIUs by over 90% in this example.

*Table 3. LU 6.2 Statistics for Processing 4 Messages*

Statistic	Traditional	Pipeline	Shared
Number of ALLOCATE verbs	4	4	1
Number of SEND_DATA verbs	4	4	4
Number of RECEIVE or ACTIVATE_ON_RECEIPT verbs	8	8	4
Number of DEALLOCATE verbs	4	8	0
Number of conversations	4	8	2
Number of PIUs	8	8	5

*Table 4. LU 6.2 Statistics for Processing 100 Messages*

Statistic	Traditional	Pipeline	Shared
Number of ALLOCATE verbs	100	100	1
Number of SEND_DATA verbs	100	100	100
Number of RECEIVE or ACTIVATE_ON_RECEIPT verbs	200	200	100
Number of DEALLOCATE verbs	100	200	0
Number of conversations	100	200	2
Number of PIUs	200	200	16

---

## Sample Transaction Programs

When describing program-to-program communications, you must keep in mind your perspective. Program-to-program communications relies on the presence of a local program and a complimentary remote program. The function provided by this pair of programs is distributed in the 2 systems.

The following sample TPF transaction programs (TPs) show how you can use the TPF/APPC support package to have 2 TPF systems communicate with each other to perform a simple task. To reduce confusion, the following sample programs are described as the **requesting** program and the **requested** program. The **requesting** program initiates the communications process and asks the **requested** program to provide some information. The **requesting** program then accepts the response from the **requested** program, takes some action based on the response and then terminates the communications process. In the following example, the simple task is to have 1 TPF system (the **requesting** TPF system) check the validity of a FACE record type and ordinal number on a remote TPF system (the **requested** TPF system).

These samples are provided solely to illustrate the interrelationship of the 2 transaction programs and are not intended to demonstrate all the functions or possibilities of the TPF/APPC support package. In fact, the sample programs described below ignore most of the error situations that may occur and that a fully

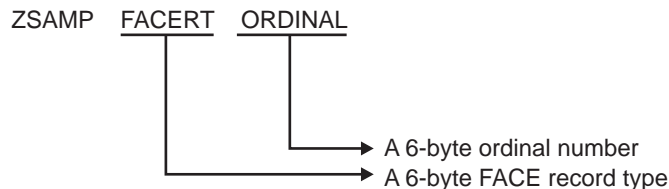
functional program would be expected to handle. It is beyond the scope of these sample programs to demonstrate the necessary error handling of a fully functional program.

## Sample Transaction Program Functions

The sample transaction programs provide the following functions:

1. Initiate the requesting TPF TP with a command, and pass a FACE record type and ordinal number.
2. The requesting TPF TP allocates a conversation with the requested TPF TP.
3. The requesting TPF TP sends the record type and ordinal number to the requested TPF TP.
4. The requested TPF TP receives the record type and ordinal number.
5. The requested uses the FACE program interface in the remote system to check the validity of the record type and ordinal number.
6. The requested attempts to read the record if they are valid.
7. The requested TPF TP returns a message to the requesting TPF TP with the results of the operation.
8. The requesting TPF TP interprets the returned data and writes a message to the operator based on the reply to indicate the results of the requested TPF TP.
9. The requesting TPF TP then deallocates the conversation.

For the purpose of this sample, assume that the requesting TPF system has a command with the following format:



Also, assume that this command activates the TPF E-type program segment named SMPS. This process is described in “Requesting TPF Transaction Program”.

The TPF TP in the remote requested system is activated as a result of the TPPCC ALLOCATE macro being executed in the local requesting TPF TP. Therefore, the remote requested TPF TP must be defined in the remote TPF system. For the purpose of this sample, assume that the following entry is defined in the transaction program name table (TPNT) on the remote TPF system with the ITPNT macro:

ITPNT MF=L,TPN=VERIFY RECORD TYPE,PGM=SMPR,IS=ANY

If both TPF systems are generated as previously described to accept the ZSAMP command and to define the TP name in the TPNT, either TPF system could be both the **requesting** TPF system and the **requested** TPF system.

The **requesting** TPF TP is described in “Requesting TPF Transaction Program” and the **requested** TPF TP is described in “Requested TPF Transaction Program” on page 116.

## Requesting TPF Transaction Program

In this sample, the **requesting** TPF TP is activated by the command previously described. After parsing the command, the **requesting** TPF TP must allocate a conversation with the **requested** TPF TP by issuing a TPPCC ALLOCATE macro, as

shown in “Sample Requesting TPF Transaction Program” on page 117, at the label `START_CONVERSATION`. The `LUNAME` parameter points to the RVT entry of the LU defining the remote TPF system’s TPF/APPC support package. The `TPN` parameter gives the name of the remote TP as defined in the ITPNT of the remote TPF system. The `RESID` parameter requests that the conversation ID be pointed to by register 3, and the `RCODE` parameter requests that the return code be pointed to by register 4.

**Note:** For these examples, all return codes are assumed to be OK.

After allocating the conversation to the remote requested TPF TP, the requesting TPF TP then builds a message for the requested TP containing the FACE record type and ordinal number that was present on the command. In this example, the message is passed on D0 and consists of the following record:

Field	Length	Value	Description
AM0CCT	2	19	Total message length
AM0TXT	2	14	Logical record length
AM0TXT+2	6	variable	Record type from command
AM0TXT+8	6	variable	Ordinal number from command

Once the message is built on D0, the requesting TPF TP sends the message by issuing the `TPPCC SEND_DATA` macro, as shown in “Sample Requesting TPF Transaction Program” on page 117 at label `SEND_RECORD_TO_REQUESTED_TP`.

The `TPPCC FLUSH` then causes the message to be sent to the remote requested TPF TP. This requesting TPF TP must now change to receive state and wait for the response from the remote requested TPF TP. This requesting TPF TP accomplishes this by issuing the `TPPCC RECEIVE` macro as shown in “Sample Requesting TPF Transaction Program” on page 117 at label `RECEIVE_AND_WAIT_FOR_REPLY`.

This requesting TPF TP instance (in TPF terms, the ECB) is now suspended until the remote requested TPF TP sends the response. Assuming all has g1 well, the `RECEIVE` is satisfied with a return code of OK and a `WHATRCV` indicator of `DATA`. The data message received consists of a logical length field and a 2-byte message that gives the results of the requested TPF TP system’s checks. In our sample, the 2-byte message can be any of the following:

- OK** The record type and ordinal number were valid and the record was successfully read on the remote TPF system.
- NR** The record type was invalid on the remote TPF system.
- NO** The ordinal number was invalid on the remote TPF system.
- IO** The record type and ordinal number were valid, but the record could not be successfully read on the remote TPF system.

The requesting TPF TP now interrogates the response and informs the operator of the results of the action. For example, the logic could be implemented to send a message to the local TPF console as shown in “Sample Requesting TPF Transaction Program” on page 117 at label `PROCESS_REPLY`.

The function is now basically complete, except that the local and remote TPF TPs are still in conversation. The requesting TPF TP ends the conversation by issuing a `TPPCC DEALLOCATE` macro, as shown in “Sample Requesting TPF Transaction

Program” on page 117 at label END\_CONVERSATION. The requesting TPF TP then frees up any other resources that it has end exits.

## Requested TPF Transaction Program

When the requesting TPF TP issues the TPPCC ALLOCATE macro, an ATTACH FMH5 is forwarded to this TPF system. The TPF/APPC support package recognizes the ATTACH as the beginning of a new conversation and assigns a transaction control block identifier (TCB ID), a conversation identifier (CCB ID) and activates the program associated with the transaction program name in the TPF transaction program name table (TPNT). When this TPF program, SMPR in our example, is activated, the TCB ID and CCB ID are passed in the ECB, and the conversation is placed in receive state. This interface is shown in “Sample Requested TPF Transaction Program” on page 121 at label BEGIN\_SAMPLE\_RECEIVE.

No data has been passed to the TP when it is activated, so the requested TPF TP issues the TPPCC RECEIVE macro to request the first data message, as shown in “Sample Requested TPF Transaction Program” on page 121 at label RECEIVE\_AND\_WAIT\_FOR\_REQUEST.

In this sample, the requesting TPF TP is issuing the TPPCC SEND\_DATA macro to forward the FACE record type and ordinal number. If this message has not yet arrived, this ECB is suspended until the message does arrive. Control is returned to the next sequential instruction after the TPPCC verb macro only when there is something available.

When control is returned, this requested TPF TP can use the FACE record type and ordinal number in the data message and perform its processing to check the validity. However, this TPF TP is still in receive state and must wait for the partner requesting TPF TP to change direction and place this TP in send state. Therefore, this requested TPF TP saves the information in the request and again issues a TPPCC RECEIVE macro to wait for the change in direction, as shown in “Sample Requested TPF Transaction Program” on page 121 at label RECEIVE\_AND\_WAIT\_FOR\_SEND\_IND. The change in direction is given to this requested TPF TP when the value in the WHAT\_RECEIVED (WHATRCV) parameter is set to LU62WR\_SEND. Once the send-indication has arrived, this requested TPF TP may then process the original request as shown in “Sample Requested TPF Transaction Program” on page 121 at label PROCESS\_REQUEST.

This requested TPF TP then issues the TPPCC SEND\_DATA macro to forward the results of the process as a reply to the partner requesting TPF TP. The partner requesting TPF TP has issued a TPPCC RECEIVE macro to await the reply.

After sending the reply, this requested TPF TP causes a change of direction by issuing the TPPCC RECEIVE macro and waits for the partner requesting TPF TP to respond.

The requesting TPF TP receives the reply sent by the TPPCC SEND\_DATA macro and continues its processing. It then issues a TPPCC RECEIVE again, and gets the send-indication in the WHATRCVD parameter. Because all processing is complete, the requesting TPF/APPC TP issues the TPPCC DEALLOCATE macro, which causes this requested TPF/APPC TP's RECEIVE verb to be satisfied by passing the deallocation request in the WHATRCVD parameter. This is shown in “Sample Requested TPF Transaction Program” on page 121 at label RECEIVE\_AND\_WAIT\_FOR\_END\_CONV.



When this RECEIVE is satisfied with the deallocation request, the requested TPF TP then issues the TPPCC DEALLOCATE macro with the TYPE=LOCAL option to free up the conversation resources and end the conversation, as shown in "Sample Requested TPF Transaction Program" on page 121 at label END\_CONVERSATION. The requested TPF TP can then issue the TPF EXITC macro.

## Sample Requesting TPF Transaction Program

```
*****
*      THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM "
*      COPYRIGHT = 5748-T12 (C) COPYRIGHT IBM CORP 1979,1988
*      LICENSED MATERIAL - PROGRAM PROPERTY OF IBM
*      REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
*****
      BEGIN NAME=SMPS,VERSION=24,IBM=YES
*
*****
*
* MODULE NAME..... SMPS - (Local TPF/APPC TP)
* RELATED MODULE.. SMPR - (Remote partner TP)
* DOCUMENT NAME... N/A
* DESCRIPTION..... Sample requesting (SEND) TPF/APPC TP
* LEVEL..... N/A
*
* FUNCTION..... This simple sample program illustrates a
*                TPF/APPC transaction program (TP) that
*                requests information from a remote
*                TPF/APPC TP. The command ZSAMP *
*                activates this program.
*
*                This requesting TP allocates a conversation*
*                with a remote TP. This program passes a *
*                FACE record type and ordinal number to the *
*                remote side TPF/APPC TP. The remote TP *
*                then verifies if the passed record type is *
*                valid on the remote TPF system and replies *
*                accordingly. This requesting side then *
*                interprets the response and formats an *
*                operator message based on the response.
*
*                This sample program is intended to show
*                only the format and use of some of the
*                TPPCC verb macros and is NOT intended
*                to be a fully-functional or practical
*                TP.
*
*
*
*
* MODULE ATTRIBUTES..
* TYPE..... 'E' (ECB CONTROLLED)
* ENVIRONMENT... Sample code only
* ENTRY POINT... BEGIN_SAMPLE_SEND
*
*****
* INTERFACE REQUIREMENTS:
*
* INPUT..... ZSAMP #FACER ORDINAL
*          -----
*          I      I--- A six byte ordinal
*          I----- A FACE record type
*
* RESTRICTIONS... This program assumes that the remote
*                TPF system has a entry in its ITPNT
*                table for the requested (RECEIVE side) TP, *
```

```

*                named 'VERIFY_RECORD_TYPE'                *
*                                                         *
*                                                         *
* ECB ON ---> * INPUT..                * OUTPUT..        *
* -----*-----*-----*-----*-----*-----*-----*
* WORK AREA... * N/A                * N/A                *
* DATA LEVELS.. * D0 - Input Zmsg  * N/A                *
* REGISTERS.... * N/A                * N/A                *
*               *                   *                   *
* -----*-----*-----*-----*-----*-----*
* EJECT
*****
*                                                         *
*               Define Data Records Used                  *
*                                                         *
*****
* AM0SG REG=R1          Define Input message format
* TP PCE
* SPACE 3
* STUFF DSECT ,          Define Save Stuff Area
* SFACER DS CL6          Six byte FACE Record Type
* SFACE0 DS CL6          Six byte Face Ordinal number
* SRCODE DS 0CL6         Full 6 byte return code
* SRCODEP DS CL2         2 byte primary return code
* SRCODES DS CL4         4 byte secondary return code
*               DS 0F         Full word alignment
* SRESID DS XL4          Resource (Conversation) ID
* SRTS DS XL1            Request_To_Send_Received Indicator
* SWHAT DS XL1           What_Received Indicator
* USING STUFF,R2         Establish Addressability
* $IS$ CSECT ,          Continue Program CSECT
* EJECT
*****
*                                                         *
*               Begin Mainline logic                      *
*                                                         *
*****
* BEGIN_SAMPLE_SEND DS 0H
* L R1,CE1CR0          Get address of command text
* GETCC D2,L0           Get a block to save stuff in
* LR R2,R14             Get address of block
* MVC SFACER,AM0TXT+4   Save Record Type
* MVC SFACE0,AM0TXT+11 Save Ordinal Number
* RELCC D0              Discard command
* SPACE 3
*****
*                                                         *
*               Allocate a conversation with RECEIVE side TP
*                                                         *
*****
* START_CONVERSATION DS 0H
* TP PCC
*               ALLOCATE,
*               LUNAME=KLUNAME,
*               TPN=KTPNAME,
*               RESID=SRESID,
*               RCODE=SRCODE,
*               SYNC=N1
* CLC SRCODEP,=AL2(LU62RC_OK) Any bad news on ALLOCATE
* BNE TERMINATE_CONVERSATION Yes - terminate the conversation
* SPACE 3
*****
*                                                         *
*               Send the data to the remote TP            *
*                                                         *
*****
* BUILD_RECORD_TO_SEND DS 0H
* GETCC D0,L1          Get a 381 byte block

```

```

X
X
X
X
X

```

```

LR      R1,R14                      Gets its address
XC      0(27,R1),0(R1)              Clear AM0SG header
MVC     AM0TXT+2(L'SFACER+L'SFACE0),SFACER  Move data to be sent
MVC     AM0CCT,=AL2(L'SFACER+L'SFACE0+7) Set count field
MVC     AM0TXT(2),=AL2(L'SFACER+L'SFACE0+2) logical record length
SEND_RECORD_TO_REQUESTED_TP  DS  0H
TPPCC   SEND_DATA,                      X
        RESID=SRESID,                  X
        RCODE=SRCODE,                  X
        RTSRCVD=SRTS
CLC     SRCODEP,=AL2(LU62RC_OK)  Any bad news on SEND_DATA
BNE     TERMINATE_CONVERSATION  Yes - terminate the conversation
CLI     SRTS,LU62_RTSND_RCVYES  Did we get Request_To_Send
BE      TERMINATE_CONVERSATION  Yes - don't honor it
SPACE   3
*****
*          FLUSH the data to be sent          *
*****
TPPCC   FLUSH,                      X
        RESID=SRESID,                  X
        RCODE=SRCODE
CLC     SRCODEP,=AL2(LU62RC_OK)  Any bad news on SEND_DATA
BNE     TERMINATE_CONVERSATION  Yes - terminate the conversation
SPACE   3
*****
*          *
*          Issue RECEIVE and wait for the answer          *
*          *
*****
RECEIVE_AND_WAIT_FOR_REPLY  DS  0H
TPPCC   RECEIVE,                      X
        RESID=SRESID,                  X
        RCODE=SRCODE,                  X
        RTSRCVD=SRTS,                  X
        WHATRCV=SWHAT,                  X
        WAIT=YES
CLC     SRCODEP,=AL2(LU62RC_OK)  Any bad news on RECEIVE
BNE     TERMINATE_CONVERSATION  Yes - terminate the conversation
CLI     SRTS,LU62_RTSND_RCVYES  Did we get Request_To_Send
BE      TERMINATE_CONVERSATION  Yes - don't honor it
CLI     SWHAT,LU62WR_DATACOMPLETE Did we get complete data
BNE     TERMINATE_CONVERSATION  No - terminate the conversation
*          Looks like we got the answer
*****
*          *
*          Process reply received from requested TP          *
*          *
*****
PROCESS_REPLY              DS  0H
L       R1,CE1CR0          Point to block received
SELECT
  WHEN  KOK,=,AM0TXT+2
        WTOPC TEXTA=MOK,SUB=(CHARA,SFACER,CHARA,SFACE0),
        PREFIX=SAMP,NUM=01,LET=I
        X
  WHEN  KNR,=,AM0TXT+2
        WTOPC TEXTA=MNR,SUB=(CHARA,SFACER,CHARA,SFACE0),
        PREFIX=SAMP,NUM=02,LET=I
        X
  WHEN  KNO,=,AM0TXT+2
        WTOPC TEXTA=MNO,SUB=(CHARA,SFACER,CHARA,SFACE0),
        PREFIX=SAMP,NUM=03,LET=I
        X
  WHEN  KIO,=,AM0TXT+2
        WTOPC TEXTA=MIO,SUB=(CHARA,SFACER,CHARA,SFACE0),
        PREFIX=SAMP,NUM=04,LET=I
        X
  OTHERW
        WTOPC TEXTA=MUN,PREFIX=SAMP,NUM=05,LET=I
ENDSEL
RELCC   D0

```

```

SPACE 3
*****
*
*      WAIT for SEND indicator to get into SEND state
*
*****
TPPCC      RECEIVE,
           RESID=SRESID,
           RCODE=SRCODE,
           RTSRCVD=SRTS,
           WHATRCV=SWHAT
           WAIT=YES
CLC      SRCODEP,=AL2(LU62RC_OK)  any bad news ?
BNE      TERMINATE_CONVERSATION  Yes - terminate the conversation
CLI      SWHAT,LU62WR_SEND        Did we get send indication ?
BNE      TERMINATE_CONVERSATION  No - terminate the conversation
*****
*
*      Free up resources and DEALLOCATE
*
*****
END CONVERSATION      DS 0H
TPPCC      DEALLOCATE,
           RESID=SRESID,
           RCODE=SRCODE,
           TYPE=SYNC
CLC      SRCODEP,=AL2(LU62RC_OK)  Any bad news on DEALLOCATE
BNE      TERMINATE_CONVERSATION  Yes - terminate the conversation
RELCC     D2                      Discard save area
EXITC     ,                      All done
SPACE     3
*****
*
*      Terminate conversation on bad return code
*
*****
TERMINATE_CONVERSATION      DS 0H
WTOPC  TEXTA=MCF,SUB=(HEX4A,SRCODEP,HEX4A,SRCODES),
       PREFIX=SAMP,NUM=99,LET=I
SELECT
  WHEN  SRCODEP,=,=AL2(LU62RC_ALLOC_ERROR)
        TPPCC DEALLOCATE,
              RESID=SRESID,
              RCODE=SRCODE,
              TYPE=LOCAL
        X
        X
        X
  OTHERW
        TPPCC DEALLOCATE,
              RESID=SRESID,
              RCODE=SRCODE,
              TYPE=ABENDP
        X
        X
        X
ENDSEL
CRUSA   S0=0
RELCC   D2                      Discard save area
EXITC   ,
EJECT   ,
*****
*
*      Define WTOPC message texts
*
*****
DEFINE_MESSAGES      DS 0H
MOK      DC AL1(43)
          DC C'ORDINAL ..... FOR TYPE ..... OK ON REMOTE'
MNR      DC AL1(29)
          DC C'TYPE ..... INVALID ON REMOTE'
MNO      DC AL1(48)
          DC C'ORDINAL ..... FOR TYPE ..... TOO BIG ON REMOTE'

```

```

MIO      DC  AL1(48)
          DC  C'ORDINAL ..... FOR TYPE ..... I/O ERR ON REMOTE'
MUN      DC  AL1(48)
          DC  C'UNKNOWN RESPONSE FROM REMOTE'
MCF      DC  AL1(34)
          DC  C'CONVERSATION FAILURE .... ..'
*****
*
*          Define Constants Used
*
*****
DEFINE CONSTANTS          DS  0H
KLUNAME  DS  0CL16          Remote FQN
KLUNAMEN DC   CL8'NETID    '      Remote NETID
KLUNAMEL DC   CL8'LUNAME   '      Remote LUNAME
KTPNAME  DS  0CL65          Max area
KTPNAMEL DC  AL1(L'KTPNAMEN)      TP_NAME Length
KTPNAMEN DC  C'VERIFY_RECORD_TYPE'
KOK      DC  C'OK' ==> Record type and ordinal all okay
KNR      DC  C'NR' ==> Record type was invalid in remote system
KNO      DC  C'NO' ==> Ordinal number was too big in remote system
KIO      DC  C'IO' ==> Could not read the record in the remote system
          LTORG ,
          FINIS ,
          END   ,

```

## Sample Requested TPF Transaction Program

```

*****
*          THIS PRODUCT CONTAINS "RESTRICTED MATERIALS OF IBM "
*          COPYRIGHT = 5748-T12 (C) COPYRIGHT IBM CORP 1979,1988
*          LICENSED MATERIAL - PROGRAM PROPERTY OF IBM
*          REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
*****
          BEGIN NAME=SMPR,VERSION=24,IBM=YES
*
*****
*
* MODULE NAME..... SMPR - The sample requested TP
* RELATED MODULE.. SMPS - The sample requesting TP
* DOCUMENT NAME... N/A
* DESCRIPTION..... Sample requested (RECEIVE) TPF/APPC TP
* LEVEL..... N/A
*
* FUNCTION..... This sample TPF/APPC transaction program
*               illustrates a TP that receives a request
*               from a TPF/APPC TP. An ATTACH sent from
*               the requesting side activates this TP.
*
*               This requested side of the TPF/APPC TP
*               receives a FACE record type and ordinal
*               number in the first message from the
*               requesting TP. This TP uses this
*               information to call FACE on THIS TPF
*               system. Upon successful return from FACE,
*               this program attempts to read the record
*               from DASD. The results are returned to the
*               requesting TP as follows:
*               OK ==> Request record could be read
*               NR ==> Record type invalid
*               NO ==> Ordinal number invalid
*               IO ==> Error reading record from DASD
*
*               This sample program is intended to show
*               only the format and use of some of the
*               TPPCC verb macros and is NOT intended
*               to be a fully functional or practical

```

```

*                                     TP.                                     *
*                                                                           *
*                                                                           *
*                                                                           *
* MODULE ATTRIBUTES..                                                    *
* TYPE..... 'E' (ECB CONTROLLED)                                         *
* ENVIRONMENT... Sample code only                                         *
* ENTRY POINT... BEGIN_SAMPLE_RECEIVE                                     *
*                                                                           *
*                                                                           *
*****
* INTERFACE REQUIREMENTS:                                                 *
*                                                                           *
* INPUT..... Standard TPF/APPC ATTACH interface                         *
*                                                                           *
* RESTRICTIONS... This program assumes that there is an                 *
*                  entry in the ITPNT for this TPF system                *
*                  that directs incoming ALLOCATES (ATTACH)             *
*                  to this program. The TP_NAME is                      *
*                  VERIFY_RECORD_TYPE.                                   *
*                                                                           *
*                                                                           *
* Input/Output Interface                                                 *
*   Standard TPF/APPC ATTACH Interface:                                   *
*   ECB      * INPUT..      * OUTPUT..                                   *
*   -----*-----*-----*-----*                                   *
* WORK AREA... * N/A      * N/A      *                                   *
* DATA LEVELS.. * N/A      * N/A      *                                   *
* REGISTERS.... * N/A      * N/A      *                                   *
*               *          *          *                                   *
* -----*-----*-----*-----*                                   *
* EJECT
*****
*                                                                           *
* Define Data Records Used                                               *
*                                                                           *
*****
* AM0SG REG=R1      Define Input message format                         *
* TPPCE ,          Define TPF/APPC values                               *
* DCLREG ,         Define registers for SPM use                         *
* SPACE 3
* STUFF DSECT ,     Define Save Stuff Area                             *
* SFACER DS CL6     Six byte FACE Record Type                         *
* SFACEO DS CL6     Six byte Face Ordinal number                     *
* SRCODE DS 0CL6    Full 6 byte return code                           *
* SRCODEP DS CL2    2 byte primary return code                       *
* SRCODES DS CL4    4 byte secondary return code                     *
*          DS 0F     Full word alignment                             *
* SRESID DS XL4     Resource (Conversation) ID                       *
* SRTS   DS XL1     Request_To_Send_Received Indicator               *
* SWHAT  DS XL1     What Received Indicator                           *
* STCBID DS XL1     TCB_ID                                           *
*          DS 0D
* SDWORD DS D       Double word work area                             *
* SFWORD DS F       Full word work area                               *
* USING STUFF,R2    Establish Addressability                           *
* $IS$ CSECT ,     Continue Program CSECT                             *
* EJECT
*****
*                                                                           *
* Begin Mainline logic (ATTACH Received)                                 *
*                                                                           *
*****
* BEGIN_SAMPLE_RECEIVE DS 0H                                           *
* GETCC D2,L0        Get a block to save stuff in                     *
* LR R2,R14          Get address of block

```

```

MVC  SRESID,EBCCBID  Save RESID value
MVC  STCBID,EBTCBID  Save TCBID value
SPACE 3
*****
*
*          Issue RECEIVE and wait for the message
*
*****
RECEIVE_AND_WAIT_FOR_REQUEST  DS 0H
TPPCC      RECEIVE,
            RESID=SRESID,
            RCODE=SRCODE,
            RTSRCVD=SRTS,
            WHATRCV=SWHAT,
            WAIT=YES
CLC  SRCODEP,=AL2(LU62RC_OK)  Any bad news on RECEIVE
BNE  TERMINATE_CONVERSATION  Yes - terminate the conversation
CLI  SRTS,LU62_RTSND_RCVDYES  Did we get Request_To_Send
BE   TERMINATE_CONVERSATION  Yes - don't honor it
CLI  SWHAT,LU62WR_DATACOMPLETE  Did we get complete data
BNE  TERMINATE_CONVERSATION  No - terminate the conversation
*
*          Looks like we got the message
L     R1,CE1CR0              Point to block received
MVC  SFACER,AM0TXT+2  Save Record Type
MVC  SFACE0,AM0TXT+8  Save Ordinal Number
RELCC D0              Discard message
*****
*
*          Wait for SEND indicator
*
*****
RECEIVE_AND_WAIT_FOR_SEND_IND  DS 0H
TPPCC      RECEIVE,
            RESID=SRESID,
            RCODE=SRCODE,
            RTSRCVD=SRTS,
            WHATRCV=SWHAT,
            WAIT=YES
CLC  SRCODEP,=AL2(LU62RC_OK)  Any bad news on RECEIVE
BNE  TERMINATE_CONVERSATION  Yes - terminate the conversation
CLI  SRTS,LU62_RTSND_RCVDYES  Did we get Request_To_Send
BE   TERMINATE_CONVERSATION  Yes - don't honor it
CLI  SWHAT,LU62WR_SEND        Did we get SEND indicator
BNE  TERMINATE_CONVERSATION  No - terminate the conversation
*****
*
*          Initialize the response block
*
*****
PROCESS_REQUEST  DS 0H
GETCC  D0,L1          Get a block
LR     R1,R14         Point to the msg block
XC     0(27,R1),0(R1)  Clear AM0SG header
MVC    AM0CCT,=AL2(2+2+5)  Set length
MVC    AM0TXT(2),=AL2(2+2)  Dup in LL field
*****
*
*          Call FACE with requested data
*
*****
XC     CE1FM6,CE1FM6    Clear FARW
PACK   SDWORD,SFACE0    Convert Ordinal to packed
CVB    R0,SDWORD        and then binary
LA     R6,SFACER        Point to Record Type
LA     R7,CE1FA6        Point to Return Area
ENTRC  FACS             Call FACE Program
LTR    R0,R0            Check Return Code

```

```

BE      FACE_ERROR          Bad News
FINWC   D6,FIND_ERROR      Read Record and branch on err
MVC     AM0TXT+2(2),KOK     Set response = OK
B       SEND_REPLY
FIND_ERROR          DS 0H
MVC     AM0TXT+2(2),KIO     Set response = IO
B       SEND_REPLY
FACE_ERROR          DS 0H
IF      R7,=,1           If bad record type
      THEN
      MVC AM0TXT+2(2),KNR     Set response = NR
      ELSE
      MVC AM0TXT+2(2),KNO     Set response = NO
ENDIF
*****
*                                           *
*      Send the data to the remote TP      *
*                                           *
*****
SEND_REPLY          DS 0H
TPPCC      SEND_DATA,
           RESID=SRESID,
           RCODE=SRCODE,
           RTSRCVD=SRTS
CLC      SRCODEP,=AL2(LU62RC_OK) Any bad news on SEND_DATA
BNE      TERMINATE_CONVERSATION Yes - terminate the conversation
CLI      SRTS,LU62_RTSND_RCDYES Did we get Request_To_Send
BE       TERMINATE_CONVERSATION Yes - don't honor it
SPACE    3
*****
*                                           *
*      FLUSH data to remote                *
*                                           *
*****
TPPCC      FLUSH,
           RESID=SRESID,
           RCODE=SRCODE
CLC      SRCODEP,=AL2(LU62RC_OK) Any bad news on FLUSH ?
BNE      TERMINATE_CONVERSATION Yes - terminate the conversation
*****
*                                           *
*      Issue PREPARE TO RECEIVE to put remote in SEND state
*                                           *
*****
TPPCC      PREPARE_TO_RECEIVE,
           RESID=SRESID,
           RCODE=SRCODE,
           TYPE=SYNC
CLC      SRCODEP,=AL2(LU62RC_OK) Any bad news ?
BNE      TERMINATE_CONVERSATION Yes - terminate the conversation
*****
*                                           *
*      Issue RECEIVE and wait for DEALLOCATE
*                                           *
*****
RECEIVE_AND_WAIT_FOR_END_CONV DS 0H
TPPCC      RECEIVE,
           RESID=SRESID,
           RCODE=SRCODE,
           RTSRCVD=SRTS,
           WHATRCV=SWHAT,
           WAIT=YES
CLC      SRCODEP,=AL2(LU62RC_DLLOC_NORMAL)
BNE      TERMINATE_CONVERSATION Yes - terminate the conversation
*****
*                                           *
*      Free up resources and DEALLOCATE    *
*                                           *

```



```

*
*****
END_CONVERSATION      DS  0H
TPPCC      DEALLOCATE,
           RESID=SRESID,
           RCODE=SRCODE,
           TYPE=LOCAL
CLC      SRCODEP,=AL2(LU62RC_OK)  Any bad news on DEALLOCATE
BNE      TERMINATE_CONVERSATION  Yes - terminate the conversation
CRUSA    S0=2,S1=0                Discard any blocks
EXITC    ,                        All done
SPACE    3
*****
*
*      Terminate conversation on bad return code
*
*****
TERMINATE_CONVERSATION      DS  0H
WTPC    TEXTA=MCF,SUB=(HEX4A,SRCODEP,HEX4A,SRCODES),
           PREFIX=SAMP,NUM=99,LET=I
TPPCC    DEALLOCATE,
           RESID=SRESID,
           RCODE=SRCODE,
           TYPE=ABENDP
CRUSA    S0=2,S1=0                Discard any blocks
EXITC    ,
EJECT    ,
*****
*
*      Define Constants Used
*
*****
DEFINE_CONSTANTS      DS  0H
KOK      DC  C'OK' ==> Record type and ordinal all okay
KNR      DC  C'NR' ==> Record type was invalid in remote system
KNO      DC  C'NO' ==> Ordinal number was too big in remote system
KIO      DC  C'IO' ==> Couldn't read the record in the remote system
MCF      DC  AL1(34)
           DC  C'CONVERSATION FAILURE .... ..'
LTORG    ,
FINIS    ,
END      ,

```



---

## High-Performance Routing (HPR) Support

High-performance routing (HPR) support is an extension to the SNA Advanced Peer-to-Peer Networking (APPN) architecture. This support increases network performance and throughput, and can dynamically reroute sessions if there are failures in the network. All of this is done by installing new software in existing network components. No new hardware is required.

---

### Benefits of Using HPR Support

The following table shows some of the important benefits of HPR support by comparing it to traditional SNA (PU 5 and base PU 2.1 support).

*Table 5. Comparing Traditional SNA Support to HPR Support*

Traditional SNA	HPR Support
If any node along the LU-LU session path fails, the session fails.	If any node along the LU-LU session path fails, a path switch is automatically performed to obtain a new route for the session. No data is lost during the path switch, no operator intervention is required, and the whole process is transparent to end users.
In PU 5 networking, flow control is end-to-end using fixed-size window-based pacing. In PU 2.1 networking, flow control is performed on a hop-by-hop basis using an adaptive-sized, window-based pacing algorithm.	Flow control is performed on an end-to-end basis using an adaptive method that is time-based rather than window-based. This method is much more effective than window-based flow control mechanisms and is better at reacting to change to prevent network congestion.
Intermediate nodes along the session path have full session awareness. Storage for session control blocks must be allocated on every intermediate node for every session passing through the node.	Intermediate nodes have no session awareness and, therefore, do not require any storage for session control blocks.
There is reliable hop-to-hop delivery. Each node along the path is responsible for making sure that the data has been received by the next node along the route, the data arrives in the correct order, and no duplicate data is sent. If data is too large to be sent across the next hop, the node will segment the data into smaller pieces and the adjacent node will reassemble the data into one message again.	There is reliable end-to-end delivery. Intermediate nodes no longer examine the data. Instead, intermediate nodes simply forward the data to the next node and are not concerned whether the adjacent node receives the data. Intermediate nodes are no longer responsible for detecting lost data, segmentation and reassembly of messages, or any other aspect of reliable delivery. These functions are performed only by the two end points rather than by every node along the path, which greatly improves end-to-end throughput and performance.

Traditional SNA support is connection oriented. The primary weakness of connection-oriented protocols is single points of failure in the network. While protocols that are not connection oriented do not suffer from single points of failure, they lack the correct flow control mechanisms, especially in high-volume networks. HPR support is a mixture of connection-oriented and connectionless protocols. In doing so, the architecture incorporated the best features of both protocol types.

---

### HPR Node Types

There are two types of HPR nodes in the network:

- Automatic network routing (ANR) nodes, which can be only intermediate nodes for a rapid transport protocol (RTP) connection.

- RTP nodes, which can be end points or intermediate nodes for an RTP connection. RTP nodes support the RTP tower and, optionally, control flows over the RTP tower. The TPF system is an RTP node. See “RTP Connections” on page 131 for more information about RTP connections.

The major difference between an ANR node and an RTP node is the amount of work that must be performed. In HPR support, most of the work is performed by the two RTP nodes that are the endpoints of an RTP connection, also known as RTP endpoints. RTP endpoints provide the following functions:

- They detect failures in the network and then start the path switch process to route the RTP connection around the failure. See “Detecting Network Failures” on page 144 for more information.
- They provide end-to-end flow control. See “Flow Control” on page 168 for more information.
- They provide the selective retransmission function, which includes retransmitting specific messages that were lost in the network and requesting that specific messages be retransmitted. See “Selective Retransmission” on page 173 for more information.
- They segment output messages and reassemble input messages when necessary. See “Segmentation and Reassembly” on page 174 for more information.

---

## ANR Labels

In APPN support, a link is identified by its transmission group (TG) number and the control point (CP) name of the adjacent node. In HPR support, a link is identified by an automatic network routing (ANR) label.

ANR labels identify paths through the HPR network. There are two ANR labels assigned to each HPR-capable link, one label for each direction that data flows across the link. The ANR labels are created when an HPR link is activated during the exchange identifier (XID) process.

An ANR label is a 1- to 8-character identifier representing the path from node A to node B across a specific link. Because an ANR label implies direction, there are two ANR labels assigned to each link. When a link between node A and node B is activated, node A assigns the ANR label representing the path from node A to node B across the link. Node B assigns the ANR label representing the path from node B to node A across the link.

ANR labels are unique only per node; they are not necessarily unique in the whole network. Assume node A assigned ANR label 123 to a link connecting node A to node B. Next, a link from node B to node C is activated. Node B can assign ANR label 123 to this link. However, if node A activates a link to node C, node A cannot assign ANR label 123 because node A has already assigned this ANR label to another link.

Data sent on an RTP connection is called a network layer packet (NLP). There is a section at the beginning of each NLP called the network layer header (NHDR), which contains the list of ANR labels representing the path from the origin RTP endpoint to the destination RTP endpoint. Because ANR labels are sent in every NLP, it is desirable to create ANR labels that are as short as possible to reduce the size of NLPs. See “NLPs” on page 145 for more information about NLPs.

A different link identification method (ANR labels) was created for HPR support because:

- The combination of TG number and fully qualified CP name is 17 characters. Sending an extra 17 bytes of data for each hop in every message could cause the message (NLP) size to become very large. ANR labels are typically only 2 or 3 bytes long.
- The identification of a link by HPR support must remain the same for as long as the link is active. In APPN, the TG number and owning CP name can change while the link is active. The ANR labels assigned to a link do not change while the link is active.

Besides having ANR labels assigned to HPR links, there are special ANR labels called *network connection endpoint* (NCE) identifiers. An NCE identifier is assigned to one or more logical units (LUs) in an RTP node. In the TPF system, there is one NCE representing all TPF applications.

Figure 63 shows the APPN and HPR view of a network:

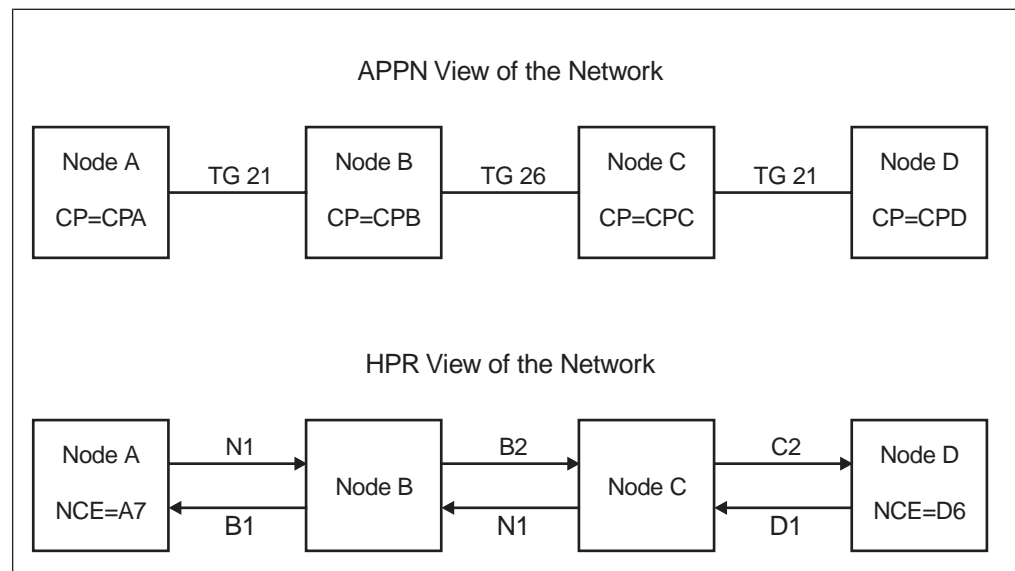


Figure 63. Sample Network with APPN TGs and HPR ANR Labels

In Figure 63, there is one link connecting each of the following node pairs: A and B, B and C, C and D. In APPN terminology, the path from node A to node D is described as follows (assuming all nodes are in the NET1 network):

1. TG 21 to NET1.CPB (node A to node B)
2. TG 26 to NET1.CPC (node B to node C)
3. TG 21 to NET1.CPD (node C to node D).

The reverse path (node D to node A) is:

1. TG 21 to NET1.CPC (node D to node C)
2. TG 26 to NET1.CPB (node C to node B)
3. TG 21 to NET1.CPA (node B to node A).

In the HPR view of the network, there are two lines drawn between each pair of nodes, but the two lines represent one link. Each line is showing the ANR label assigned to that link in the indicated direction. In HPR support, ANR labels describe a route through the network, so the path from node A to node D is:

1. N1 (node A to node B)
2. B2 (node B to node C)
3. C2 (node C to node D)
4. D6 (the NCE of node D where the destination LU resides).

The reverse path (node D to node A) is:

1. D1 (node D to node C)
2. N1 (node C to node B)
3. B1 (node B to node A)
4. A7 (the NCE of node A where the destination LU resides).

Notice that this example uses ANR label N1 more than once in the HPR network.

---

## Activating Links

The sequence of exchange identifier (XID) flows is the same for HPR support as it is for base APPN support. If a node supports HPR, additional information now flows in the XID. Whether an adjacent node supports HPR is determined dynamically during the XID process. The same is true for determining whether the adjacent node supports APPN.

If a node supports HPR, it includes control vector (CV) X'61' in its XID. Bits in CV X'61' indicate whether the node supports the RTP tower and control flows over the RTP tower. The ANR label assigned by this node to the link is also included in CV X'61'.

A link is HPR capable if, and only if, the XIDs sent by both sides include CV X'61'. If one or both XIDs do not include CV X'61', the link is base APPN and is not HPR capable.

If HPR support is enabled on the TPF processor where the link is being activated, CV X'61' is included in the XID sent by the TPF system. See “Installation and Tuning” on page 176 for more information about how to enable HPR support in the TPF system.

A link that supports APPN, but does not support HPR, is limited to base APPN flows, meaning that only format identification 2 (FID2) path information units (PIUs) can flow across this link. An HPR-capable link supports not only FID2 PIUs, but also NLPs.

Because the TPF system is an APPN end node (EN), whenever a new link is activated, the TPF system sends a topology database update (TDU) request to register the new link in the APPN network. The TDU now includes HPR support information.

Use the ZNAPN command with the TOPOLOGY parameter specified to display information about the active PU 2.1 links connected to the TPF loosely coupled complex. The information displayed indicates whether the link supports HPR and, if so, whether the link connects to an RTP node or an ANR node. See *TPF Operations* for more information about the ZNAPN command.

---

## RTP Connections

An RTP connection is a logical pipeline between two RTP nodes over which one or more LU-LU sessions exist. The two RTP nodes are the RTP endpoints for the RTP connection, and these two nodes are not necessarily adjacent. RTP connections are based on SNA class of service (COS), meaning all LU-LU sessions for a given RTP connection have the same COS. Between a pair of RTP nodes, there can be multiple RTP connections, each with the same COS or a different COS.

At any point in time, an RTP connection is in one of the following states:

- CONNECTED** This is the normal state of an RTP connection. Data traffic is flowing between the RTP endpoints.
- MOVING** The TPF system is attempting to find a better route for the RTP connection because the ZNRTP SWITCH command was entered. Data traffic continues to flow on the current route. See *TPF Operations* for more information about the ZNRTP SWITCH command.
- RESYNC** The RTP connection resynchronization process is in progress for the RTP connection. See “RTP Connection Resynchronization Process” on page 163 for more information about the RTP connection resynchronization process.
- STARTING** The RTP connection is being activated and the ROUTE\_SETUP process is still in progress. See “ROUTE\_SETUP Process” on page 136 for more information about the ROUTE\_SETUP process.
- SWITCHING** A path switch is in progress because the TPF system detected a failure in the network. No data is flowing. See “Path Switches” on page 141 for more information about the path switch process.

If only part of the network supports HPR, it is possible for an RTP connection to be established for only the HPR-capable part of the network and have the LU-LU session extend beyond that point.

Figure 64 on page 132 shows an example of whether an RTP connection can be started and, if so, how far the RTP connection goes based on the levels of HPR support in the nodes in the network:

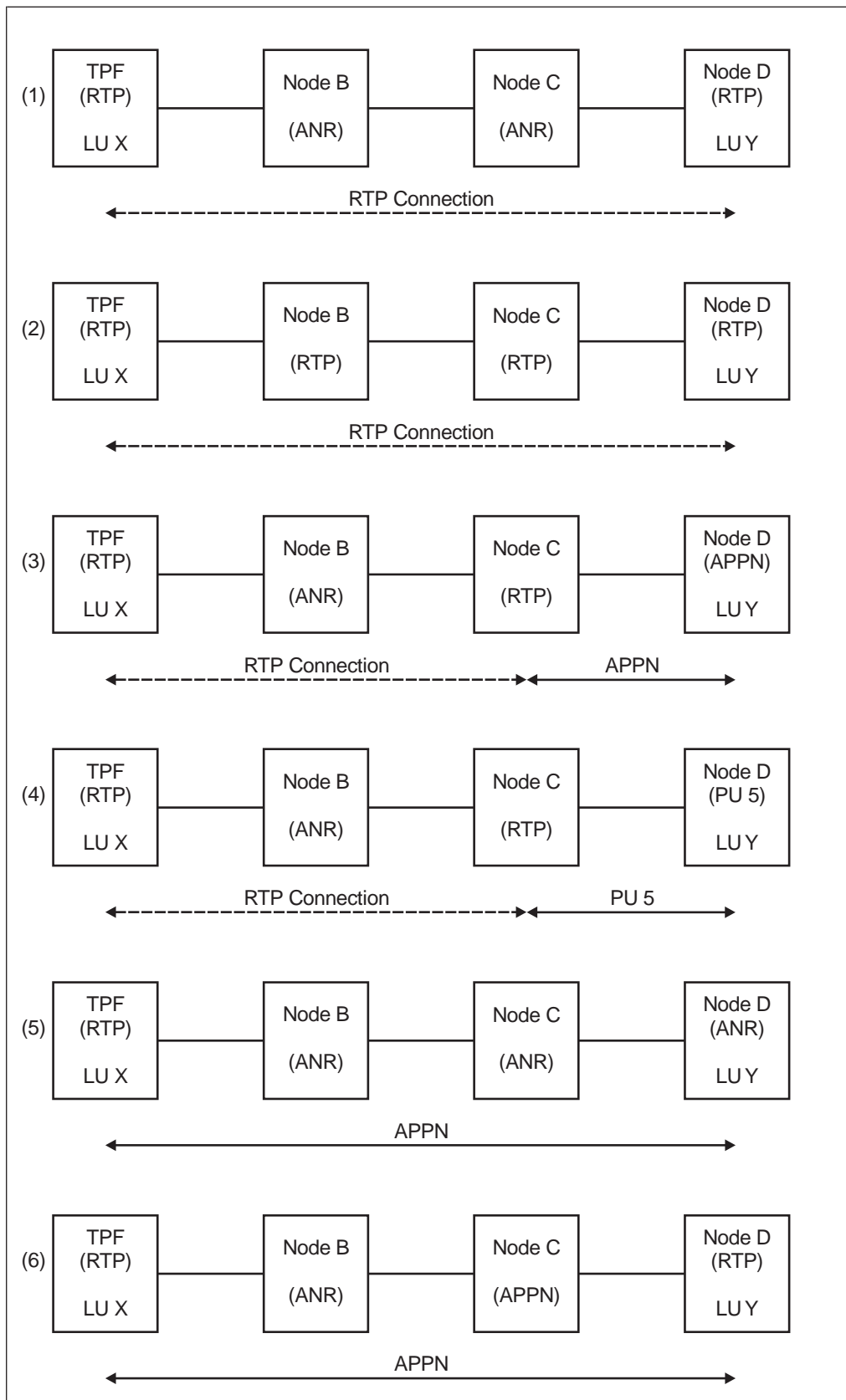


Figure 64. Sample Networks and RTP Connections



In Figure 64 on page 132, LU X in the TPF system is in session with LU Y in node D across a three-hop route. The capabilities of the nodes along the path determine if HPR support can be used and, if so, for how much of the session route. The following six examples are shown in Figure 64 on page 132:

1. All intermediate nodes support HPR, and the node containing the destination LU (LU Y in node D) not only supports HPR, but is an RTP node. This means that you can use HPR support for the entire session route, and the RTP connection will be from the TPF system to node D.
2. All intermediate nodes support HPR, and the node containing the destination LU is an RTP node. Again, you can use HPR support for the entire session route. Even though nodes B and C are RTP nodes, they function like ANR nodes in this example.
3. Node D supports only APPN; it does not support HPR. This means that you cannot use HPR support for the entire path. However, you can use HPR support for part of the path because one intermediate node (node C) is an RTP node, and all nodes between that node and the TPF system support HPR. In this example, the RTP connection exists between the TPF system and node C. Base APPN flows are used between nodes C and D.
4. Node D does not support APPN or HPR support. In this example, the RTP connection exists between the TPF system and node C. PU 5 flows are used between nodes C and D.
5. Even though all nodes support HPR, you cannot use HPR in this example because none of the intermediate nodes or the destination node are an RTP node. No RTP connection can be established; therefore, base APPN flows are used for the entire route.
6. Even though the destination node is an RTP node, you cannot use HPR support in this example because node C does not support HPR. Flows in and out of node C must be base APPN.

The following example shows an RTP connection that exists for only part of the LU-LU session path:

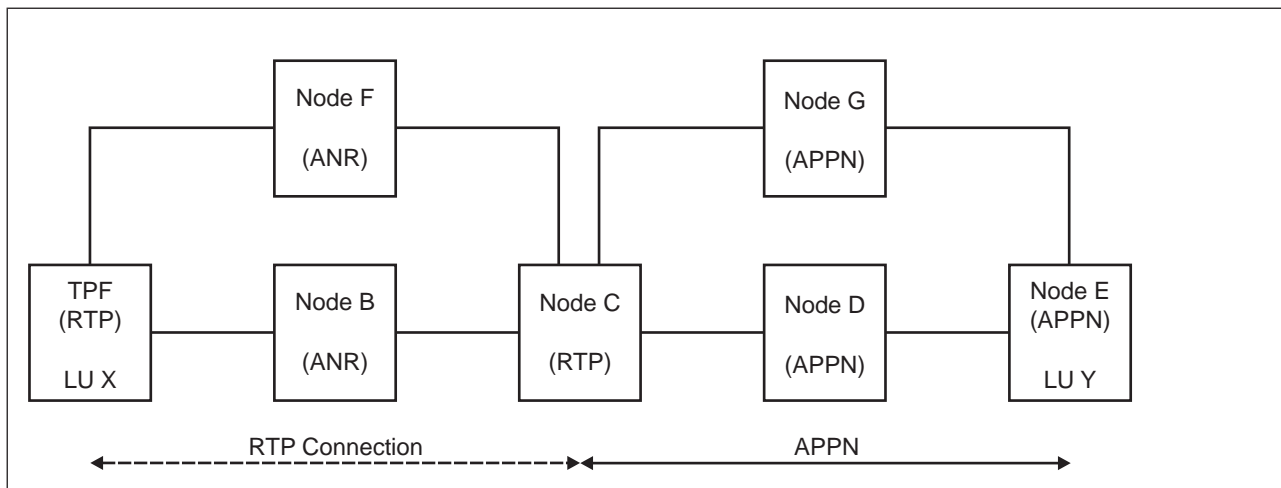


Figure 65. Combination of HPR Support and APPN

“Starting RTP Connections” on page 134 discusses how to determine if you can use HPR support and, if so, for how much of the session route. You get the benefits of HPR support only on the HPR part of the route (the RTP connection). For example, in Figure 65 there is a session between LU X in the TPF system and LU Y in node E. Assume the path goes from the TPF system to node B, then to node C, then to

node D, and then to node E. The RTP connection is between the TPF system and node C. If node B fails, the RTP connection would automatically switch and go through node F (rather than node B). The session between LU X and LU Y would remain active. However, if node D fails, the LU-LU session fails even though another path exists between nodes C and E (through node G).

## TCIDs

Because an RTP node can have several RTP connections active at the same time, a token called a *transport connection identifier* (TCID) is assigned to uniquely identify an RTP connection. Each RTP endpoint assigns a TCID to each RTP connection. Therefore, there are two TCIDs assigned to each RTP connection. For example, assume there is an RTP connection from node X to node Y. The TCID assigned by node X is TCID<sub>yx</sub> and the TCID assigned by node Y is TCID<sub>xy</sub>. Each message sent on an RTP connection contains one TCID in the transport header (THDR) section of the NLP. Whenever possible, the sending node should use the TCID that the remote RTP endpoint assigned. When node X sends NLPs to node Y, TCID<sub>xy</sub> should be placed in the THDR section of the NLP. This is done for performance reasons because a node should generate a TCID in such a way that it can be used as an index or pointer into control blocks defined in that node.

Just like ANR label generation, TCIDs are only unique per node, not in the whole network.

See “NLPs” on page 145 for more information about NLPs. See “THDRs” on page 149 for more information about the format of the THDR and how it is used.

## Starting RTP Connections

Unlike links and sessions, which you can start yourself, you can never start an RTP connection. Instead, RTP connections are always started automatically by RTP nodes during the LU-LU session activation process. See “Starting LU-LU Sessions” on page 135 for more information about how RTP connections are started.

## Deactivating RTP Connections

RTP connections are deactivated automatically by one of the RTP endpoints when the connection is no longer being used. This happens some time after the last LU-LU session that was using the RTP connection ends. The TPF system delays ending the RTP connection in case a new LU-LU session is started that can use that RTP connection.

How long a node waits to deactivate the RTP connection depends on the implementation. In the TPF system, the amount of time to wait is based on the value of the *alive timer*. See “Alive Timer” on page 145 for more information about the alive timer.

You can deactivate an RTP connection yourself at any time by using the ZNRTP INACT command. The remote operator can deactivate an RTP connection at any time as well. When you deactivate an RTP connection, it is immediate and unconditional. The RTP connection is cleaned up and all LU-LU sessions that were using the RTP connection are also cleaned up. See *TPF Operations* for more information about the ZNRTP INACT command.

## Displaying RTP Connections

To display information about RTP connections:

- Use the ZNRTP DISPLAY command to display information about one or more RTP connections.
- Use the ZNRTP ROUTE command to display APPN and HPR route information for an RTP connection.
- Use the ZNRTP SUMMARY command to display current and high-water mark information about resources used by RTP connections.
- Use the ZNDLU command to display all the LU-LU sessions using a particular RTP connection.
- Use the ZNMON command with the ALS parameter specified to display the number of RTP connections and HPR LU-LU sessions currently active in the TPF system.

See *TPF Operations* for more information about the ZNRTP, ZNDLU, and ZNMON commands.

---

## Starting LU-LU Sessions

HPR support does not change the existing procedures that are used to start LU-LU sessions. The APPN search (LOCATE command flows on the CP-CP sessions) is still used to calculate the route for the LU-LU session. Once a route has been calculated, it is passed to the node that owns the primary logical unit (PLU) in the route selection control vector (RSCV), which is CV X'2B'. The RSCV contains a hop-by-hop list of the path through the network from the PLU to the secondary logical unit (SLU).

With HPR support, additional information is included in the RSCV for each hop along the route whether the link (hop) supports HPR or not and, if the link supports HPR, whether it is connected to an RTP node or not.

By examining the RSCV, the node that contains the PLU determines if HPR support can be used for this LU-LU session and, if so, how much of the route can use HPR support. This is done automatically and does not require any new network definitions or new parameters on operator messages to use HPR support. On a session by session basis, the system software determines if HPR support can be used.

In base APPN, LU-LU session initiation involves two key steps:

1. One or more LOCATE commands flow. Eventually, a LOCATE command containing the RSCV will be sent to the node containing the PLU.
2. The PLU sends out the BIND request (as a FID2 PIU) to start the LU-LU session.

With HPR support, additional processing is done between steps 1 and 2. The RSCV is examined to see if HPR support can be used. If HPR support cannot be used, there is no change to the existing processing. A FID2 BIND request is sent out.

If the RSCV indicates that HPR support can be used, the node containing the PLU selects the RTP connection to use for this LU-LU session. How this is done depends on the implementation, but basically involves either starting a new RTP connection or using one of the existing RTP connections. No matter what, the first step is to determine how far HPR support can be used along the LU-LU session route or, in other words, how far the RTP connection will go. The node where the RTP connection ends is referred to as the *remote RTP endpoint*. If the RTP

connection goes the entire route from the PLU to the SLU, the remote RTP endpoint is the node that contains the SLU. Otherwise, HPR support is being used for only part of the route, and the SLU resides in a node beyond the remote RTP endpoint.

If an LU-LU session is being started, the PLU resides in the TPF system, and HPR support can be used for at least part of the route, the TPF system selects which RTP connection to use. The following describes how the TPF system selects the RTP connection to use:

1. The remote RTP endpoint (node Y) is determined by examining the RSCV.
2. The table of active RTP connections is searched to see if any of the RTP connections can be used. For an existing RTP connection to be a candidate, all of the following conditions must be true:
  - The remote RTP endpoint of the RTP connection is node Y.
  - The current route of the RTP connection is the same as that of the route in the RSCV of the new LU-LU session up to node Y.
  - The class of service (COS) of the RTP connection is the same as the COS of the LU-LU session being started.
3. If none of the existing RTP connections are candidates for use, the TPF system starts a new RTP connection.
4. If one or more existing RTP connections are candidates for use, the Select an RTP Connection user exit (URTP) is called. URTP can either select one of the existing RTP connections to use or select to have a new RTP connection started. See *TPF System Installation Support Reference* for more information about URTP.

## ROUTE\_SETUP Process

When an RTP connection starts, there is a new flow called a ROUTE\_SETUP command that occurs after the APPN search (LOCATE flows) and before the BIND request is sent out. The purpose of the ROUTE\_SETUP process is to gather the HPR information for the route, including:

- The list of ANR labels in the forward direction (the path from the node containing the PLU to the remote RTP endpoint)
- The list of ANR labels in the reverse direction (the path from the remote RTP endpoint to the node containing the PLU)
- The maximum packet size, which is the value of the smallest link size along the route between the RTP endpoints.
- The maximum amount of time that the remote RTP endpoint requires for a path switch.

The ROUTE\_SETUP request is sent by the node containing the PLU and is processed by every node along the route, up to and including the remote RTP endpoint, to obtain the forward route information. When the ROUTE\_SETUP request reaches the remote RTP endpoint, a ROUTE\_SETUP reply is sent back to gather the reverse route information.

Figure 66 on page 137 shows an example of the flows during the ROUTE\_SETUP process.

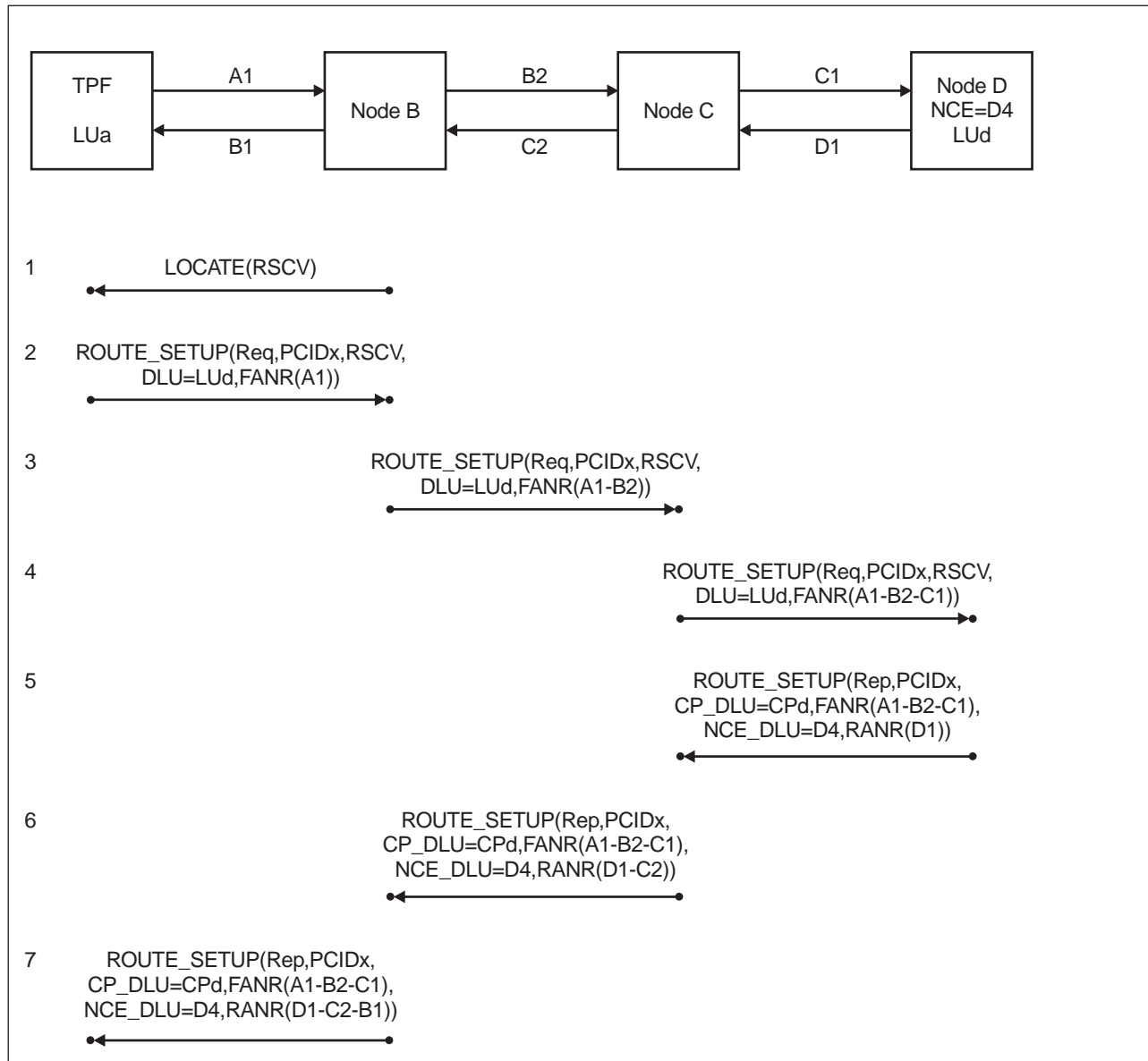


Figure 66. ROUTE\_SETUP Process

In the figure, a session between LUa (the PLU) in the TPF system and LUD in node D is starting. The RSCV calculated by the network is a three-hop route, the TPF system determined that HPR support can be used for the entire route, and a new RTP connection will be started. The step-by-step description of the ROUTE\_SETUP process is as follows:

1. The TPF system receives a LOCATE command on the CP-CP sessions containing the RSCV.
2. The TPF system builds the ROUTE\_SETUP request; the following information is included:

**Req** The ROUTE\_SETUP command is marked as a request.

**PCIDx** A new procedure correlation identifier (PCID) to identify this ROUTE\_SETUP process. This is not the PCID of the LU-LU session being started.

**RSCV** The route for the LU-LU session.

**DLU** Name of the destination LU, which, in this example, is LUd.

**FANR** Forward ANR field. The TPF system includes the ANR label for the first link (which is A1) here. Each node along the route will add to this field.

3. Node B adds ANR label B2 to the FANR field and then, using the RSCV, passes the ROUTE\_SETUP request to node C.
4. Node C adds ANR label C1 to the FANR field and then, using the RSCV, passes the ROUTE\_SETUP request to node D.
5. Node D builds the ROUTE\_SETUP reply; the following fields are included:

**Rep** The ROUTE\_SETUP command is a reply.

**PCIDx** The PCID from the ROUTE\_SETUP request.

**CP\_DLU**

CP name of the remote RTP endpoint (CP name of node D).

**FANR** Forward ANR field. This is copied from the ROUTE\_SETUP request.

**NCE\_DLU**

NCE identifier assigned to the destination LU in the remote RTP endpoint, which is D4 in this example.

**RANR** Reverse ANR field. The remote RTP endpoint puts ANR label D1 here. Each node along the route will add to this field.

6. Because the ROUTE\_SETUP command is a reply, intermediate nodes add to the RANR field rather than the FANR field. Node C adds ANR label C2 to the RANR field.
7. Node B adds ANR label B1 to the RANR field.

When the ROUTE\_SETUP reply is received by the TPF system, the RTP connection is started by sending out an NLP marked as a new connection. The NLP also contains the BIND request to start the new LU-LU session.

The ROUTE\_SETUP process is also used during the path switch process. See “Path Switches” on page 141 for more information about the path switch process.

## LU-LU Session Activation Flows

If an LU-LU session is starting and an existing RTP connection is used, the ROUTE\_SETUP process is skipped and a BIND request is sent in an NLP instead. The following examples show the flows in and out of the TPF system during LU-LU session activation based on whether HPR support is used or not:

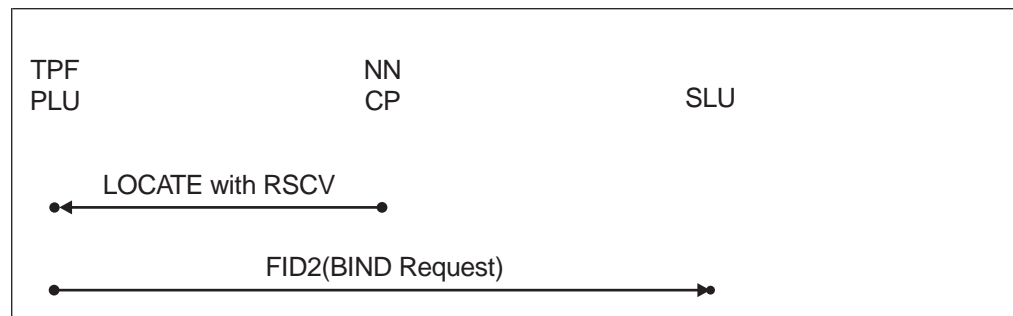


Figure 67. LU-LU Session Activation, HPR Support Is Not Used

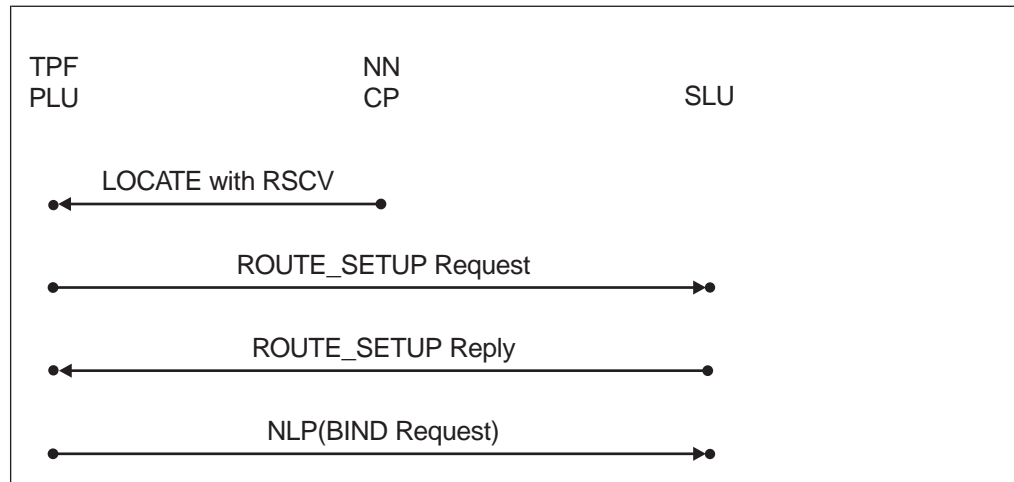


Figure 68. LU-LU Session Activation, a New RTP Connection Is Started

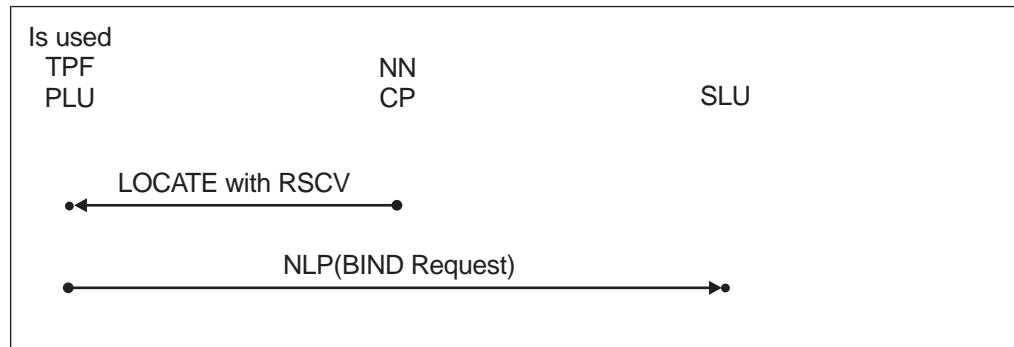


Figure 69. LU-LU Session Activation, an Existing RTP Connection

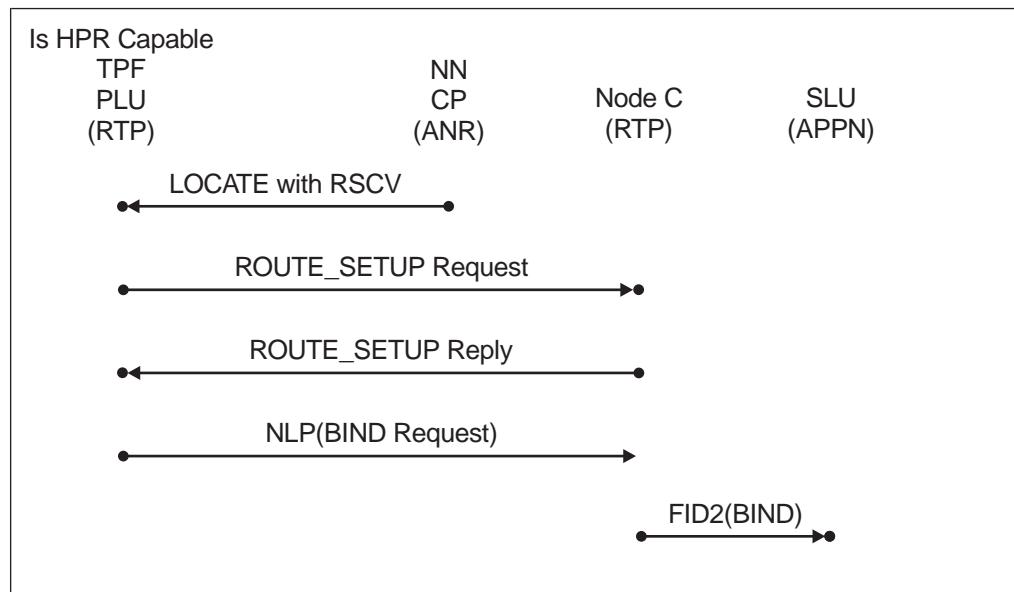


Figure 70. LU-LU Session Activation, Part of the Route

# Session Addresses

For an LU-LU session in a PU 5 network, the combination of the network address (NA) of the PLU with the NA of the SLU uniquely identifies the session. The NAs flow in the transmission header (TH) section of an FID4 PIU.

For an LU-LU session in a PU 2.1 network, a session identifier (SID) together with the origin destination assignment indicator (ODAI) indicator uniquely identify the session between a pair of adjacent nodes. The SID and ODAI indicator flow in the TH section of an FID2 PIU.

For an LU-LU session in an HPR network, a session address (SA) uniquely identifies the session. The SA flows in the TH section of an NLP.

An SA is an 8-byte token that uniquely identifies an LU-LU session in the HPR network. An SA is unique per RTP connection, not per node. There are two SAs assigned to an LU-LU session, one assigned by each RTP endpoint. One SA flows in the TH section of an NLP. Whenever possible, the SA that the remote RTP endpoint assigned is placed in the TH. For example, if node X is sending data to node Y, the SA assigned by node Y is placed in the TH of the NLP.

The reason for having two SAs assigned to an LU-LU session is the same reason that two TCIDs are assigned to an RTP connection. The intention is that a node will create an SA in such a way that it can be used as an index or pointer to control blocks defined in that node.

The SA assigned by the RTP endpoint that owns the PLU is sent in the TH section of the NLP containing the BIND request. The SA assigned by the RTP endpoint that owns the SLU is sent in the BIND response in a new control vector, CV X'62'.

The following figure shows an example of how session addresses are used:

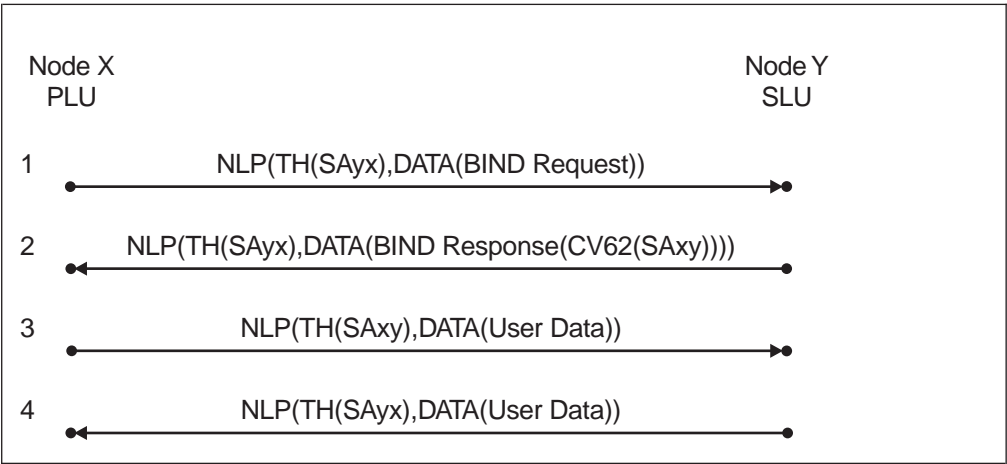


Figure 71. Session Address Usage

In Figure 71:

1. Node X creates a new SA called SA<sub>yx</sub> and sends the BIND request to the SLU across the selected RTP connection. The TH of the NLP contains the SA assigned by node X because the SA assigned by the remote RTP endpoint (node Y) is not known yet.



**Note:** A bit in the SA indicates whether the sending or receiving node assigned the SA.

2. Node Y accepts the BIND request and assigns an SA of its own called SA<sub>Y</sub>, which is appended to the BIND response in CV X'62'.

**Note:** The TH of the NLP contains the SA assigned by node X.

3. Once the BIND response has been received, node X now knows the SA assigned by node Y and will put that SA (SA<sub>Y</sub>) in the TH of every subsequent NLP sent on this LU-LU session.
4. The RTP endpoint owning the SLU (node Y) always knows the SA assigned by the RTP endpoint that owns the PLU and, therefore, always puts SA<sub>Y</sub> in the TH of NLPs sent on this LU-LU session.

See “NLPs” on page 145 for more information about NLPs.

---

## Path Switches

When an RTP connection is started, a specific route in the network is assigned to that RTP connection. The RTP connection will use this route until a problem with the route is detected; for example, a node along the route fails. A path switch is the process by which an RTP connection changes its route. The path switch is non-disruptive, meaning no sessions or data is lost, and is transparent to the end user.

The path switch process is started automatically when a problem with the current route is detected. See “Detecting Network Failures” on page 144 for more information about how problems in the network are detected.

You can start the path switch process yourself at any time by issuing the ZNRTP SWITCH command. When you enter the ZNRTP SWITCH command, the APPN network is searched to see if a better route exists. While the search is taking place, the RTP connection continues to use its existing route. If the search returns a better route, the path switch process continues and the RTP connection is switched over to that better route. However, if the search fails or indicates that the RTP connection is already using the best route, the path switch process ends and the RTP connection continues to use its existing route. See *TPF Operations* for more information about the ZNRTP SWITCH command.

Regardless of what caused a path switch to start, the TPF operator is notified when a path switch ends successfully or fails. The fact that a path switch started usually indicates a problem in the network that needs to be investigated. This is true regardless of the outcome of the path switch.

A path switch is done at the RTP connection level; that is, when a path switch is done, all LU-LU sessions on that RTP connection switch to the new route. A path switch is not done for individual LU-LU sessions.

## Path Switch Process

The path switch process is very similar to the LU-LU session activation process; that is, an APPN search is performed, followed by the ROUTE\_SETUP process. The LOCATE command that flows during the APPN search is marked as *search only* because a new LU-LU session is not being started. Normally, LOCATE commands flow only during the LU-LU session activation process and do so to calculate a route between the two LUs. During a path switch, the APPN search,

instead, calculates a new route between the two RTP endpoints. The search origin and destination fields in the LOCATE command are set to the control point (CP) names of the RTP endpoints.

If the APPN search finds a new route connecting the RTP endpoints, the ROUTE\_SETUP process is performed to gather the HPR information for the new route. When the ROUTE\_SETUP process ends, an NLP is sent along the new route to the remote RTP endpoint. A Switching Information (SI) segment, which contains information about the new route, is included in the NLP. The presence of an SI segment in an NLP indicates that a path switch has taken place. See “ROUTE\_SETUP Process” on page 136 for more information about the ROUTE\_SETUP process.

The following figure shows an example of the flows during the path switch process:

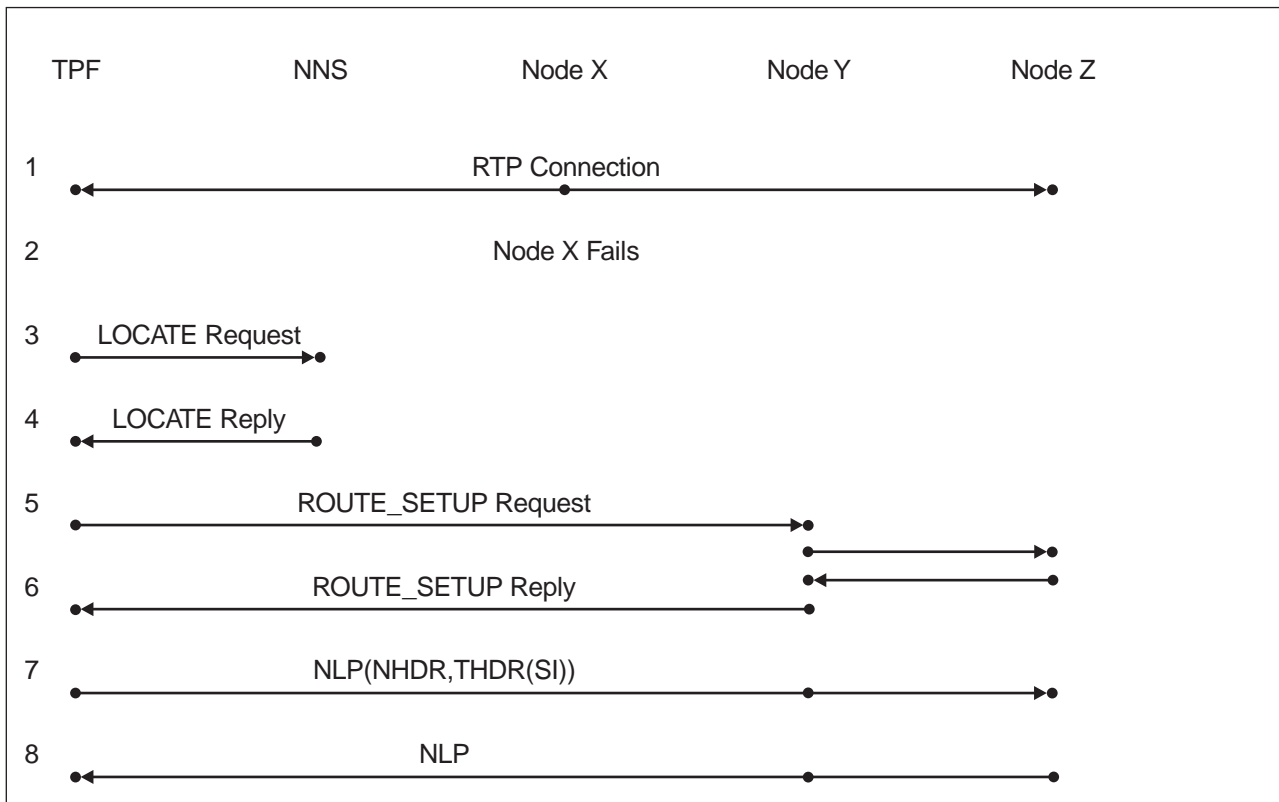


Figure 72. Path Switch Process

In Figure 72:

1. An RTP connection exists between the TPF system and node Z. The route is through node X.
2. Node X fails.
3. The TPF system detects the failure in the network and sends a LOCATE request on the CP-CP sessions to start the path switch process.
4. The LOCATE reply is received and contains a new route between the TPF system and node Z. The new route is through node Y.
5. The TPF system builds the ROUTE\_SETUP request and sends it to node Y. Node Y adds information to the ROUTE\_SETUP request and passes it to node Z.

6. Node Z builds the ROUTE\_SETUP reply and sends it to node Y. Node Y adds information to the ROUTE\_SETUP reply and passes it to the TPF system.
7. The TPF system sends an NLP to node Z along the new route. The NLP includes the SI segment, which tells node Z the new route to use from now on.
8. NLPs sent by node Z now use the new route (through node Y).

## Path Switch Timer

The path switch timer is started when the RTP endpoint begins the path switch process. The timer continues to run until the path switch process ends successfully, which is signaled by the receipt of an NLP from the remote RTP endpoint acknowledging the path switch. The function of the path switch timer is to detect when the remote RTP endpoint has failed or when no working path exists to the remote RTP endpoint.

The path switch timer (and path switch process) is stopped if an NLP is received on the RTP connection on the existing route. This can happen when network congestion delays the arrival of the status reply.

You can define the value of the path switch timer. When an RTP connection is started, each RTP endpoint informs its partner RTP endpoint how long that node requires to complete a path switch. The HPRPST parameter on the SNAKEY macro in CTK2 defines how long the TPF system tells the remote RTP endpoint to wait when that node starts a path switch before considering that the path switch has failed. You can also use the ZNKEY command with the HPRPST parameter specified to specify a new value for the path switch timer. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro. See *TPF Operations* for more information about the ZNKEY command.

**Note:** Once the value of the path switch timer is set for an RTP connection, that value is used for the life of that RTP connection. Therefore, if you specify a new value for the path switch timer, that new value is used only for new RTP connections that are started. Existing RTP connections continue to use the original value of the path switch timer.

The logic for choosing the value of the HPRPST parameter is the same as the logic that is done today for determining an appropriate value for the automatic network shutdown (ANS) parameter in the NCP generation deck. For example, if you want PU 5 or PU 2.1 LU-LU sessions to remain active across an IPL of the TPF system, you must set the ANS parameter in the NCP to a value higher than the time it takes the TPF system to complete the IPL. If you want RTP connections (and, therefore, HPR LU-LU sessions) to remain active across an IPL of the TPF system, you need to set the HPRPST parameter to a value higher than the time it takes the TPF system to complete the IPL.

## Stationary and Mobile RTP Nodes

An RTP endpoint can be defined as a stationary or mobile node. Most nodes are stationary, but the TPF system is defined as a mobile node to avoid having the network flooded with path switch requests if the TPF system is dumping storage for a long time or performs an IPL. When the short request timer expires and a path switch is needed, the path switch timer is started, but that node does not necessarily send a request to the network to ask for a new path. If the remote RTP endpoint is a stationary node, the local RTP endpoint that detected the network failure will send a request to the network right away to ask for a new path. However, if the remote RTP node is mobile, the local RTP node (if it is stationary) does not send a request for a new path to the network even though its path switch

timer is running. When this occurs, it is up to the remote mobile node to do the path switch if there was an actual failure in the network. If the path switch timer expires and the node has not asked for a new path, it will ask the network for a new path at this time as a last chance effort to keep the RTP connection active.

See “Short Request Timer” for more information about the short request timer. See “Path Switch Timer” on page 143 for more information about the path switch timer.

To understand why having the TPF system defined as a mobile node is important, assume that there are 100 000 RTP connections with the TPF system and a system error is taken that lasts for 10 seconds. During that 10-second interval when the TPF system is not sending or receiving data from the network, the remote RTP endpoints will detect that there is a problem and start their path switch timers. When the dump ends and data traffic resumes, the remote RTP endpoints will receive data from the TPF system and stop their path switch timers. If the TPF system was defined as a stationary node, the network would be flooded with 100 000 path switch requests while the TPF system was dumping. This would cause unnecessary high-priority messages (APPN LOCATE commands) to flow and result in the delay of processing data traffic.

---

## Detecting Network Failures

When the route being used by an RTP connection fails, one of the RTP endpoints will detect the failure and start the path switch process. How and when the TPF system detects failures in the HPR network can be broken into two categories:

- An adjacent link station (ALS) directly connected to the TPF system fails and RTP connections were going through that ALS. This immediately starts the path switch process for each RTP connection going through the failing ALS.
- A node farther out in the network fails and RTP connections were going through that node. When this happens, the TPF system will detect the network failure because acknowledgements from the remote RTP endpoint will not be received. How long it takes to detect the failure is based on the short request timer and, possibly, the alive timer.

## Short Request Timer

A short request timer exists for each RTP connection. Whenever a status request is sent to the remote RTP endpoint, the short request timer is started. Its purpose is to detect failures in the network. Whenever a status reply is received, the short request timer is stopped. If no status reply is received and the short request timer expires, the TPF system sends another NLP with no data to verify the route and the short request timer is restarted. If the short request timer expires two consecutive times before a status reply is received, a path switch is started.

The value of the short request timer is based on the *smoothed round-trip time* (SRTT) of the RTP connection. The SRTT is the average time it takes the TPF system to receive a status reply on the RTP connection after sending a status request. Each time a status reply is received, the SRTT for the RTP connection is adjusted to take into account changing network conditions.

When the short request timer is started, it is set to a value larger than the SRTT to allow for normal network delays and congestion. This is particularly important when the SRTT is a very low number; for example, in the millisecond range.

## Alive Timer

The primary purpose of the alive timer is to detect network failures for idle RTP connections. If the TPF system has not sent or received data on an RTP connection for a given period of time, a status request will be sent in an HPR control message to verify that the route is still active. This is also referred to as a *heartbeat message*. Because the control message contains a status request, the short request timer is started. Normal short request timer processing takes over from this point. If no status reply is received, the TPF system starts a path switch.

The goal of sending these heartbeat messages is to detect network failures in a timely way. For example, if a node along the route of an active RTP connection fails, the next data message for this RTP connection is not going to be sent for several minutes. Without the alive timer, the network failure would not be detected until that first data message is sent minutes later. Processing of that data message would be delayed until a path switch was started and completed successfully.

The other purpose of the alive timer is to keep limited resource links active. If there are limited resource links along the route of an RTP connection, traffic must be sent across that link every so often; otherwise, the intermediate nodes in the network will deactivate the link thinking that there are no sessions using it. Intermediate nodes have no awareness of HPR LU-LU sessions.

The HPRALIVE parameter on the SNAKEY macro in CTK2 defines the alive timer value used by the TPF system. You can also use the ZNKEY command with the HPRALIVE parameter specified to define a value for the alive timer. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro. See *TPF Operations* for more information about the ZNKEY command.

---

## NLPs

In PU 5 and PU 2.1 networks, a message (either an SNA command or user data) flows in a path information unit (PIU). In HPR support, a message flows in a network layer packet (NLP), which is an extension to the existing PIU format.

Figure 73 on page 146 compares the parts of a PIU to the parts of an NLP.

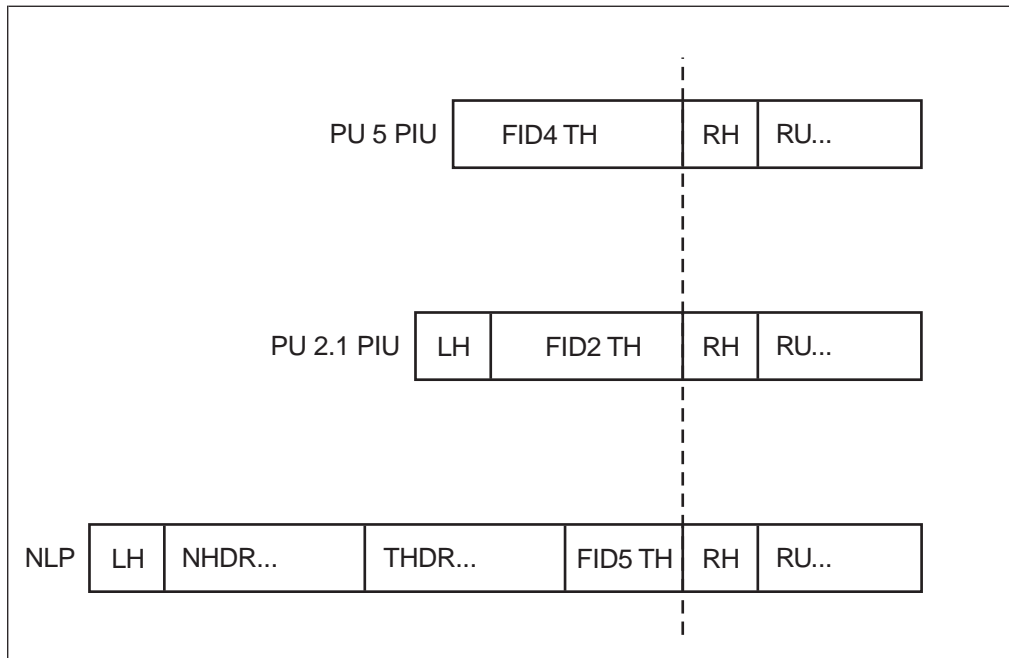


Figure 73. Comparing an NLP to PIUs

The following information describes the parts of a PIU and an NLP, and how these parts are used:

- Link Header (LH)
  - Exists only in PU 2.1 PIUs and NLPs
  - Same format for both PU 2.1 PIUs and NLPs
  - Indicates the size of the PIU or NLP.
- Network Layer Header (NHDR)
  - Exists only in an NLP
  - Used to route the NLP from one RTP endpoint to the other
  - Variable length.
- Transport Header (THDR)
  - Exists only in an NLP
  - Identifies the RTP connection
  - Optionally contains control information about the RTP connection
  - Variable length.
- Transmission Header (TH)
  - Identifies the session
  - Format (FID type) is different based on network type.
- Request Header (RH)
  - Identifies the type of data in the RU
  - Same format regardless of network type.
- Request Unit (RU)
  - Either an SNA command or user data
  - Same format regardless of network type
  - Variable length.

In a PU 5 or PU 2.1 network, the TH is required in a PIU, but the RH and RU are optional; therefore, possible combinations of a PIU are:

- TH only (PU 5 virtual route pacing response)
- TH and RH, but no RU
- TH and RU, but no RH (TH chained message)

- TH, RH, and RU.

The FID5 TH, RH, and RU are often referred to as the DATA portion of an NLP. The NHDR and THDR are required in an NLP, but all of the parts in the DATA portion are optional. The valid combinations of an NLP are:

- NHDR and THDR, but no DATA (referred to as an HPR control message)  
See “HPR Control Messages” on page 151 for more information about HPR control messages.
- NHDR, THDR, TH, RH, and RU
- NHDR, THDR, TH, and RH, but no RU
- NHDR, THDR, TH, and RU, but no RH (TH chained message)  
See “Reassembling Input Messages” on page 176 for more information about TH chained messages.
- NHDR, THDR, and RU, but no TH or RH (THDR chained message).  
See “Segmentation and Reassembly” on page 174 for more information about THDR chained messages.

See *IBM Systems Network Architecture Network Product Formats* for more information about PIUs and NLPs.

## NHDRs

The network layer header (NHDR) portion of the NLP is used by the network to route the NLP from the origin RTP endpoint to the destination RTP endpoint along a specific path. When the origin RTP endpoint builds the NLP, the list of ANR labels representing the path from the origin node to the destination node is placed in the ANRF field of the NHDR. When an intermediate node receives an NLP, the ANRF field in the NHDR is examined to find the ANR label of the next hop, that ANR label is removed from the ANRF field, and the NLP is then routed to the next node. The NHDR is the only part of an NLP that is examined by or modified by intermediate nodes in the network. When the NLP arrives at the remote RTP endpoint, the only part of the ANRF field that remains is the NCE representing the destination LU.

The following figure shows an example of how NLPs are routed:

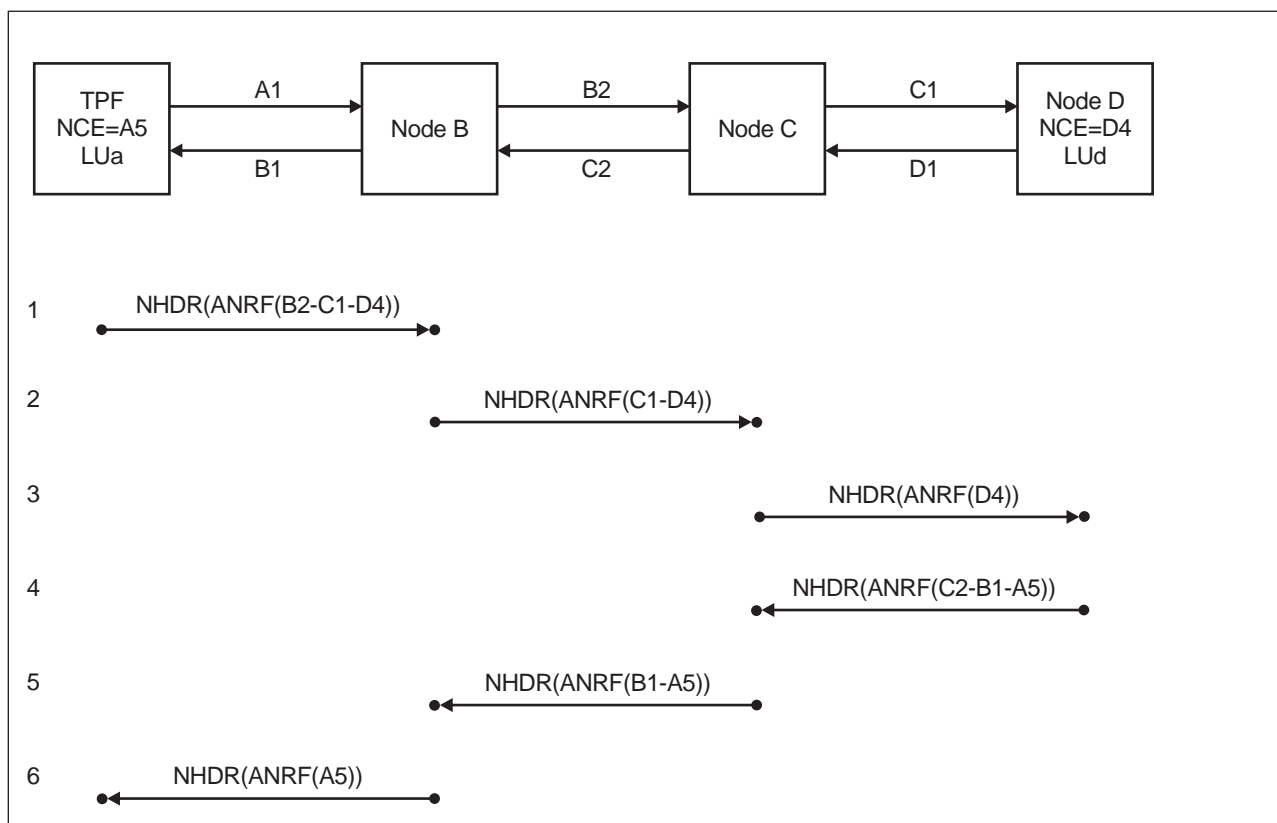


Figure 74. Routing an NLP through the Network

In Figure 74:

- An RTP connection exists between the TPF system and node D.
- A session between LUa and LUd exists on that RTP connection.
- The current path (ANR labels) for the RTP connection from the TPF system to node D is A1-B2-C1.
- The current path for the RTP connection from node D to the TPF system is D1-C2-B1.
- The NCE of the TPF system is A5.
- The NCE of node D is D4.

The following are the steps for routing NLPs between the TPF system to node D:

1. LUa has data to send to LUd. An NLP is built by the TPF system. The first label of the current path (A1) indicates that the NLP needs to be routed to node B. The remainder of the current path (B2-C1) is placed in the ANRF field of the NHDR followed by NCE of the remote RTP endpoint (D4). The NLP is sent to node B.
2. When node B receives the NLP, it removes the first ANR label (B2) from the ANRF field of the NHDR and uses that label to determine where to send the NLP next. Node B then sends the NLP to node C.
3. When node C receives the NLP, it removes the first ANR label (C1) from the ANRF field of the NHDR and uses that label to determine where to send the NLP next. Node C then sends the NLP to node D.

When Node D receives the NLP, the entire NLP is processed. The NLP was routed correctly because the ANRF field contains the NCE of node D.



4. LUd has data to send to LUa. An NLP is built by node D. The first label of the current path (D1) indicates that the NLP needs to be routed to node C. The remainder of the current path (C2-B1) is placed in the ANRF field of the NHDR followed by the NCE of the remote RTP endpoint (A5). The NLP is sent to node C.
5. When node C receives the NLP, it removes the first ANR label (C2) from the ANRF field of the NHDR and uses that label to determine where to send the NLP next. Node C then sends the NLP to node B.
6. When node B receives the NLP, it removes the first ANR label (B1) from the ANRF field of the NHDR and uses that label to determine where to send the NLP next. Node B then sends the NLP to the TPF system.

When the TPF system receives the NLP, the entire NLP is processed. The NLP did get routed correctly because the ANRF field contains the NCE of the TPF system.

The NHDR of each NLP includes the specific path that the NLP will take in the network. If the current route fails and a path switch is done, a different set of ANR labels is placed in the ANRF field of subsequent NLPs to tell the intermediate nodes the new path to use. Because intermediate nodes only examine the NHDR, they have no knowledge of the RTP connection or LU-LU session associated with the NLP.

Because NLPs and FID2 PIUs can flow on the same link, the first 3 bits after the link header (LH) are used to indicate the type of message. In a FID2 PIU, these are the first 3 bits of the TH and are always set to X'001'. In an NLP, these are the first 3 bits of the NHDR and are always set to X'110'.

There are two slowdown indicators in the NHDR. When an NLP is first built by the origin RTP endpoint, both of these slowdown indicators are cleared. If any intermediate node along the route is congested, it can set one of the slowdown indicators in the NHDR. When the NLP arrives at the destination RTP endpoint, the slowdown indicators in the NHDR are examined and used to adjust the rate at which data is sent on the RTP connection. See “ARB Pacing” on page 168 for information about how the slowdown indicators in the NHDR are used.

## THDRs

The transport header (THDR) portion of the NLP contains control information about the RTP connection. The THDR is created by the RTP endpoint that sends the NLP and is processed by the remote RTP endpoint that receives the NLP. Intermediate nodes do not examine or modify the THDR.

The first part of the THDR consists of fixed-length fields that are always present. The key fields include:

- Transport connection identifier (TCID), which identifies the RTP connection over which this NLP flowed.
- Start-of-message (SOM) and end-of-message (EOM) indicators, which specify whether the NLP is a complete message or part of a THDR chained message. See “Segmentation and Reassembly” on page 174 for more information about THDR chained messages and the use of the SOM and EOM indicators.
- Status Request (SR) indicator, which, when set, indicates that the receiver of the NLP must respond by sending an NLP with a Status segment.
- Data offset, which is the offset to the data portion of the NLP (TH, RH, RU) relative to the start of the THDR.

- Data length, which is the length of the data portion of the NLP (TH, RH, and RU). If the data length is 0, the NLP is an HPR control message.
- Byte sequence number (BSN), which is the sequence number of the NLP sent on this RTP connection. The BSN is used to detect lost and duplicate data in the network.

Following the fixed part of the THDR, there are zero or more optional segments, each of which contains control information about the RTP connection. See “THDR Optional Segments” for more information about THDR optional segments.

When the TPF system receives an NLP, the NHDR and the THDR are processed immediately during read interrupt processing. SNA Opzero processes the TH, RH, and RU sections of a PIU and an NLP.

When the TPF system needs to send control information (one or more optional segments) to the remote RTP endpoint, the optional segments and user data are sent in the same NLP whenever possible. In the context of this discussion, user data means either application data or an SNA command (for example, a BIND). Piggybacking of control data and user data in the same NLP is done to reduce the number of NLPs that flow on an RTP connection. If one or more optional segments need to be sent when no user data is waiting to be sent, and no user data is generated in the next SNA polling interval, the TPF system will send an HPR control message (which is an NLP that contains just an NHDR and THDR, but no data).

### THDR Optional Segments

The following list contains the optional segments that can be included in the THDR:

Adaptive Rate-Based	ARB (X'22')	Used to control the flow of data on the RTP connection.
Connection Fault	CF (X'12')	Sent whenever a protocol violation has been detected, or to immediately deactivate an RTP connection.
Connection Identifier Exchange	CIE (X'10')	When node X starts an RTP connection with node Y, the first NLP sent by node Y back to node X includes the CIE segment, which contains the TCID that node Y assigned to the RTP connection that was just started.
Client “Out of Band” Bits	COB (X'10')	Used to deactivate the RTP connection normally.
Status	STATUS (X'0E')	Acknowledges the receipt of data and informs the RTP endpoint of missing data that needs to be retransmitted.
Connection Setup	CS (X'0D')	Sent when a new RTP connection is being started. It identifies the control point (CP) name of the destination RTP endpoint and the network connection endpoint (NCE) identifier in the destination RTP endpoint.
Switching Information	SI (X'14')	Sent by node X to node Y to inform node Y of the path it should now use for the RTP connection. The SI segment is included in the NLP that starts the RTP connection and anytime a path switch has occurred.

## SYNC and ECHO Numbers

RTP endpoints maintain SYNC numbers for each RTP connection. A SYNC number is sent as part of the STATUS segment when present in the THDR of an NLP. The SYNC number is used to detect old control information. Each time a state change takes place on the RTP connection, the SYNC number is incremented. Each RTP endpoint has its own SYNC number. In addition, each RTP endpoint keeps track of the most current SYNC number received from the remote RTP endpoint. When building a STATUS segment, an RTP endpoint places its own SYNC number in the SYNC field of the STATUS segment and then places the current SYNC number received from the remote RTP endpoint in the ECHO field of the STATUS segment.

The SYNC and ECHO numbers are used to detect a path switch race condition, which means that both RTP endpoints do a path switch at the same time. When an RTP endpoint does a path switch, it increments its SYNC number and sends an NLP containing the STATUS segment and the SI segment, which contains the new route. When a STATUS segment is received whose ECHO field contains the value of the new SYNC number of this RTP endpoint, the remote RTP endpoint has acknowledged that a path switch has taken place. A path switch race condition exists if the NLP with the SI segment is sent and then an NLP with an SI segment is received before receiving an NLP with a STATUS segment that acknowledges the path switch.

SYNC and ECHO numbers also play an important role in the RTP connection resynchronization process. See “RTP Connection Resynchronization Process” on page 163 for more information about the RTP connection resynchronization process.

## Data

The data portion of an NLP can contain a transmission header (TH), request header (RH), and request unit (RU), all of which are optional. The data portion of an NLP exists when there is either application data or an SNA command to send on an LU-LU session. The RH and RU sections of an NLP are identical to the RH and RU sections in an FID2 PIU.

The TH section in the data portion of an NLP is an FID5 TH. It is a different format than an FID2 or FID4 TH, but provides the same function in that it identifies the LU-LU session. The session address (SA) field is contained in the FID5 TH. The combination of the SA from the TH and the TCID from the THDR uniquely identifies the LU-LU session. As with the FID2 and FID4 TH, the FID5 TH includes the sequence number field (SNF), which identifies the sequence number of the message for this LU-LU session.

## HPR Control Messages

An HPR control message is an NLP that includes an NHDR and THDR, but no data (no TH, RH, or RU). Control messages are sent when one RTP endpoint has control information (THDR optional segments) to send to the partner RTP endpoint regarding a specific RTP connection, but has no user data to send for any of the LU-LU sessions on the RTP connection.

Before HPR support, every PIU that flowed in or out of the TPF system contained data for an LU-LU session, except for virtual route (VR) pacing responses. With HPR support, there are two more conditions where there is no LU-LU session data:

- FID2 ROUTE\_SETUP command, which is a PIU used to obtain the HPR path information for an RTP connection and does not flow on any LU-LU session

- HPR control message, which contains control information about the RTP connection sent by one RTP endpoint to the remote RTP endpoint.

## Network Considerations for NLPs

Because an NLP is larger than a PIU, messages that used to fit in a single read buffer before HPR support was installed may now span more than one read buffer. To account for this condition, consider increasing the size of the read buffers used to read in data from the network.

Use the UNITSZ parameter on the SNAKEY macro in CTK2 to change the size of the read buffers. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro.

---

## HPR Control Blocks

The following control blocks were created for HPR support:

- The rapid transport protocol control block (RTPCB) table, which contains information about RTP connections.
- The high-performance routing session address table (HPRSAT), which contains the session addresses of HPR LU-LU sessions.
- The high-performance routing message table (HPRMT), which contains output messages sent on RTP connections that are waiting for an acknowledgement from remote RTP endpoints.

## RTPCB Table

The rapid transport protocol control block (RTPCB) table contains hash buckets, the RTPCB header, and one entry for every RTP connection. Each RTPCB entry is made up of three parts. The following figure shows an example of the RTPCB table layout:

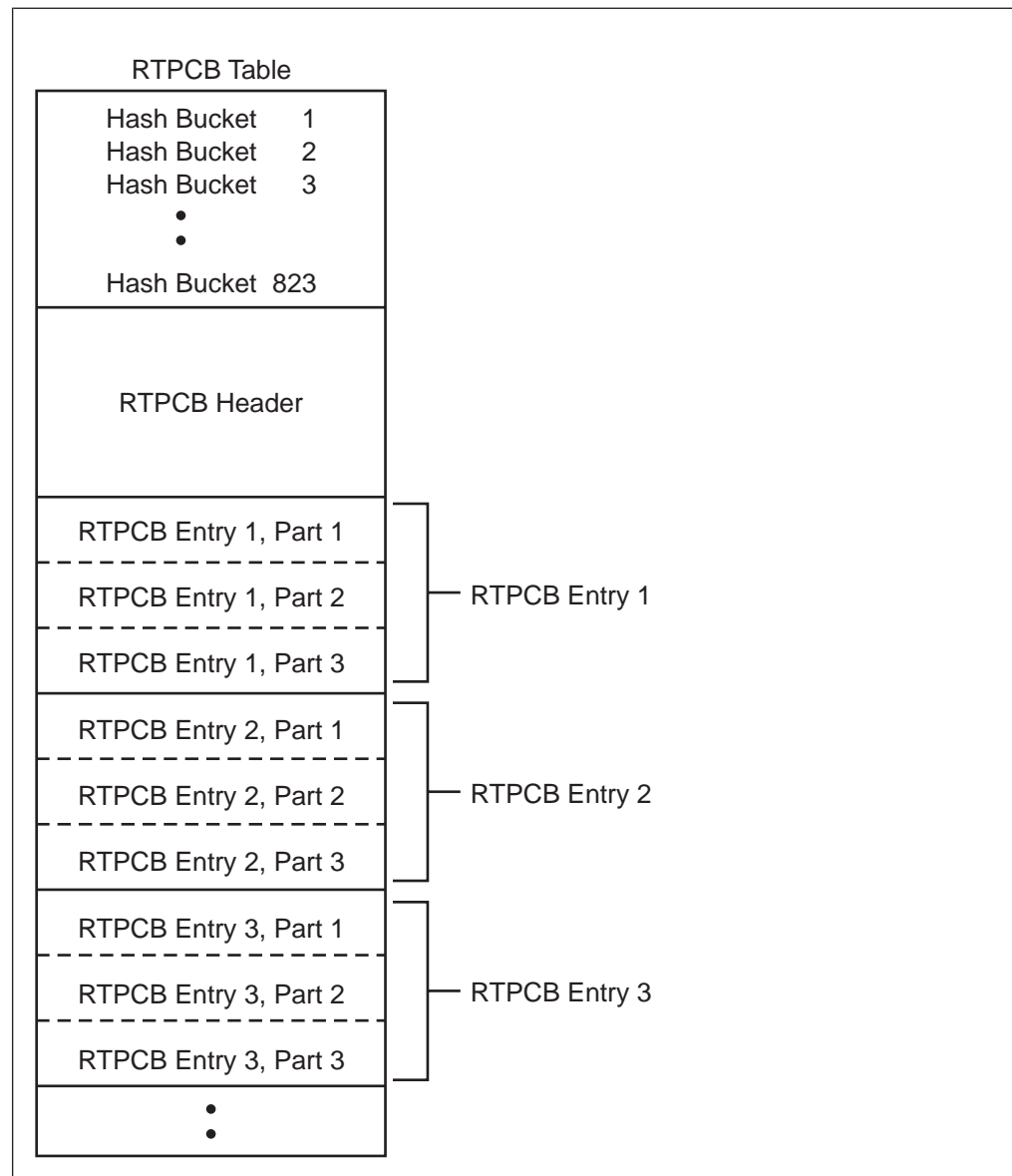


Figure 75. RTPCB Table Layout

In Figure 75 an RTPCB entry is referenced using its RTPCB index, which is a direct index into the RTPCB table. For example, the RTPCB index of the first RTPCB entry is 1. Because RTP connections do not have names like other SNA resources (like LUs, CPs, ALSSs, and so on), they are identified by their RTPCB index. Therefore, the RTPCB index is used as input to some of the SNA commands, including the ZNRTP commands, and shows up in the PIU trace table.

When an RTP connection is started, an RTPCB entry is assigned and added to the appropriate hash bucket based on the control point name of the remote RTP endpoint. When the RTP connection ends, the RTPCB entry is removed from whatever hash bucket it is in and the RTPCB entry is returned to the system.

For performance reasons, all RTPCB entries that represent RTP connections going to the same remote RTP endpoint are chained together using the hash bucket mechanism. When an HPR LU-LU session is starting, the TPF system needs to know if there are existing RTP connections that can be used. Rather than

sequentially searching every RTPCB entry for a matching route, the hash bucket algorithm allows for a quick and effective search of the RTPCB table.

Each RTPCB entry is broken into three parts for the following reasons related to keypointing:

- Part 1 of the RTPCB entry contains static information about the RTP connection; for example the TCIDs, COS name, CP name of the remote RTP endpoint, and the path to that remote node (both RSCV and ANR labels). Because part 1 is large and the information does not change very often, part 1 of an RTPCB entry is written to DASD (keypointed) immediately whenever the RTP connection starts, ends, or completes a path switch. The #RT1RI fixed file records contain part 1 of the RTPCB entries. See “Defining Fixed File Records for the RTPCB Table” on page 155 for more information about defining #RT1RI records.
- Part 2 of the RTPCB entry contains information necessary to resynchronize the RTP connection if a hardware IPL of the TPF system is performed. Whenever information in part 2 of an RTPCB entry is changed, that entry is marked for keypointing. When the time-initiated SNA keypointing task gets control, it files part 2 of all RTPCB entries that were marked for keypointing during the previous SNA keypointing interval. The RTP connection resynchronization process needs fairly recent information about the RTP connection, which is why this information is written to DASD periodically. The #RT2RI fixed file records contain part 2 of the RTPCB entries. See “Defining Fixed File Records for the RTPCB Table” on page 155 for more information about defining #RT2RI records. See “RTP Connection Resynchronization Process” on page 163 for more information about the RTP connection resynchronization process.
- Part 3 of the RTPCB entry contains information about the RTP entry that does not need to be keypointed. This information is either initialized or rebuilt following an IPL of the TPF system.

The RTPCB header contains statistical information about HPR support. It also contains the anchor for the various chains involving RTPCB entries. For example, all RTPCB entries for RTP connections that are doing a path switch are chained together. In another example, all RTPCB entries that have their short request timer running are chained together. This allows the TPF system to perform the necessary RTP endpoint functions effectively without having to scan the entire RTPCB table.

### Defining the RTPCB Table

After you have determined the maximum number of HPR LU-LU sessions, the next step is to decide how to spread those sessions across RTP connections and determine how many RTP connections (RTPCB table entries) will be needed on each TPF processor. This becomes the value of the MAXRTPCB parameter on the SNAKEY macro in CTK2. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro.

You need at least one RTP connection for each remote RTP endpoint that has HPR LU-LU sessions with a TPF processor. If there are many LU-LU sessions between a TPF processor and a given remote RTP endpoint across different routes, or many sessions using different class of service (COS) values, there will be multiple RTP connections between the TPF processor and remote RTP endpoint.

The URTP user exit allows you to control how many LU-LU sessions are assigned to an RTP connection. See *TPF System Installation Support Reference* for more information about the URTP user exit.

## Defining Fixed File Records for the RTPCB Table

Parts 1 and 2 of the RTPCB table are keypointed. Based on the number of RTPCB entries defined, you need to define the necessary number of #RT1RI and #RT2RI fixed file records. These are processor unique records. To determine how many records you need to define for a TPF processor, use the following formulas:

- Number of #RT1RI records = MAXRTPCB / 5
- Number of #RT2RI records = MAXRTPCB / 127

## Displaying RTPCB Entries

The ZNRTP DISPLAY command displays the contents of an RTPCB entry. The display can be either raw data or formatted. You can also use the ZDDCA command with the RTP dump tag specified to display the RTPCB table and its entries. See *TPF Operations* for more information about the ZNRTP DISPLAY and ZDDCA commands.

## Initializing RTPCB Entries

You can use the ZNRTP INITIALIZE command to initialize RTPCB entries. However, do not initialize RTPCB entries in a production environment. Doing so resets SNA control blocks in the TPF system without informing the network. Inconsistencies will occur. Therefore, use the ZNRTP INACT command, rather than the ZNRTP INITIALIZE command, whenever possible.

See *TPF Operations* for more information about the ZNRTP INITIALIZE and ZNRTP INACT commands.

## HPRSAT

The high-performance routing session address table (HPRSAT) contains one entry for every active HPR LU-LU session. Each entry contains the following information:

- The session address (SA) that the TPF system assigned
- The SA that the remote RTP endpoint assigned
- The RTPCB index of the RTP connection
- The resource identifier (RID) of the LU RVT entry.

The HPRSAT is primarily used by the PIU trace code at read interrupt completion when NLPs are received from the network. Its purpose is to provide an effective way to map the SA received in the NLP to its corresponding LU RVT entry. The HPRSAT does for HPR sessions what the network address table (NAT) does for PU 5 LU-LU sessions, and what the session identifier table (SIT) does for PU 2.1 LU-LU sessions. All of these tables enable the TPF system to quickly find the LU RVT entry for a given PIU.

The HPRSAT is not keypointed. Instead, the table is always rebuilt after an IPL of the TPF system from information in the LU RVT entries.

## Defining the HPRSAT

The HPRSAT is a single table shared by all RTP connections. The MAXHPRSA parameter on the SNAKEY macro in CTK2 defines how many entries there are in the HPRSAT, which is the maximum number of HPR LU-LU sessions that can be active at any time. An HPRSAT entry is assigned when an HPR LU-LU session starts, and the entry is cleared when that LU-LU session ends. The HPRSAT entry assigned to an LU-LU session is based on a hashing algorithm for performance reasons.



When defining the HPRSAT, you need to consider the existing APPN or PU 5 LU-LU sessions that will be migrated to HPR support as well as new LUs that will be added to the network.

See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro.

### Displaying the HPRSAT

Use the ZDDCA command with the HSA dump tag specified to display the HPRSAT. See *TPF Operations* for more information about the ZDDCA command.

## HPRMT

The high-performance routing message table (HPRMT) is used to save the data portion of HPR output messages (NLPs) until they are acknowledged by the remote RTP endpoint. A message is added to the HPRMT just before the NLP is placed on the SOUTC queue to be transmitted. A message is removed from the HPRMT when an NLP is received containing a STATUS segment that acknowledges receipt of the message.

If one or more NLPs sent by the TPF system become lost (for example, because of a failure in the network), the remote RTP endpoint will detect that data was lost and request that the TPF system retransmit the missing data. For this condition, the TPF system will find the necessary data in the HPRMT and build another NLP to resend that data to the remote RTP endpoint. All of the data and other important information about the message is saved in the HPRMT so that the message can be retransmitted without accessing the LU RVT entry because the LU-LU session may no longer be active. For example, the TPF system sends a data message followed by an UNBIND, causing the LU RVT entry to be cleaned up. The data message and UNBIND NLPs are both lost and must be retransmitted. The data message and UNBIND NLPs are rebuilt using only the information that was saved in the HPRMT.

### Defining the HPRMT

The HPRMT resides only in core memory. You can define the size of the HPRMT by using the HPRMTSIZ parameter on the SNAKEY macro in CTK2. You must decide if the TPF system will save HPR output messages until they are acknowledged by the remote RTP endpoint. Saving messages requires additional storage. However, not saving messages can cause RTP connections (and the sessions through them) to be taken down when there is a failure in the network. For example, if data becomes lost in the network and the TPF system is requested to retransmit that data, the RTP connection will be broken because there is no saved copy of the data to retransmit. If you do not want the TPF system to save HPR output messages, set the value of the HPRMTSIZ parameter on the SNAKEY macro in CTK2 to 0, which means that the HPRMT is not defined. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro.

If you want the TPF system to save HPR output messages, you need to define the size of the HPRMT. Some key factors to consider are:

- Output message rate
- Average output message size
- Turnaround time, which is how long it takes the remote RTP endpoint to acknowledge data.

If the HPRMT is defined but becomes full, HPR output messages cannot be saved until space becomes available in the table. Again, if the TPF system is requested to retransmit a message that cannot be found in the HPRMT, the RTP connection will



be broken. The HPRMT does not and cannot use an aging algorithm. For example, if the HPRMT is full and a new HPR message is being sent out, throwing away the oldest message in the HPRMT to make room to save the new message will not work because messages that have been in the HPRMT the longest time are the ones that most likely will need to be retransmitted.

Because of the volume of data, it is not practical to keypoint the HPRMT. Across a software IPL of the TPF system, the HPRMT remains as is. Across a hardware IPL of the TPF system, the HPRMT is initialized and the RTP connection resynchronization process is started. See “Host IPL Considerations” on page 163 for more information about how HPR support handles IPLs of the TPF system.

On a given TPF system, the average output message size and network turnaround time should be relatively constant regardless of the message rate. For example, when your TPF system is in a steady state and sending 500 HPR output messages per second, assume that the percentage of the HPRMT that is in use stays near 40%. Because the message rate and amount of HPRMT in use are directly proportional, you can predict that at 1000 messages per second, about 80% of the HPRMT would be in use.

### **Displaying Information about the HPRMT**

Use the ZNRTP SUMMARY command to display information about how much of the HPRMT is currently in use and the maximum amount of the HPRMT that was in use at any point in time. You can also use the ZDDCA command with the HMT dump tag specified to display the HPRMT table.

The TPF system also automatically sends warning messages to the TPF operator console when the HPRMT becomes at least 90% full or 100% full.

## **Relationship with Other SNA Control Blocks**

For each LU-LU session, one of the LU RVT entries contains the information about the session. This is called the session RVT entry. For LU 6.2 sessions, the session RVT entry is actually a session control block (SCB) entry.

### **Non-HPR LU-LU Sessions**

When a PU 5 or PU 2.1 LU-LU session is started, the session is assigned to a specific link (ALS, CTC, or NCP) and remains assigned to that link for the duration of the session. The session RVT entry points to the CCW area of the assigned link using the CCW index (RV1CCWIN) field. Whenever data is to be sent on the LU-LU session, the CCW index in the session RVT is used to determine the link over which the data will be sent. If a link fails in the network, all LU RVT entries whose CCW index matches that of the failing link are cleaned up.

The following figure shows an example of the RVT and CCW area control blocks for five LU-LU sessions across two links. These could be either PU 5 or PU 2.1 LU-LU sessions:

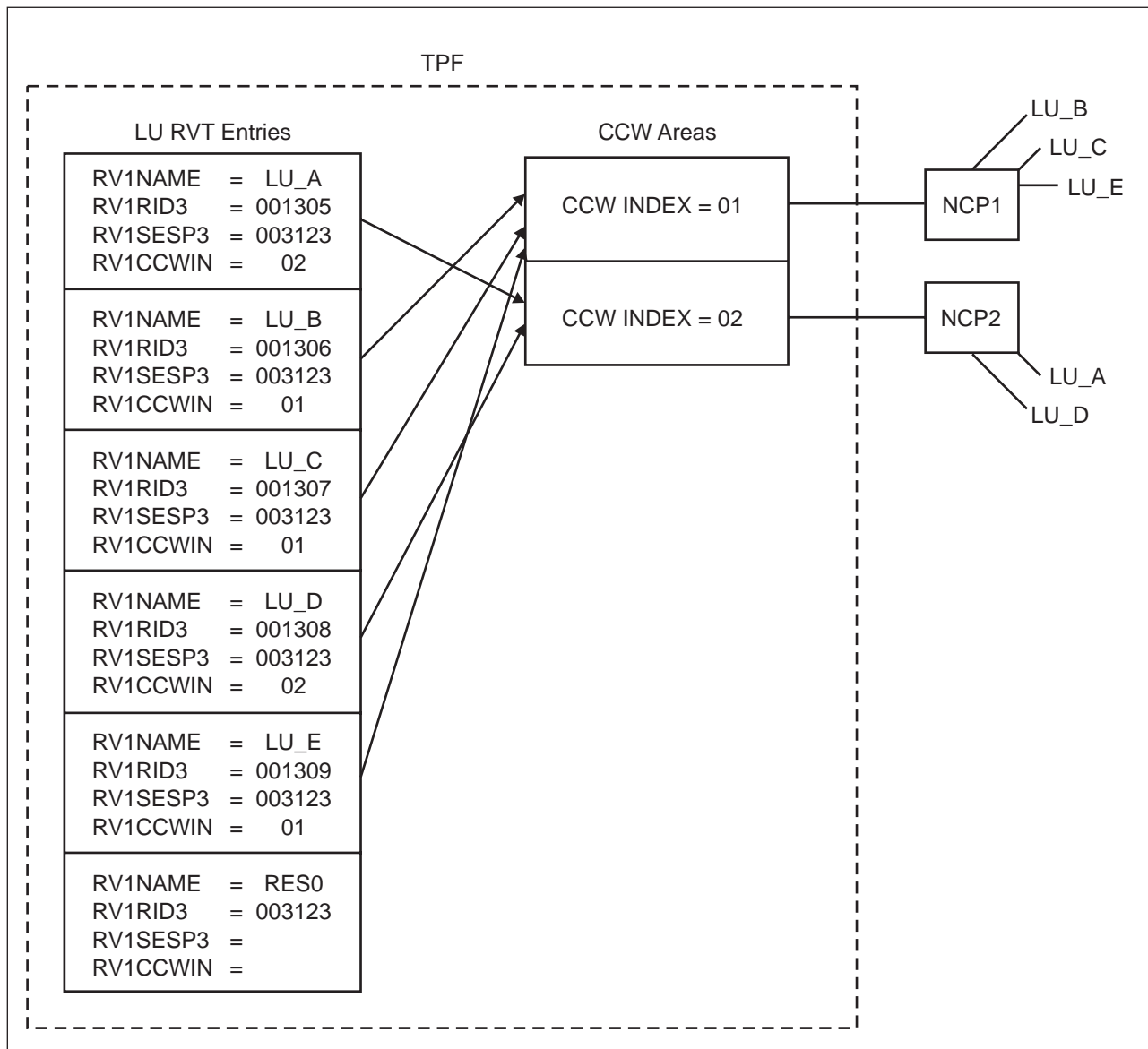


Figure 76. LU RVT Entries for PU 5 and PU 2.1

In Figure 76:

- The five remote LUs (LU\_A, LU\_B, LU\_C, LU\_D, and LU\_E) are all in session with TPF application RES0.
- For all five sessions, the remote LU RVT entry is the session RVT entry. The session partner RID (RV1SESP3) field in each of these RVT entries contains the RID of the RES0 RVT entry.
- The sessions with LU\_B, LU\_C, and LU\_E go through link NCP1, whose CCW index is 1.
- The sessions with LU\_A and LU\_D go through link NCP2, whose CCW index is 2.

If a message needs to be sent to LU\_B, the CCW index in its RVT entry (RV1CCWIN=01) indicates that NCP1 is to be used.

If the link to NCP2 fails, the RVT entries for LU\_A and LU\_D would be cleaned up because their CCW index (RV1CCWIN=2) matches the CCW index of the NCP2 link.

### **HPR LU-LU Sessions**

For HPR LU-LU sessions, the session RVT entry does not point to its link (the CCW index field in the RVT entry is not used for HPR LU-LU sessions). Instead, the session RVT entry points to the RTPCB entry of the RTP connection over which the LU-LU session exists. In turn, the RTPCB entry points to the current link assigned to the RTP connection using the CCW index field. Because the link assigned to an RTP connection can change (when a path switch takes place), the CCW index in the RTPCB entry will change. While the path switch process is in progress, there is no link assigned to the RTP connection.

When a non-HPR LU-LU session is started, the session is assigned to a specific link and remains assigned to that link for the duration of the session. When an HPR LU-LU session is started, the session is assigned to a specific RTP connection (not a specific link) and remains assigned to that RTP connection for the duration of the session. The link assigned to an RTP connection does not necessarily remain the same for the duration of the RTP connection; it can change because of a path switch.

The following figure shows an example of the RVT, RTPCB table, and CCW area control blocks for five HPR LU-LU sessions:

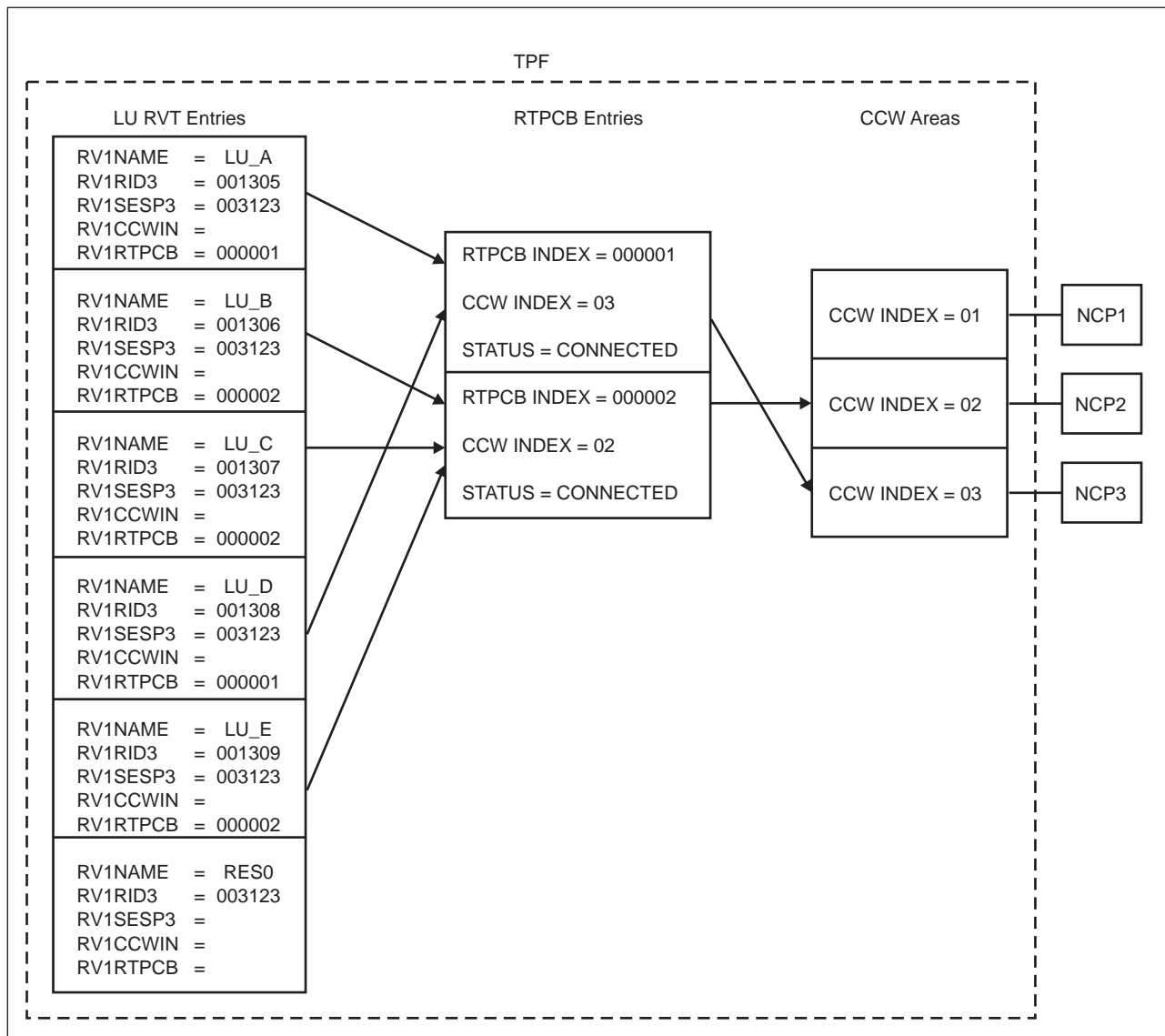


Figure 77. HPR LU RVT Entries, RTP Connections Are Active

In Figure 77:

- The five remote LUs (LU\_A, LU\_B, LU\_C, LU\_D, and LU\_E) are all in session with TPF application RES0.
- For all five sessions, the remote LU RVT entry is the session RVT entry. The session partner RID (RV1SESP3) field in each of these RVT entries contains the RID of the RES0 RVT entry.
- The sessions with LU\_A and LU\_D use RTP connection 1. Therefore, the RTPCB index field (RV1RTPCB) in these RVT entries is 1.
- The sessions with LU\_B, LU\_C, and LU\_E use RTP connection 2. Therefore, the RTPCB index field (RV1RTPCB) in these RVT entries is 2.
- RTP connection 1 is in CONNECTED state. The current route for this RTP connection goes through NCP3 (whose CCW index is 3).
- RTP connection 2 is in CONNECTED state. The current route for this RTP connection goes through NCP2 (whose CCW index is 2).

If a message needs to be sent to LU\_A, the CCW index field in this RVT entry is not examined because it is an HPR LU-LU session. Instead, the RTPCB index is used to get to the RTPCB entry (which is for RTP connection 1). The RTP connection is in CONNECTED state; therefore, the message can be sent out. The CCW index in the RTPCB entry indicates to send the message through NCP3.

Next, assume that the link to NCP3 fails. Because the current route for RTP connection 1 was through NCP3, this RTP connection starts the path switch process. The following figure shows an example of how the control blocks look at this point:

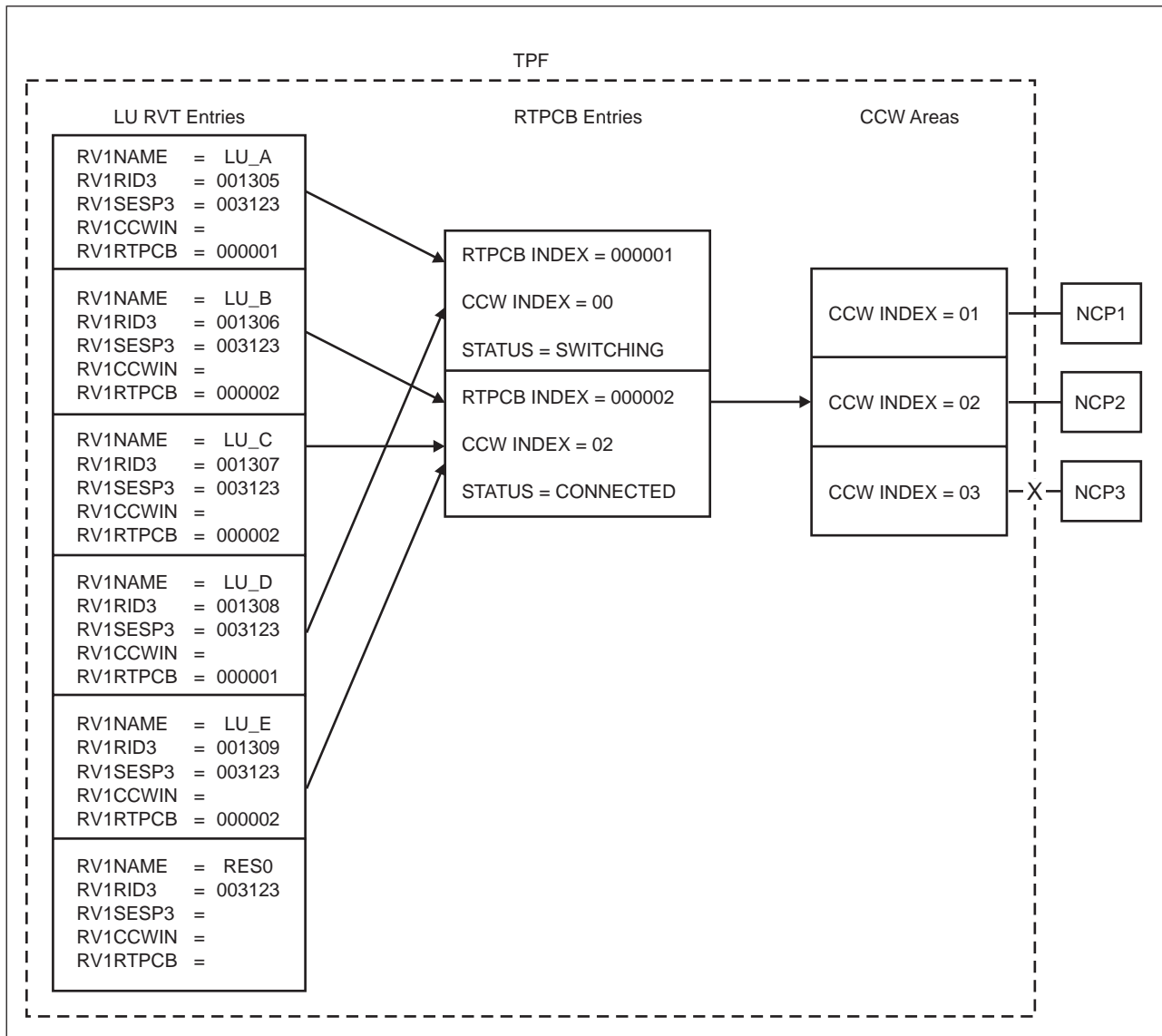


Figure 78. HPR LU RVT Entries, Path Switch Is in Progress

In Figure 78, RTP connection 1 is now in SWITCHING state and its CCW index is 0 because no route currently exists for this RTP connection. The HPR LU RVT entries are not changed at all when NCP3 fails.

While the path switch is in progress, the TPF application wants to send another message to LU\_A. Because the RTPCB entry associated with the LU\_A session is

in SWITCHING state, the message cannot be sent at this time and is placed on the RTP output queue. See “RTP Output Queue” on page 170 for more information about the RTP output queue.

The path switch process for RTP connection 1 is completed successfully and the new route goes through NCP1. The following figure shows an example of how the control blocks look at this point:

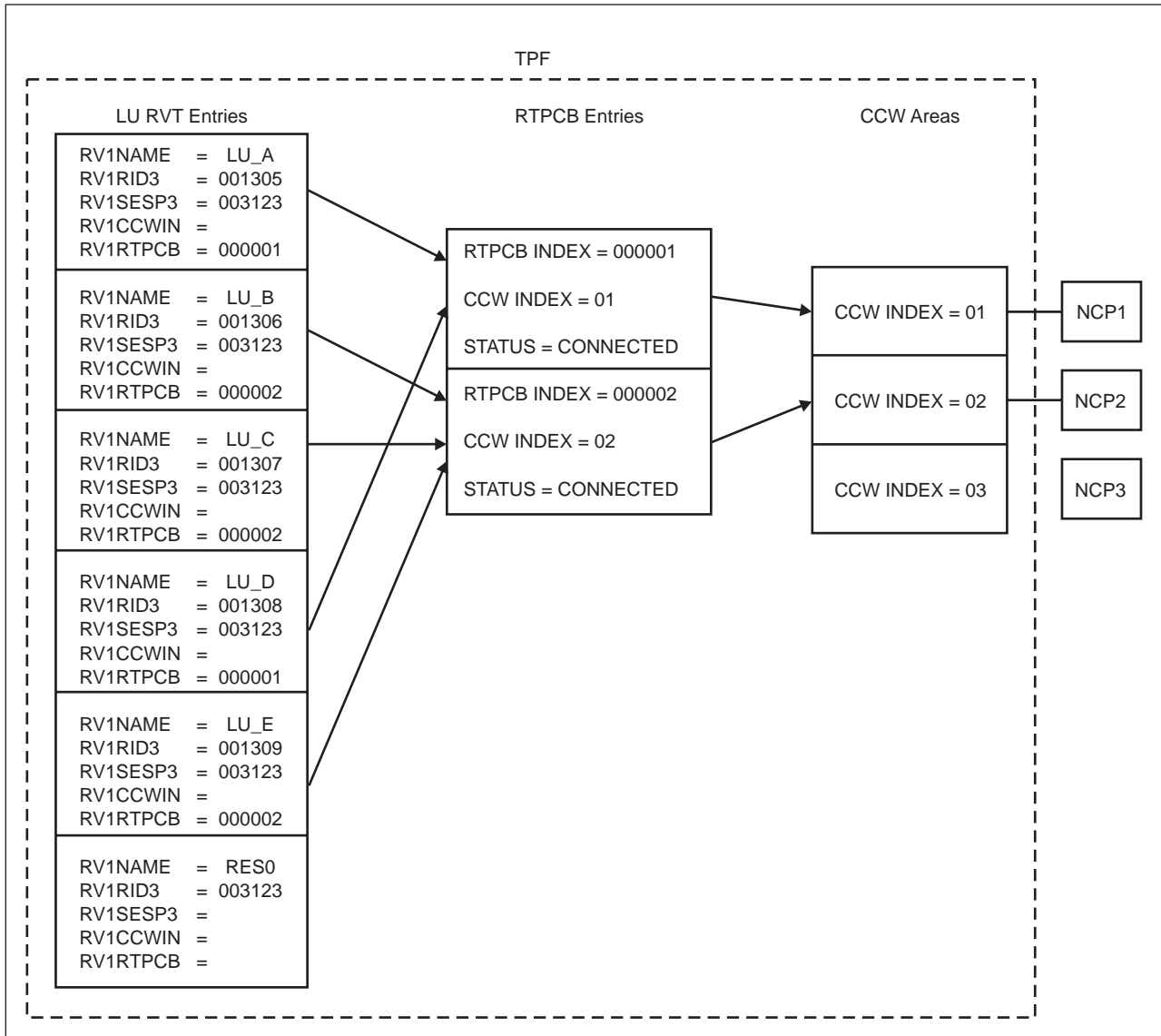


Figure 79. HPR LU RVT Entries, Path Switch Is Completed

In Figure 79, the RTPCB entry for RTP connection 1 is back in CONNECTED state and its CCW index has been updated to point to the link to NCP1. Any messages that were queued during the path switch are now sent across the new route. Throughout the entire path switch process, the HPR LU RVT entries are not changed; all of the changes pertaining to the route are made to the RTPCB entries instead.

---

## Host IPL Considerations

After the TPF system performs an IPL, it will take one of the following actions for the RTP connections:

- Clean up all RTP connections
- Perform the RTP connection resynchronization process for each RTP connection
- Do nothing.

When the TPF system takes an outage, whether the network remains active or not depends on how long it takes the TPF system to IPL. If the TPF system is down for too long and the automatic network shutdown (ANS) timers expire, the network will deactivate the links to the TPF system causing all PU 5 and PU 2.1 LU-LU sessions across those links to fail. If the links remain active across the IPL, the LU-LU sessions remain active as well.

Because there is no way of knowing how long the TPF system has been down for a hardware IPL, and because you can define the ANS timer values for each link, the TPF system verifies the status of each link after the IPL. For HPR LU-LU sessions, a link failure does not cause those sessions to fail. Instead, it triggers a path switch of the RTP connections that were using that link. If the TPF system is down for a long period of time, the network will clean up all RTP connections that went to the TPF system and, therefore, all HPR LU-LU sessions as well. When the TPF system comes back up, it needs to be able to detect this condition and internally clean up all RTP connections and HPR LU-LU sessions that were active before the outage.

If the network has cleaned up its RTP connections, the TPF system could detect this after the IPL. No responses would be received on those RTP connections causing path switches to be started. When the path switch attempts eventually time out, this would cause the TPF system to clean up the RTP connections. Rather than wait for all of this to happen, the TPF system checks to see if any HPR-capable links remained active across the IPL. If not, this indicates that the TPF system has been down too long and causes the TPF system to clean up all RTP connections and HPR LU-LU sessions.

If at least one HPR-capable link remained active across the IPL of the TPF system and the SNA tables were reloaded from file (which always occurs for a hardware IPL), information in the RTPCB entries will likely be old. If RTPRSYNC=YES is coded on the SNAKEY macro in CTK2, the RTP connections remain active and the TPF system performs the RTP connection resynchronization process for each RTP connection. If RTPRSYNC=NO is coded on the SNAKEY macro in CTK2, the TPF system ends all RTP connections.

If at least one HPR-capable link remained active across the IPL of the TPF system and the core copies of the SNA tables were reused, the RTP connections remain active. The RTP connection resynchronization process is not necessary when this occurs because the information in the RTPCB entries is accurate.

## RTP Connection Resynchronization Process

The RTP connection resynchronization process is the method used to keep RTP connections active across a hardware IPL of the TPF system. After a hardware IPL, the file copy of the SNA tables, including the RTPCB table, is reloaded from file. The RTPCB table on file is likely to be several seconds old. Therefore, the TPF system does not know the current input or output byte sequence number (BSN) values for an RTP connection. The current SYNC and ECHO values for an RTP connection are not known either. See “SYNC and ECHO Numbers” on page 151 for

more information about how SYNC and ECHO numbers are used by HPR support. The following provides an example of the problems:

1. An RTP connection is active. Time-initiated keypointing files out the RTPCB entry, which contains the following values:
  - SYNC sent = 103
  - SYNC received = 85
  - Next BSN to send = 200
  - Next expected BSN to receive = 500.
2. Messages are sent and received on the RTP connection. The RTPCB entry now contains the following values:
  - SYNC sent = 105
  - SYNC received = 88
  - Next BSN to send = 450
  - Next expected BSN to receive = 622.
3. A hardware IPL of the TPF system is done. SNA restart reloads the SNA tables from file. The RTPCB entry after the IPL contains:
  - SYNC sent = 103
  - SYNC received = 85
  - Next BSN to send = 200
  - Next expected BSN to receive = 500.

All of the values in the RTPCB entry are old, which can lead to different problems:

1. If the TPF system sends an NLP containing a STATUS segment, the remote RTP endpoint discards the control information in that NLP because the SYNC number (103) in the NLP is old. The current SYNC number is now 105. The ECHO number (85) in the STATUS segment is also old (it should be 88).
2. If the remote RTP endpoint sends an NLP containing a STATUS segment, the TPF system would accept the control information because the SYNC number (88) in the NLP is equal to or greater than the last SYNC number received (85). The problem is that the STATUS segment would acknowledge receiving messages up to BSN value 450, but the TPF system thinks it has not yet sent bytes 200–449. This would be treated as a protocol violation and cause the RTP connection to be taken down.
3. If the TPF system sends an NLP containing data, the remote RTP endpoint discards the data thinking that it is duplicate data because its BSN (200) is older than the next expected BSN (which is 450).
4. If the TPF system sends an NLP with BSN=200 and a length of 300 bytes, the first 250 bytes of data would be discarded (bytes 200–449) and the last 50 bytes would be treated as the next expected message. Because these 50 bytes are really the middle of a message, they would not have the correct start-of-message header settings. This will cause the remote RTP endpoint to break the RTP connection because of a protocol violation.
5. If the remote RTP endpoint sends an NLP containing data, the BSN in the NLP will be 622. Because the TPF system is expecting data starting with BSN=500, the TPF system queues the NLP and asks the remote RTP endpoint to retransmit bytes 500–621. Because the TPF system already acknowledged receipt of bytes 500–621 before the IPL, the remote RTP endpoint does not have that data anymore and will break the RTP connection.

The RTP connection resynchronization process prevents all of these problems. The first step after reloading the RTPCB table from file is to increase the SYNC number value of the RTP connection by a large amount to make sure it is current. Using the previous example, the SYNC number in the file copy of the RTPCB entry is 85, but the real current SYNC number is 88. The RTP connection resynchronization



process will increase the SYNC number in the RTP entry by a large amount (for example, by 100), so that the new value (185) is guaranteed to be greater than the current SYNC number. This way, control information sent by the TPF system will be accepted.

The next step is to set a flag in the RTPCB entry to indicate that when the first NLP is received after the IPL, assume that the BSN in that NLP is the BSN of the next expected message.

The final step is the most complicated part of the RTP connection resynchronization process. The TPF system sends out an HPR control message (an NLP with no data) to ask the remote RTP endpoint the BSN of the next message it is expecting. Until the response to that control message is received, the TPF system cannot send any data on this RTP connection. When the response is received, the next expected BSN value is copied into the output BSN field in the RTPCB entry and data traffic continues.

The following figure shows an example of the events leading up to a hardware IPL of the TPF system:

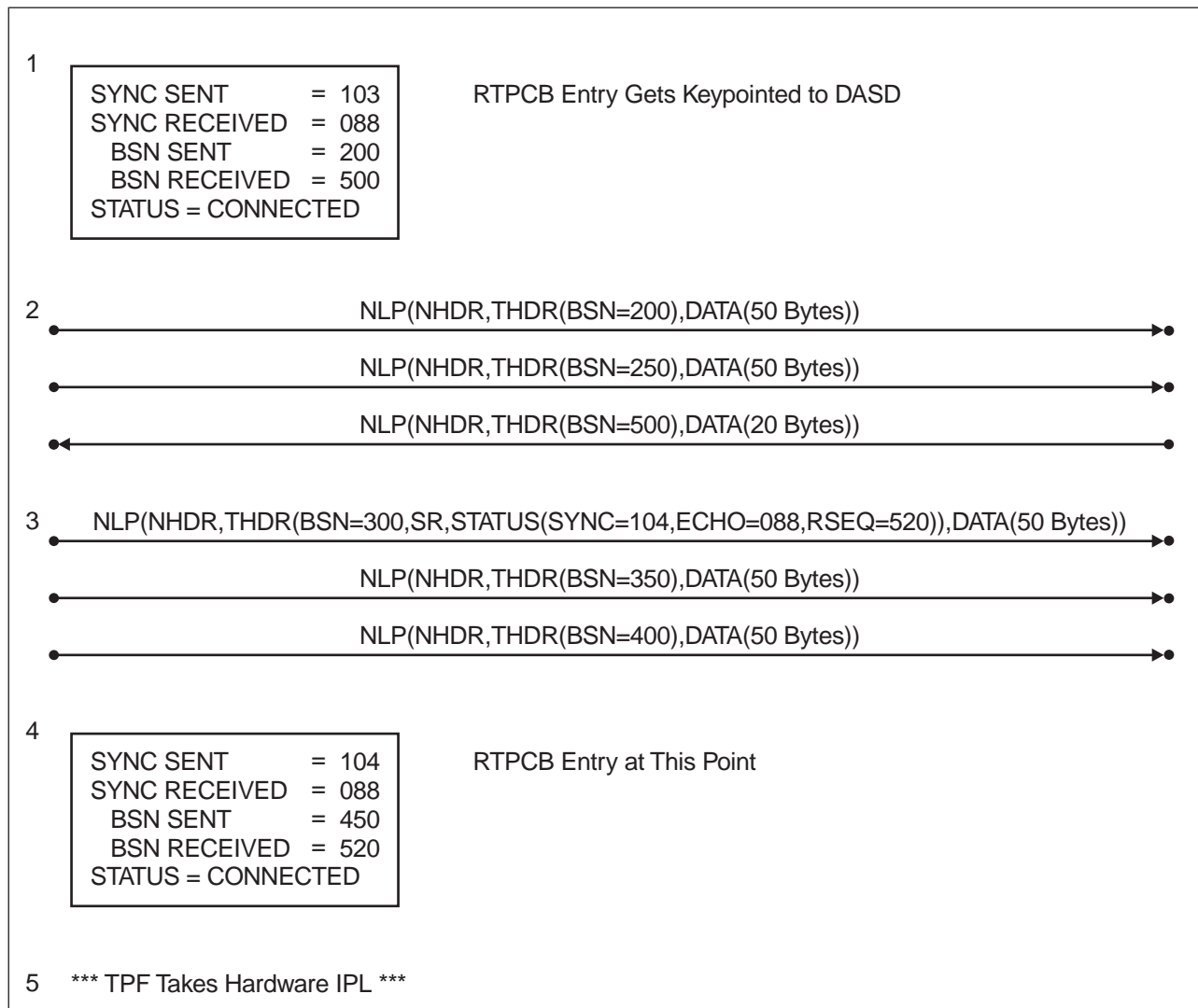


Figure 80. RTP Connection Resynchronization, before the IPL

In Figure 80 on page 165:

1. An RTP connection is active. SNA time-initiated keypointing files out the RTPCB entry with the values shown.
2. The TPF system sends two 50-byte messages and receives a 20-byte message.
3. The TPF system sends three more 50-byte messages, the first of which asks for a status reply.
4. The RTPCB entry for this RTP connection now contains the values shown.
5. A hardware IPL of the TPF system occurs.

The following figure shows an example of the steps involved in the RTP connection resynchronization process:

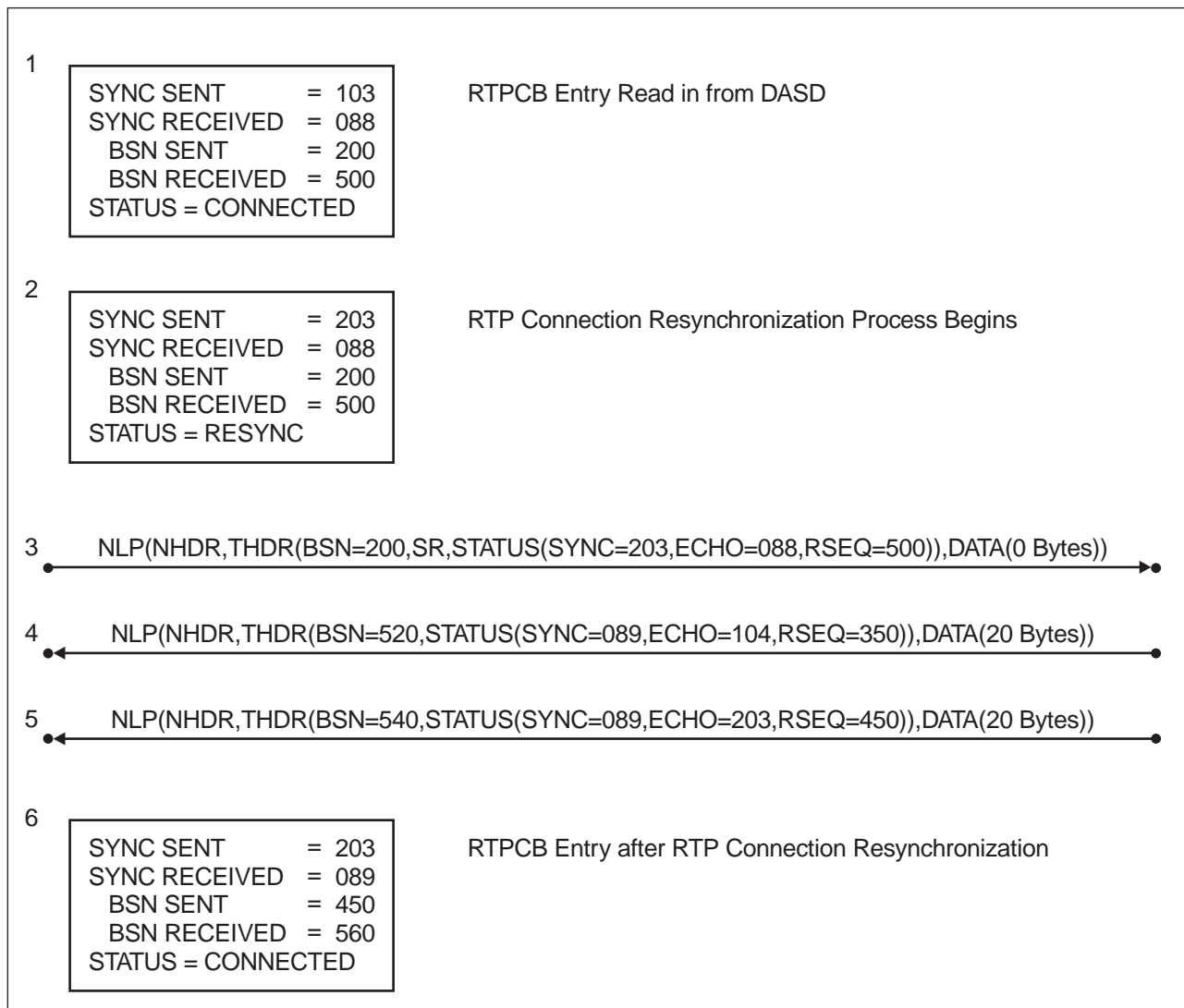


Figure 81. RTP Connection Resynchronization, after the IPL

In Figure 81:

1. The RTPCB table is reloaded from file after the IPL during SNA restart. The values in the RTPCB entry are old.

2. The RTP connection resynchronization process begins. The SYNC number is increased by a large amount (from 103 to 203) and the connection is placed in RESYNC state.
3. When the TPF system is cycled up, an NLP is sent asking for a status reply. This NLP contains no user data.
4. The TPF system receives an NLP containing 20 bytes of data and a STATUS segment. Because this is the first NLP received after the IPL, the BSN RECEIVED field in the RTPCB entry is set to the BSN value of this NLP (520). The data message is processed normally.  
The ECHO number (104) in the STATUS segment does not match the current SYNC number (203); therefore, the RTP connection resynchronization process continues. The STATUS segment in this NLP is the reply to the status request sent out just before the IPL.
5. Another NLP is received containing a STATUS segment. This time the ECHO number (203) matches the current SYNC number; therefore, this is the reply to the status request sent out by the RTP connection resynchronization process. The RSEQ value in the STATUS segment indicates the next expected message that the remote RTP endpoint is waiting for starts with a BSN value of 450. The TPF system sets its output BSN (BSN SENT) field to 450 and places the connection back in CONNECTED state.
6. The RTPCB entry contains the values as shown. The RTP connection resynchronization process is completed successfully and outbound data traffic continues.

This example shows that the first STATUS segment received after the IPL does not necessarily contain the latest information. The RSEQ value of the first STATUS segment was 350, but NLPs with BSN values 350–449 were already sent before the IPL. The RTP connection resynchronization process must send its own status request after the IPL and wait for the reply to that status request to determine the correct RSEQ value.

### Enabling the RTP Resynchronization Process

The RTPRSYNC parameter on the SNAKEY macro in CTK2 enables the RTP resynchronization process for the TPF system. You can also use the ZNKEY command with the RTPRSYNC parameter specified to enable the RTP resynchronization process. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro. See *TPF Operations* for more information about the ZNKEY command.

---

## CP-CP Session Failures

If the CP-CP sessions are not active, no new APPN LU-LU sessions can be started. No new HPR LU-LU sessions can be started either. If CP-CP sessions are active and then break for any reason, active LU-LU sessions (APPN and HPR) are not affected; they remain active.

With HPR support, the CP-CP sessions are not only used for starting LU-LU sessions, but also are involved in the path switch process. If CP-CP sessions are not active, it is not possible to perform a path switch on an RTP connection. If the TPF system detects a failure in the route of an RTP connection, but the CP-CP sessions are not active, the path switch timer is started for that RTP connection. If the CP-CP sessions are activated before the path switch timer expires, the TPF system will begin the path switch process by sending a LOCATE request on the CP-CP sessions. However, if CP-CP sessions are never activated and the path switch timer expires, the RTP connection is cleaned up.

Activation of CP-CP sessions not only starts the path switch process for RTP connections that are in SWITCHING state, but will restart the path switch process if necessary. For example, the TPF system sends a LOCATE request to start the path switch process for an RTP connection, but the CP-CP sessions fail before the LOCATE reply is received. If the CP-CP sessions are reactivated before the path switch timer expires, the TPF system will send another LOCATE request to restart the path switch process.

---

## Flow Control

Flow control is an important aspect of the HPR architecture. In addition to the normal session level (LU-LU) pacing, end-to-end flow on an RTP connection is also regulated using the adaptive rate-based (ARB) algorithm to prevent one RTP endpoint from sending data faster than the network or remote RTP endpoint can process the data.

## ARB Pacing

The adaptive rate-based (ARB) algorithm is the primary congestion and flow control mechanism used by RTP connections. An RTP endpoint is allowed to send a certain amount of data to the remote RTP endpoint during a given interval. That amount of data is called the send rate. Periodically, an RTP endpoint sends an NLP containing an ARB request. The remote RTP endpoint sends back an ARB reply with one of five possible values based on current conditions at the remote RTP endpoint. The ARB reply value indicates whether the RTP endpoint (that sent the ARB request) can raise its send rate, keep the send rate the same, or reduce its send rate and, if so, by how much.

The send rate value for an RTP connection is adaptive and changes based on conditions in the network and remote RTP endpoint. The send rate is unidirectional, which means that on a given RTP connection the amount of data RTP endpoint A is allowed to send to RTP endpoint B is not necessarily the same as the amount of data RTP endpoint B is allowed to send to RTP endpoint A.

The possible values in an ARB reply are:

- |                   |                                       |
|-------------------|---------------------------------------|
| <b>Normal</b>     | Increase the send rate, if necessary. |
| <b>Restraint</b>  | Maintain the current send rate.       |
| <b>Slowdown 1</b> | Decrease the send rate by 12.5%.      |
| <b>Slowdown 2</b> | Decrease the send rate by 25%.        |
| <b>Critical</b>   | Decrease the send rate by 50%.        |

How an RTP endpoint decides which value to place in an ARB reply is implementation dependent. Typically, the value is based on availability of resources in that RTP endpoint node. The TPF system sets the value in an ARB reply based on how many core blocks are available. The percentage of available blocks for each core block type (frame, ECB, common block, IOB, SWB) is calculated and compared against the SNA shutdown value for that core block type. The SNA shutdown values are defined by the ILWPx parameters on the SNAKEY macro in CTK2. Initially, the ARB rate reply value is set as normal. As each core block type is checked, the ARB rate reply value can be left as is or changed to a more severe value. For example, the initial value is normal, but based on the availability of frames the ARB rate reply is changed to slowdown 1. The availability of SWBs is then checked and there are enough SWBs available to warrant setting the ARB rate reply value to normal; however, because the value is already set to slowdown 1, it

remains slowdown 1. Next, the availability of ECBs is such that slowdown 2 condition is reached. Because slowdown 2 is more severe than slowdown 1, the ARB rate reply is changed to slowdown 2.

If the value of the ILWPx parameter for a given core block type is X and the percentage of available blocks for that core block type is Y, the ARB rate reply value is set based on if Y is:

<b>X+20 or higher</b>	Normal
<b>X+10 to X+19</b>	Restraint
<b>X to X+9</b>	Slowdown 1
<b>X-1 to X-9</b>	Slowdown 2
<b>X-10 or lower</b>	Critical

For example, if the value of the ILWPF parameter is 40 (meaning SNA is shut down when the percentage of available frames is 40% or less) and the percentage of frames that are available is Y, the ARB rate reply is set based on if Y is:

<b>60 or higher</b>	Normal
<b>50–59</b>	Restraint
<b>40–49</b>	Slowdown 1
<b>31–39</b>	Slowdown 2
<b>30 or lower</b>	Critical

**Note:** If the value of an ILWPx parameter is less than 20, a value of 20 is used in the calculation. If the value of an ILWPx parameter is greater than 65, a value of 65 is used in the calculation.

The flexibility of the ARB rate reply values allows RTP endpoints to react immediately to low resource conditions and even prevents seriously low resource conditions from being reached in most conditions by reacting as soon as the first sign of trouble is detected.

Intermediate nodes along the route of an RTP connection also play a role in the ARB algorithm. If an intermediate node becomes congested, it can set the slowdown 1 or slowdown 2 flag in the NHDR of any NLP. The RTP endpoint receiving the NLP lowers its send rate accordingly if either of the slowdown flags are set in the NHDR.

The ARB algorithm is not the same as traditional window-based algorithms, such as PU 5 virtual route (VR) pacing. With a traditional window-based algorithm like VR pacing, a node is allowed to send a certain amount of requests and then cannot send more requests until a pacing response has been received from the remote node. VR pacing has caused deadlock conditions in the TPF system where a large queue of output messages builds up in the TPF system, the messages cannot be sent because the TPF system is waiting for a VR pacing response, but because so many blocks are in use, the TPF system is in an input list shutdown condition and not polling the NCP; therefore, the VR pacing response cannot be read in.

Because the ARB algorithm is time-based rather than window-based, deadlock conditions are avoided. An RTP endpoint is allowed to send a certain amount of data on an RTP endpoint in a given interval; for example, 500 bytes per second. If an RTP endpoint has messages totaling 2000 bytes to send, 500 bytes are sent

each second for 4 seconds. Each time 1 second passes, the RTP endpoint is allowed to send another 500 bytes. At no time is the RTP endpoint blocked because it is waiting for any response from the network or remote RTP endpoint. Unlike session-level (LU-LU) and VR pacing, there are no pacing requests and responses in the ARB algorithm. Instead, the ARB requests and replies adjust the send rates while data is flowing.

When more data is being generated by TPF applications than the allowed send rate for an RTP connection, the RTP output queue is used to control the rate at which output messages are sent. Even though the ARB algorithm controls the flow on RTP connections, session-level pacing for LU-LU sessions is still used and important. See “RTP Output Queue” for more information.

How often an RTP endpoint sends an ARB request depends on the implementation. The TPF system sends an ARB request at least every N messages sent on an RTP connection, where N is based on the sizes of the RTPCB table and HPRMT. Whenever the TPF system is sending an NLP that includes a status request, an ARB request is also included. Several actions can cause a status request to be sent, including an NLP after a path switch that contains the new route or when a heartbeat message is sent on an idle RTP connection. When traffic is flowing at a high rate on an RTP connection, it is important to ask for acknowledgements (by sending status requests) frequently to prevent the HPRMT from becoming full.

## RTP Output Queue

The RTP output queue holds output messages that cannot be sent immediately on the RTP connection for one of the following reasons:

- No path exists for the RTP connection (path switch in progress).
- The send window on the RTP connection is closed.
- The RTP connection resynchronization process is in progress for the RTP connection.

The RTP output queue does not replace the output message transmission (OMT) queue or the SOUTC queue. Each queue type provides a different level of service:

- An OMT queue relates to an LU-LU session.
- An RTP output queue relates to an RTP connection.
- A SOUTC queue relates to an ALS.

An output message can be on only one of these queues at a time. There is also a hierarchy here: Messages on the OMT queue move to either the RTP queue or directly to the SOUTC queue. Messages on the RTP queue always move to the SOUTC queue. Note that most output messages do not need to go on the OMT or RTP queues and are placed directly on the SOUTC queue. Table 6 explains when and why output messages are added and removed from the different queues.

*Table 6. Processing Output Message Queues*

Queue Type	Why a Message Is Added to the Queue	When Messages Are Removed from the Queue
OMT	The LU-LU session pacing window is closed.	The pacing response for the LU-LU session is received; messages are dequeued until the queue becomes empty or until the pacing window becomes closed again.

Table 6. Processing Output Message Queues (continued)

Queue Type	Why a Message Is Added to the Queue	When Messages Are Removed from the Queue
OMT	Message is too large to go out in a single PIU. The amount of data is larger than the maximum request unit (MAXRU) size of the LU-LU session. The data needs to be split into smaller pieces.	Immediately after the data has been broken into smaller pieces as long as the LU-LU session pacing window is not closed.
RTP	When no path exists for the RTP connection because a path switch is in progress.	The path switch is completed successfully.
RTP	The send window on the RTP connection is closed. The TPF system is not allowed to send more data on this RTP connection during the current time interval.	The current time interval expires; the TPF system is allowed to send another window of data. Messages are dequeued from the RTP output queue until the queue becomes empty or until the send window becomes closed again.
RTP	The RTP connection resynchronization process is in progress for this RTP connection.	The RTP connection resynchronization process is completed successfully.
SOUTC	The PIU is ready to be sent to the network.	The PIU has been received by the ALS.

The main purposes of the RTP output queue are as follows:

- A place to queue output messages for an RTP connection when there is no ALS (route) currently assigned to the RTP connection. While a path switch is in progress, the TPF application programs can still generate output messages. For LU-LU sessions that are paced, output messages are placed on the RTP output queue until the pacing window is closed, and then subsequent messages for the session go on the OMT queue. All output messages for sessions that are not paced go on the RTP output queue during a path switch. Messages remain on the RTP output queue until the path switch process is completed successfully.
- Flow control. Once a message goes on the SOUTC queue, that message will be sent to the network. If TPF applications are sending data faster than the current send rate for the RTP connection, the RTP output queue is used to throttle the rate at which data is sent on the RTP connection.

For performance reasons, the RTP output queue is a core memory table only, similar to the SOUTC queue. As with the SOUTC queue, items on the RTP output queue are regular core blocks (frames). Session-level pacing is still necessary for LU-LU sessions to prevent the TPF system from running out of core blocks. This is true for all types of LU-LU sessions: PU 5, PU 2.1, and HPR. Not all LU-LU sessions need to use session-level pacing. For example, if the application profile is such that one message in generates one message out, and a second message cannot be sent in until the response to the first message is received, the session is self-paced and session-level pacing is not needed. However, other application profiles, like those using pipeline sessions, require session-level pacing. If a TPF application is generating many output messages that cannot be sent out (for example, the ALS is in slowdown mode), those messages need to be queued on file on the OMT queue. If session-level pacing is not used, the pacing window will never close and the OMT queue will not be used. The result is that these output



messages (which are core blocks) will go on the RTP output queue or the SOUTC queue causing the queues to become very large.

If the size of an RTP output queue becomes too large, the TPF system will break the RTP connection. In doing so, all the core blocks on that RTP output queue are returned to the system. This mechanism is designed to prevent a single RTP connection from running the TPF system out of core blocks. However, session-level pacing should still be used to prevent the RTP output queue from becoming too large in the first place.

---

## HPR Output Messages

Building an output message for an HPR LU-LU session involves several steps. The following example explains the steps for building an NLP containing 100 bytes of user data:

1. The application issues the ROUTC macro pointing to the RCPL and a data level (core block) containing the 100 bytes of data to be sent. The core block can be any size (381, 1055, or 4 KB).
2. The core block is converted to FID1 format. The FID1 TH is built along with the RH. The user data is the RU portion of the NLP. This block is referred to as an HPR SOUTC type-A block. Figure 82 on page 173 shows what the HPR SOUTC type-A block looks like in this example.
3. A new 4-KB core block is obtained in the system virtual memory (SVM). RH and RU data is copied from the HPR SOUTC type-A block to near the end of the new block. At the end of the new block there is the NLP pad area, which is used to save information regarding the NLP being built. A FID5 TH is built right before the RH. The HPR SOUTC type-A block is then returned to the system. The new block is referred to as an HPR SOUTC type-B block. Figure 82 on page 173 shows what the HPR SOUTC type-B block looks like in this example.
4. If the message cannot be sent right away, the HPR SOUTC type-B block is placed on the RTP output queue. If the message can be sent right away or when the message is dequeued from the RTP output queue, continue with the next step.
5. The variable length NHDR and THDR are built. Next, the FID5 TH, RH, and RU are shifted up to just after the THDR. The LH is built now that the size of the NLP is known. A complete NLP has been built. The 4-KB core block containing the complete NLP is referred to as an HPR SOUTC type-C block. Figure 82 on page 173 shows what the HPR SOUTC type-C block looks like in this example.
6. The HPR SOUTC type-C block is placed on the SOUTC queue causing the NLP to be sent to the ALS.



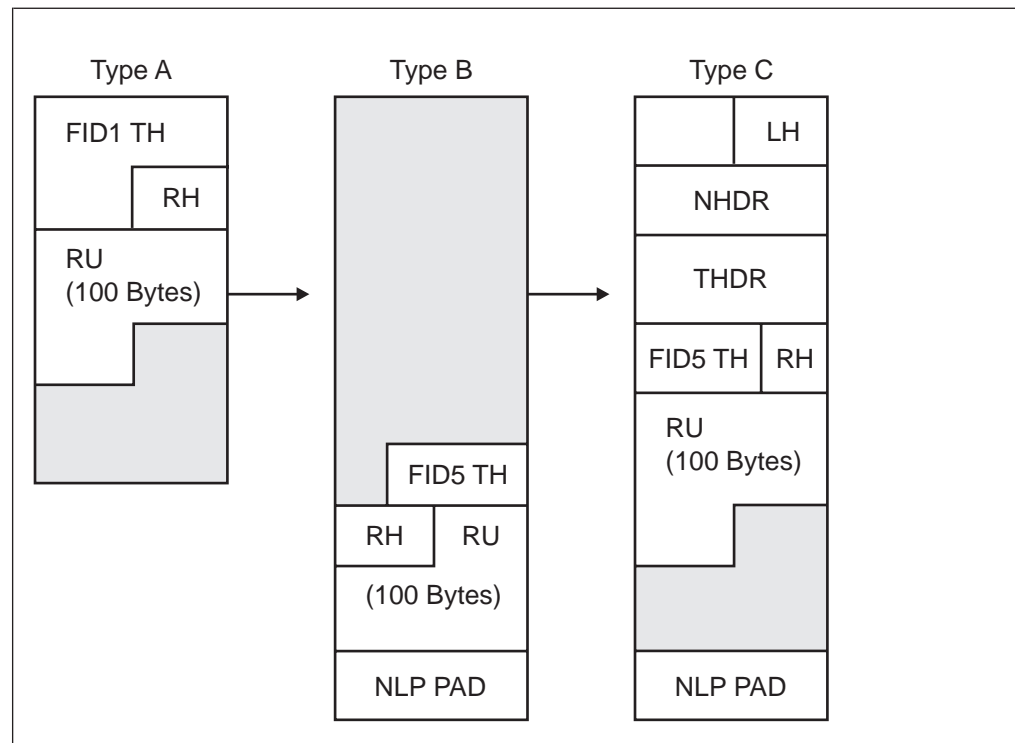


Figure 82. HPR SOUTC Block Types

If the data is too large to be sent in one NLP, the original FID1 block (HPR SOUTC type-A block) is converted to multiple HPR SOUTC type-B blocks. See “Segmenting Output Messages” on page 175 for more information.

Only HPR SOUTC type-B blocks exist on the RTP output queue. The NHDR and THDR portions of an NLP cannot be built until the NLP is ready to be sent out. For example, if a path switch is in progress, there is no current route; therefore, the routing information to put in the NHDR is unknown.

HPR SOUTC type-C blocks go only on the SOUTC queue. The beginning of the block up to and including the LH is identical to that of a FID2 PIU. This is necessary because both NLPs and FID2 PIUs can be sent on the same ALS (have blocks on the same SOUTC queue).

Some of the special HPR functions like message retransmission and sending a control message build their own HPR SOUTC type-B blocks to interface directly with the SOUTC code.

## Selective Retransmission

When an RTP endpoint sends an NLP, a copy of the data portion of the NLP is kept until the remote RTP endpoint acknowledges receipt of the data. If NLPs are lost in the network, the data needs to be retransmitted by the RTP endpoint.

HPR support uses a selective retransmission mechanism where data is not retransmitted unless instructed to do so by the remote RTP endpoint. Additionally, the remote RTP endpoint indicates which messages must be retransmitted. This sophisticated approach has the following distinct advantages over other algorithms:

- Comparing selective retransmission to the “go back N” approach, assume 10 messages are sent and messages 3 and 6 were lost in the network. Using “go

back N”, the remote node would indicate that the next message it is expecting is message 3, which would cause the sending node to retransmit messages 3–10 even though only two messages were lost. With HPR selective retransmission, the remote node would indicate that the next expected message is message 3, but the remote node did receive messages 4–5 and 7–10, which have been queued. This way, only messages 3 and 6 are retransmitted.

- When a node in the network is overloaded, data will be discarded and lost. If retransmission is triggered by a timer popping in the sending node, the same message would be sent over and over, which makes the network congestion problem even worse. In HPR support, the trigger for retransmission lies in the remote node, not the sending node. If the sending node does not receive an acknowledgement, that triggers a path switch rather than retransmission.

## Retransmitting Output Messages

When the TPF system sends an NLP that contains data, the data portion of the NLP is saved in the HPRMT. When the remote RTP endpoint acknowledges receipt of the data, the data is removed from the HPRMT.

If the TPF system receives an NLP with a STATUS segment indicating that data was lost, the lost data is retrieved from the HPRMT and retransmitted. If the TPF system is asked to retransmit data that does not exist in the HPRMT, the TPF system will break the RTP connection. This can happen when the HPRMT is not defined or if it is full.

**Note:** If the HPRMT is not defined or if it is full, this is not an error condition by itself; only if the TPF system is asked to retransmit data that is not in the HPRMT does this become an error condition.

See “HPRMT” on page 156 for more information about the HPRMT.

## Requesting That Input Messages Be Retransmitted

When the TPF system receives an NLP over an RTP connection, the byte sequence number (BSN) of the NLP is checked to see if this is the next expected message, a duplicate message, or a message received out of order. If it is the next expected message, the message is processed. If it is a duplicate message, it is discarded.

When a message is received out of order, it is queued on the RTP input queue and the TPF system asks the remote RTP endpoint to retransmit the missing data. When the missing data is retransmitted and received by the TPF system, the message on the RTP input queue is processed because it is now the next expected message.

The RTP input queue contains all messages received out of order for a given RTP connection. If the RTP input queue becomes too large or if the remote RTP endpoint does not retransmit the missing data after a certain amount of time, the TPF system will break the RTP connection.

---

## Segmentation and Reassembly

When an RTP connection is started, part of the ROUTE\_SETUP process is to find the smallest link size of all the hops along the route. This value is the *maximum link size* (MLS) and is the largest size of an NLP that can be sent across the RTP connection. The MLS value can change (increase or decrease) when a path switch occurs.

The function of intermediate (ANR) nodes is to route NLPs, not examine or process them. For that reason, intermediate nodes do not segment or reassemble NLPs. This means that when an RTP endpoint transmits an NLP, the size of that NLP cannot be greater than the MLS value for that RTP connection. If a message is larger than the MLS value, the message is segmented by the origin RTP endpoint and sent as multiple NLPs that are THDR chained. The first NLP is marked as start-of-message (SOM) and the last NLP is marked as end-of-message (EOM). The remote RTP endpoint will reassemble the pieces of the message and then pass the complete message to the application for processing.

The use of segmentation and reassembly is transparent to the application; therefore, the application does not know or have to worry about the MLS value. However, segmentation and reassembly does involve overhead at the RTP endpoints. To avoid that overhead, it is recommended that you configure your network with link sizes large enough to prevent segmentation from being necessary.

The minimum link size for an HPR link is 768 bytes. The NHDR and THDR sections of an NLP cannot be segmented.

### Segmenting Output Messages

When the SOUTC code builds an NLP, the NHDR and THDR are built first. Subtracting the size of the LH, NHDR, and THDR from the MLS determines how much data will fit in this NLP. If all of the data from the message will fit, segmentation is not necessary and the message is sent in a single NLP. However, if all of the data does not fit, the message must be segmented and sent as multiple NLPs.

The following example shows a THDR chained message:

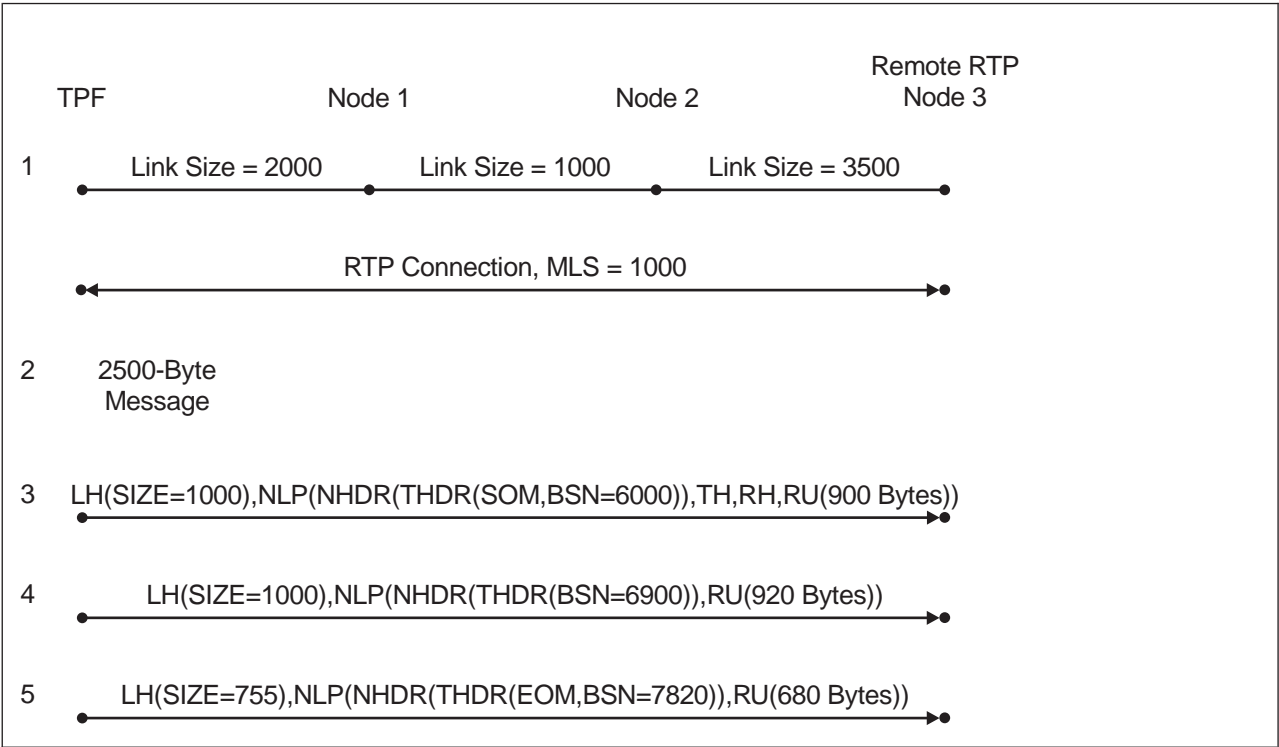


Figure 83. Segmenting an Output Message

In Figure 83:

1. An RTP connection exists between the TPF system and Node 3. The route has three hops with link sizes of 2000 bytes, 1000 bytes, and 3500 bytes. The MLS is the smallest of the link sizes, which is 1000 bytes.
2. A 2500-byte message needs to be sent on an LU-LU session that is over this RTP connection; therefore, the message must be segmented.
3. The first NLP for the message is built and marked as SOM. The combined size of the LH, NHDR, THDR, TH, and RH is 100 bytes in this example. With the MLS value of 1000, the first 900 bytes of data are sent in this NLP. Because there is more data to send in this message, the NLP is not marked as EOM.
4. The second NLP for the message is built. The combined size of the LH, NHDR, and THDR is 80 bytes this time. The next 920 bytes of the message are sent in this NLP. Because there is still more data to send, this NLP is not marked as EOM either.
5. The third NLP for the message is built. The combined size of the LH, NHDR, and THDR is 75 bytes this time. The remaining part of the message (the remaining 680 bytes) is sent in this NLP, and the NLP is marked as EOM.

In the previous example, the combined size of the LH, NHDR, and THDR was different for each NLP in the chained message. The LH size is fixed. For a given THDR chained message, the NHDR size and contents will be the same for all the NLPs that make up the message (unless a path switch occurs in the middle of sending the chained message). The THDR is variable length based on which optional segments are included.

## Reassembling Input Messages

When the TPF system receives a THDR chained (segmented) message, SNA Opzero reassembles the message and then passes the complete message to the application. When an NLP marked as SOM, but not EOM is received, this indicates the start of a chained message. The message is reassembled in a core block called the THDR chained input message block, which is pointed to by the RTPCB entry. As subsequent pieces of the message (NLPs) are received, the data from those NLPs is added to the core block to rebuild the message. When the NLP marked as EOM arrives, its data is added to the core block and the message is then passed to the application for processing.

Before HPR support, the TPF system supported TH chained messages for LU-LU sessions other than LU 6.2. The long message assembly (LMA) package puts a TH chained message back together before passing it to the application. With HPR support, TH chaining for LU 6.2 must be supported over RTP connections. SNA Opzero, not LMA, performs the TH chaining reassembly function for LU 6.2 sessions over RTP connections. This reassembly is very similar to THDR chaining reassembly except that the message is rebuilt in a core block called the TH chained input message block, which is pointed to by the SCB entry representing the LU 6.2 session.

An input message can be both THDR chained and TH chained.

---

## Installation and Tuning

To use HPR support, you must install it in your TPF system as well as in other components in the network. This section discusses only how to install HPR support in the TPF system. To install HPR support on another network component, refer to the documentation for that product.

The TPF Migration Guide: Program Update Tapes explains the necessary steps to install HPR support in the TPF system; for example, what new records and parameters must be defined. This section explains how to determine appropriate values for those new parameters. In logical order, the following are the tasks you need to perform:

1. Define the maximum number of HPR LU-LU sessions, which determines the size of the HPRSAT and becomes the value of the MAXHPRSA parameter on the SNAKEY macro in CTK2. See “Defining the HPRSAT” on page 155 for more information.
2. Define the maximum number of RTP connections, which determines the size of the RTPCB table and becomes the value of the MAXRTPCB parameter on the SNAKEY macro in CTK2. See “Defining the RTPCB Table” on page 154 for more information.
3. Define the #RT1RI and #RT2RI fixed file records that are used to keypoint parts of the RTPCB table. See “Defining Fixed File Records for the RTPCB Table” on page 155 for more information.
4. Define the size of the HPRMT, which becomes the value of the HPRMTSZ parameter on the SNAKEY macro in CTK2. See “Defining the HPRMT” on page 156 for more information.
5. Define the alive timer value, which becomes the value of the HPRALIVE parameter on the SNAKEY macro in CTK2. See “Alive Timer” on page 145 for more information.
6. Define the path switch timer value, which becomes the value of the HPRPST parameter on the SNAKEY macro in CTK2. See “Path Switch Timer” on page 143 for more information.
7. Decide whether or not to keep RTP connections active across a hardware IPL of the TPF system, which determines the value of the RTPRSYNC parameter on the SNAKEY macro in CTK2. See “Host IPL Considerations” on page 163 for more information.
8. Define the HPR flow control parameters, which are the ILWPE, ILWPF, ILWPI, and ILWPS parameters on the SNAKEY macro in CTK2. See “ARB Pacing” on page 168 for more information.
9. Define the size of the read buffers, which becomes the value of the UNITSZ parameter on the SNAKEY macro in CTK2. See “Network Considerations for NLPs” on page 152 for more information.

See the *TPF Migration Guide: Program Update Tapes* for more information about installing HPR support.

See *TPF ACF/SNA Network Generation* for more information about the SNAKEY parameters in CTK2.

---

## Diagnostic Information

Use the PIU trace facility to display the NLPs that are sent between the TPF system and remote RTP endpoints. See Appendix D, “Using the Path Information Unit (PIU) Trace Facility” on page 291 for more information about the PIU trace facility.

## Sense Codes Unique to the TPF System

When the TPF system breaks an RTP connection for a reason that is not architected, an implementation-specific sense code is sent in the connection fault

(CF) segment to indicate the reason for the failure. The following are the sense codes and reasons why the TPF system sends them:

- X'A001EE01'** The TPF operator issued a ZNRTP INACT command causing the connection to be unconditionally broken.
- X'A001EE02'** The TPF system is asked to retransmit a message, but the HPRMT is not defined; therefore, no saved message exists.
- X'A001EE03'** The TPF system is asked to retransmit a message, but the message was not found in the HPRMT.
- X'A001EE04'** The number of items on the RTP output queue for this RTP connection is too large.
- X'A001EE05'** The number of items on the RTP input queue for this RTP connection is too large.
- X'A001EE06'** The TPF system asked for data to be retransmitted, but the remote RTP endpoint did not retransmit that data.
- X'A001EE07'** Two NLPs marked as start-of-message (SOM) were received without an end-of-message (EOM) between them.
- X'A001EE08'** An NLP marked as middle-of-message was received without a start-of-message (SOM) being received first.
- X'A001EE09'** An NLP marked as end-of-message (EOM) was received without an NLP marked as start-of-message (SOM) having been received first.
- X'A001EE0A'** The TPF system sent a COB request to end the RTP connection, but a COB response was not received.

---

## TPF/SNA Control Block Structures

The following information describes the control block structures that are used by the TPF system for SNA communications.

---

### Resource Vector Table (RVT) and Related Control Block Structures

The RVT is a TPF/SNA control block structure located in main storage that contains information about the resources defined in the TPF system. Therefore, each entry in the RVT can be referred to as a resource definition. A resource definition includes information about a resource, such as the following:

- Network ID and resource name
- Resource ID (RID)
- Ordinal number of the fixed file NCB record (for resources defined using the OSTG program)
- File address of the NCB record
- Resource type
- Pointers to additional control block structures that contain more information about the resource.

Each processor in the TPF system has its own unique RVT. Therefore, resource definitions are not necessarily consistent across all of the processors in a loosely coupled TPF system. For more information about creating resource definitions, see “Defining SNA Resources to the TPF System” on page 193.

There are a number of other control block structures located in main storage that are used to access the RVT. These control block structures include:

- Resource name hash control table (RANHCT)
- Resource name hash prime table (RANHPT)
- Resource name hash entry table (RANHET)
- RVT available list
- RVT termination list.

The following information describes these control block structures and the RVT in more detail.

### How the RVT Is Organized

The RVT is divided into the following sections:

#### **Non-LU section**

Contains resource definitions for the following resources:

- Adjacent link station (ALS) resources
- Cross-domain resource manager (CDRM) resources
- Channel-to-channel (CTC) resources
- Network control program (NCP) resources
- Local system services control point (SSCP).

**Note:** This section of the RVT is also known as the ALS section.

#### **LU section**

Contains resource definitions for the logical unit (LU) resources.

Define the size of the RVT using the MAXRVT parameter in the SNAKEY macro.  
Define the size of the non-LU section using the NUMALS parameter in the SNAKEY

macro. The size of the LU section is implicitly defined by the difference of these two values. For more information about the SNAKEY macro, see *TPF ACF/SNA Network Generation*.

Each entry in the RVT is assigned an RID after the SNA fresh load function is performed during SNA restart. The RID is a direct index into the RVT. For example, RID 3 always corresponds to the third entry in the RVT.

Initially, RIDs are consistent across all of the processors in a loosely coupled TPF system. However, as new SNA resources are defined using the ZNDYN ADD command or dynamic LU support, the RIDs will no longer be consistent. For example, an LU resource can be assigned RID 5 on processor B and a different LU resource can also be assigned RID 5 on processor C.

Spare entries can exist in both the non-LU section and LU section of the RVT. A *spare entry* is an entry that is not currently being used for a resource definition. These entries are used to create resource definitions for resources that are defined using the following:

- Dynamic LU support
- ZNDYN ADD command
- ZNOPL LOAD command.

Figure 84 shows a simplified example of the RVT and how it is organized.

R V T			
Storage Address	Name	RID	
950	NCP001	000001	
1000	ALS026	000002	
1050	TPFB	000003	
1100	VTAM2	000004	
1150		000005	Non-LU Section
1200	LU05	000006	
1250	APPL	000007	
1300	TPFDB2T	000008	
1350	LU0ZZ	000009	LU Section
1400		00000A	

Figure 84. RVT Example

In Figure 84:

- There are 10 entries in the RVT (that is, MAXRVT=10).
- There are 5 entries in the non-LU section of the RVT (that is, NUMALS=5).
- There are 5 entries in the LU section of the RVT. This size is implicitly defined by the values of the MAXRVT and NUMALS parameters (MAXRVT-NUMALS).
- There is 1 spare entry in the non-LU section of the RVT (RID 000005).
- There is 1 spare entry in the LU section of the RVT (RID 00000A).



- The first and last entry of the RVT, which are referred to as the *RVT delimiters*, are not shown in the example.

**Note:** The last entry in the RVT contains all X'FF's.

## RVT-Related Control Block Structures

The following processor-unique control block structures are used to access the entries in the RVT, which is also a processor-unique control block structure. These control block structures are located in main storage and are built during SNA restart.

### Resource name hash control table (RNHCT)

A table that contains pointers to the other resource name hash (RNH) tables, the RVT available list, and the RVT termination list. It also contains statistical information about the RNH tables.

### Resource name hash prime table (RNHPT)

A table that contains entries referred to as RNHPT hash buckets. Each RNHPT hash bucket points to the first entry on its RNHET synonym chain and contains a count of the number of RNHET entries on that RNHET synonym chain.

Define the number of RNHPT hash buckets in the TPF system using the MAXPRIM parameter in the SNAKEY macro. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro.

### Resource name hash entry table (RNHET)

A table that contains an entry for each RVT entry. Each RNHET entry contains pointers that maintain the RNHET synonym chain, RVT available list, and RVT termination list. The RNHET entries also contain the RID and address of an RVT entry.

The *RVT available list* is a linked list of the RNHET entries for all of the available (or spare) RVT entries in the LU section of the RVT. When a new LU resource is defined to the TPF system, an RNHET entry is removed from the RVT available list, assigned to the LU resource, and placed on the RNHET synonym chain in the appropriate RNHPT hash bucket. The RNHET entry contains the RID and address of the RVT entry that will be used to create the resource definition.

Spare entries in the non-LU section of the RVT are **not** placed on the RVT available list. When new non-LU resources are defined to the TPF system, the TPF system sequentially searches the non-LU section of the RVT for a spare entry. If no spare entries exist in the non-LU section of the RVT, the TPF system uses the RVT entry of an ALS resource that is no longer active, as long as that non-LU resource was **not** defined using the OSTG program. Once an RVT entry is found for the new non-LU resource, the TPF system places the RNHET entry for that RVT entry on the RNHET synonym chain in the appropriate RNHPT hash bucket.

The *RNHET synonym chain* is a linked list of the RNHET entries that are assigned to a particular RNHPT hash bucket. One RNHET synonym chain exists for each RNHPT hash bucket.

The *RVT termination list* is a linked list of the RNHET entries for all of the LU resources that were defined using dynamic LU support and are no longer in session. (These resources are called dynamic LU resources.) When a dynamic LU resource ends its session, its RNHET entry is placed on the RVT termination list. (If the dynamic LU resource is an LU 6.2 resource, its RNHET entry is placed on the

RVT termination list only when the **last** session for that LU 6.2 resources is ended.) If a session is started again with that dynamic LU resource, its original RNHET entry is removed from the RVT termination list and reused. Because the RNHET entry contains the RID and address of the RVT entry for that dynamic LU resource, its original RVT entry is also reused.

If the original RNHET entry for a dynamic LU resource is no longer on the RVT termination list (that is, the RNHET entry was already reused by another dynamic LU resource), the TPF system uses the oldest RNHET entry on the RVT termination list. In doing so, the dynamic LU resource is assigned a new RVT entry and, therefore, a new RID as well.

**Note:** When a non-LU resource becomes inactive, its RNHET entry is **not** placed on the RVT termination list.

Figure 85 contains a simplified example of the RNHCT, RNHPT, and RNHET.

R N H P T			R N H E T							R V T		
Storage Address	First HET	SYN COUNT	Storage Address	SYN FORM	SYN BACK	HPT ADDR	NEXT AVAIL	RVT ADDR	RID	Storage Address	Name	RID
2000	3000	2	3000	3125	0	2000	0	1200	6	1200	LU05	6
2010	3025	1	3025	0	0	2010	0	1250	7	1250	APPL	7
2020	0	0	3050	3100	0	2030	0	1300	8	1300	TPFDB2T	8
2030	3050	2	3075	0	0	2040	0	1350	9	1350	LU0ZZ	9
2040	3075	1	3100	0	3050	2030	0	1400	A	1400	ROBIN	A
R N H C T			3125	0	3000	2000	0	1450	B	1450	BATMAN	B
			3150	0	0	0	3175	1500	C	1500		C
			3175	0	0	0	3200	1550	D	1550		D
			3200	0	0	0	3225	1600	E	1600		E
			3225	0	0	0	0	1650	F	1650		F
AVAIL TOP		3150										
AVAIL BOT		3225										
AVAIL COUNT		4										

Figure 85. RNHCT, RNHPT, and RNHET Example. The RVT1 delimiters are not included in this figure.

In Figure 85:

- Only the LU section of the RVT is shown in the example.
- There are 10 entries in the LU section of the RVT and, therefore, 10 entries in the RNHET in the example.
- Only the following RNHPT fields are shown in the example:

Field	Description
<b>FIRST HET</b>	Address of the first RNHET entry on the RNHET synonym chain.
<b>SYN COUNT</b>	Number of RNHET entries on the RNHET synonym chain.

- Only the following RNHET fields are shown in the example:

Field	Description
<b>SYN FORW</b>	Address of the next RNHET entry on the RNHET synonym chain.
<b>SYN BACK</b>	Address of the previous RNHET entry on the RNHET synonym chain.

**HPT ADDR** Address of the RNHPT hash bucket for this RNHET entry.

**NEXT AVAIL** Address of the next RNHET entry on the RVT available list if this RNHET entry is on the RVT available list.

**RVT ADDR** Address of the RVT entry for this RNHET entry.

**RID** RID of the LU resource that is assigned this RNHET entry.

- Only the following RNHCT fields are shown in the example:

Field	Description
-------	-------------

<b>AVAIL TOP</b>	Address of the first RNHET entry on the RVT available list.
------------------	---

<b>AVAIL BOT</b>	Address of the last RNHET entry on the RVT available list.
------------------	--

<b>AVAIL COUNT</b>	Number of RNHET entries on the RVT available list.
--------------------	--

- Five RNHPT entries, or RNHPT hash buckets, exist in the example.
- The RNHET entries for the following LU resources are on the synonym chain in the first RNHPT hash bucket:
  - LU05
  - BATMAN.
- Only the RNHET entry for the APPL LU resource is on the synonym chain in the second RNHPT hash bucket.
- There are no RNHET entries on the synonym chain in the third hash bucket.
- The RNHET entries for the following LU resources are on the synonym chain in the fourth RNHPT hash bucket:
  - TPFDB2T
  - ROBIN.
- Only the RNHET entry for the LU0ZZ LU resource is on the synonym chain in the fifth RNHPT hash bucket.
- There are 4 spare entries in the RVT. An RNHET entry exists for each of these spare RVT entries, and all of these RNHET entries are on the RVT available list.

The TPF system determines the RNHPT hash bucket for a particular resource using the DHASHC macro, which is a hashing function that is based on the name of the resource. See *TPF System Macros* for more information about the DHASHC macro.

### Displaying Information about the RVT-Related Control Block Structures

Enter the ZNDYN DISPLAY command to display the RNHET entry for a particular resource, the RNHET synonym chain for a particular resource, or statistical information about the RNH tables. See *TPF Operations* for more information about the ZNDYN DISPLAY command.

---

## Node Control Block (NCB) Records and Related Structures

The following information describes the NCB records and their related control block structures.

### Types of NCB Records

NCB records are processor-shared TPF/SNA records that contain information about the message queues for LU resources. The following types of NCB records exist in the TPF system:

- 381-byte fixed file NCB records
- 381-byte long-term pool file NCB records.

Although there are two different types of NCB records, the content of these NCB records is the same.

### 381-Byte Fixed File NCB Records

The 381-byte fixed file NCB records are assigned to the LU resources that are defined using the offline ACF/SNA table generation (OSTG) program. One 381-byte fixed file NCB record is created and assigned to each LU resource when the OSTG resource definitions are loaded to the TPF system using the SNA fresh load or dynamic load function.

**Note:** These NCB records are also assigned to non-LU resources that are defined using the OSTG program; however, the NCB records are not used by the TPF system.

Each NCB record is assigned an ordinal number, which is the same across all of the processors in a loosely coupled TPF system.

### 381-Byte Long-Term Pool File NCB Records

The 381-byte long-term pool file NCB records are assigned to LU resources that are defined using dynamic LU support. These types of resources are referred to as dynamic LU resources.

As many as 8 different 381-byte long-term pool file NCB records can exist for each dynamic LU resource. This allows a dynamic LU resource to log on to different applications and maintain separate message queues. The address of each NCB record assigned to a dynamic LU resource is stored in an entry in the NCB directory records. Each NCB directory record entry contains 8 NCB slots (0–7), which are used to maintain the addresses of the NCB records for a dynamic LU resource. See “NCB-Related Control Block Structures” for more information about NCB directory records.

You can use the DEVTYPE parameter in the MSGRTA macro to define for each application in the TPF system the NCB slot that is used when a dynamic LU resource logs on to that application. For example, if you want dynamic LU resources that log on to the APPL application to use the NCB record whose address is stored in the last NCB slot in the NCB directory record entry (that is, NCB slot 7), you would specify the option for the DEVTYPE parameter in the MSGRTA macro that corresponds to the last NCB slot. A dynamic LU resource can use different NCB records when it logs on to different TPF applications or it can use the same NCB record for all of the TPF applications.

See *TPF System Generation* for more information about the MSGRTA macro.

Unlike 381-byte fixed file NCB records, 381-byte long-term pool file NCB records are **not** assigned to non-LU resources. These NCB records are assigned to only LU resources and only when sessions are started with those LU resources.

## NCB-Related Control Block Structures

The following control block structures are used to organize and access the 381-byte long-term pool file NCB records that exist in the TPF system. These control block structures are located on file.

### NCB control record

A fixed file record that contains information about the NCB directory records.

### NCB directory records

Fixed file records that contain the addresses of the 381-byte long-term pool file NCB records assigned to a dynamic LU resource. Each NCB directory record contains 84 entries. Each entry contains the name of a dynamic LU resource and 8 NCB slots. Each NCB slot contains the address of a 381-byte long-term pool file NCB record if one exists for that NCB slot.

There are actually two types of NCB directory records defined in the TPF system; the current and staged NCB directory records. Specify the number of current and staged NCB directory records to define in the system initialization program (SIP) stage I deck when the TPF system is generated.

The TPF system uses the current NCB directory records to organize and access the 381-byte long-term pool file NCB records. The current NCB directory records are generically referred to as the NCB directory records.

The staged NCB directory records are used only by the NCB reorganization function to increase the number of NCB directory records in the TPF system.

Initially, the current NCB directory records have a record type of #NCBN4 and the staged NCB directory records have a record type of #NCBN5. Each time you perform the NCB reorganization function, these record types are swapped. That is, if the current NCB directory records have a record type of #NCBN4 before you start the NCB reorganization function, they will have a record type of #NCBN5 after the NCB reorganization function ends.

See “Increasing the Number of NCB Directory Records in the TPF System” on page 188 for more information about the NCB reorganization function.

When a session is started with a new dynamic LU resource, the TPF system creates a 381-byte long-term pool file NCB record and an entry in the NCB directory records for that dynamic LU resource. The address of the NCB record is stored in the appropriate NCB slot in the NCB directory record entry. The NCB slot used depends on the application with which the dynamic LU resource is starting a session.

**Note:** The NCB slot for an application is specified using the DEVTYPE parameter in the MSGRTA macro. See *TPF System Generation* for more information about the MSGRTA macro.

Entries are created in the NCB directory records for only dynamic LU resources. LU resources that were defined using the OSTG program are assigned 381-byte fixed file NCB records and, therefore, do not require an entry in the NCB directory records.

Figure 86 is a simplified example of the NCB directory records and how they are organized.

Ordinal Number 0				Ordinal Number 1				Ordinal Number 2			
NUMBER OF ENTRIES: 1				NUMBER OF ENTRIES: 0				NUMBER OF ENTRIES:			
ENTRY	NAME	NCB SLOT	NCB FILE ADDRESS	ENTRY	NAME	NCB SLOT	NCB FILE ADDRESS	ENTRY	NAME	NCB SLOT	NCB FILE ADDRESS
1	ROBIN	0	C0000090	1		0	0	1	BATMAN	0	C0000440
		1	0			1	0			1	C0000370
		2	0			2	0			2	C00001A0
2		0	0	2		0	0	2	JOKER	0	
		1	0			1	0			1	C0000560
		2	0			2	0			2	
3		0	0	3		0	0	3		0	
		1	0			1	0			1	
		2	0			2	0			2	
4		0	0	4		0	0	4		0	
		1	0			1	0			1	
		2	0			2	0			2	

Figure 86. NCB Directory Record Example. The RVT1 delimiters are not included in this figure.

In Figure 86:

- Each NCB directory record is assigned an ordinal number beginning with ordinal number 0. There are 3 NCB directory records (ordinal number 0–2) in the example.
- The example shows only 4 entries in each NCB directory record. Each NCB directory record in the TPF system actually has 84 entries.
- Entries exist in the NCB directory records for only the LU resources that were defined using dynamic LU support. In this example, these LU resources are BATMAN, ROBIN, and JOKER.

The NCB directory records do not contain entries for resources that were defined using the OSTG program because these resources use 381-byte fixed file NCB records.

- The example shows only 3 NCB slots (0–2) for each LU resource in the NCB directory record. The TPF system actually has 8 NCB slots (0–7) for each LU resource in the NCB directory record. NCB slots contain the address of an NCB record.
- The first NCB directory record contains an entry for the ROBIN LU resource, the second NCB directory record contains entries for no LU resources, and the third NCB directory record contains an entry for the BATMAN LU resource and the JOKER LU resource.
- One NCB record exists for the ROBIN LU resource. The address of this NCB record is stored in NCB slot 0. If ROBIN logs on to an application that uses NCB slot 0, the TPF system will retrieve this NCB record. If ROBIN logs on to an application that uses NCB slot 2, the TPF system will obtain a new long-term pool file NCB record and store its address in NCB slot 2.
- Three NCB records exist for the BATMAN LU resource and one long-term pool file NCB record exists for the JOKER LU resource.

The TPF system determines the ordinal number of the NCB directory record for a particular LU resource using the DHASHC macro, which is a hashing function that is based on the name of the LU resource. See *TPF System Macros* for more information about the DHASHC macro.

## Displaying Information about NCB Records and Related Structures

Enter the ZNNCB DISPLAY command to display information about a particular NCB record or the NCB directory records. See *TPF Operations* for more information about the ZNNCB DISPLAY command.

## Initializing NCB Records

The TPF system initializes all of the 381-byte fixed file NCB records each time a fresh load is performed.

You can also use the NCB initialization function at any time to initialize the NCB records in the TPF system. This function initializes both the 381-byte fixed file NCB records and 381-byte long-term pool file NCB records.

**Note:** An NCB record is initialized only if the LU resource assigned that NCB record is not in session.

Enter the ZNNCB command to start the NCB initialization function. See *TPF Operations* for more information about the ZNNCB command.

## Reclaiming NCB Directory Records and NCB Records

The TPF system creates 381-byte long-term pool file NCB records and NCB directory record entries for dynamic LU resources that start sessions with the TPF system. Periodically reclaim the NCB records and NCB directory records that are no longer in use. Otherwise, the TPF system can run out of resources and no new dynamic LU resources can start sessions with the TPF system.

You can use the NCB reconciliation function to reclaim the NCB records and NCB directory record entries that are no longer in use. This function returns to the TPF system the following resources:

- 381-byte long-term pool file NCB records that are no longer in use. An NCB record is not in use if the LU resource that is assigned that NCB record is no longer in session and no OMT queue exists for that LU resource.
- NCB directory record entries that are no longer in use. An NCB directory record entry is not in use if no NCB records exist for the LU resource that was assigned that NCB directory record entry.

See “To Run the NCB Reconciliation Function” on page 188 for information about how to run the NCB reconciliation function.

After you run the NCB reconciliation function, the following information is displayed:

- Largest number of entries being used in an NCB directory record
- Average number of entries being used in an NCB directory record
- Number of 381-byte long-term pool file NCB records returned to the TPF system
- Number of 381-byte long-term pool file NCB records still in use.

You can use this information to determine if you need to define more NCB directory records. For example, if the largest number of entries in an NCB directory record is approaching the maximum number of entries that can exist in an NCB directory



record (which is 84 entries), increase the number of NCB directory records in the TPF system. See “Increasing the Number of NCB Directory Records in the TPF System” for more information about increasing the number of NCB directory records.

### To Run the NCB Reconciliation Function

Use the following procedure to run the NCB reconciliation function and reclaim the unused NCB records and NCB directory record entries:

1. Enter the ZDSYS command to ensure that the TPF system is in NORM state.  
If the TPF system is not in NORM state, enter **ZCYCL NORM** to cycle the TPF system to NORM state.
2. Enter **ZNNCB DISPLAY ALL** to ensure that the NCB reorganization function is not running.  
If the NCB reorganization function is running, wait for it to be completed or end it before starting the NCB reconciliation function. See “Increasing the Number of NCB Directory Records in the TPF System” for more information about the NCB reorganization function.
3. Also ensure that the NCB initialization function is not running.  
If the NCB initialization function is running, wait for it to be completed before starting the NCB reconciliation function. See “Initializing NCB Records” on page 187 for more information about the NCB initialization function.
4. Enter **ZNNCB RECON ALL** to start the NCB reorganization function.  
The TPF system displays statistical information about the NCB directory records and NCB records when the NCB reconciliation function has been completed.

### To End the NCB Reconciliation Function

Once you start the NCB reconciliation function, you can end it at any time by entering the ZNNCB RECON command with the ABORT parameter.

If you are ending the NCB reconciliation function from a processor other than the processor where it was started, you must also specify the BP parameter for the ZNNCB REORG command.

### Additional Information

See *TPF Operations* for more information about the following commands:

- ZDSYS
- ZCYCL
- ZNNCB DISPLAY
- ZNNCB RECON.

## Increasing the Number of NCB Directory Records in the TPF System

When the NCB directory records start to become full, run the NCB reconciliation function to return the unused NCB directory record entries to the TPF system. If most of the entries in the NCB directory records are being used and the NCB reconciliation function does not return many unused NCB directory record entries to the TPF system, increase the number of NCB directory records in the TPF system using the NCB reorganization function.

See “Reclaiming NCB Directory Records and NCB Records” on page 187 for more information about the NCB reconciliation function.

The NCB reorganization function allows you to increase the number of NCB directory records in any TPF system state while the SNA network is active. The function is performed as follows:



1. The entries in the current NCB directory records are redistributed among the larger number of staged NCB directory records.
2. If a new LU resource logs on to the TPF system while the NCB reorganization function is running, an entry is created for that new LU resource in both the current and staged NCB directory records.
3. When all of the entries in the current NCB directory records are copied to the staged NCB directory records, the TPF system displays a message to prompt you to switch to the staged NCB directory records.  
Remember that after all of the entries are copied to the staged NCB directory records, there are fewer entries per record than there were in the current NCB directory records because more staged NCB directory records exist.
4. When you enter the command to switch the NCB directory records, the following occurs:
  - The staged NCB directory records become the new current NCB directory records.
  - The **old** current NCB directory records are initialized.
  - The **old** NCB directory records become the new staged NCB directory records.

There are now more NCB directory records defined in the TPF system and the NCB reorganization function is completed.

**Note:** There are now fewer staged NCB directory records in the TPF system than there are current NCB directory records. Therefore, if you need to increase the number of NCB directory records in the TPF system again, remember to increase the number of staged NCB directory records before you start the NCB reorganization function.

For more information about how to run the NCB reorganization function, see “To Run the NCB Reorganization Function”.

You can enter the ZNNCB DISPLAY command with the ALL parameter to display the number of current and staged NCB directory records that are defined in the TPF system.

### To Run the NCB Reorganization Function

Use the following procedure to run the NCB reorganization function and create more NCB directory records:

1. Enter **ZNNCB DISPLAY ALL** to ensure that the correct number of staged NCB directory records are defined in the TPF system.  
If there are not enough staged NCB directory records defined, define more before continuing.
2. Enter **ZNNCB DISPLAY ALL** to ensure that the NCB reconciliation function is not running.  
If the NCB reconciliation function is running, wait for it to be completed or end it before starting the NCB reorganization function. See “Reclaiming NCB Directory Records and NCB Records” on page 187 for more information about the NCB reconciliation function.
3. Also ensure that the NCB initialization function is not running.  
If the NCB initialization function is running, wait for it to be completed before starting the NCB reorganization function. See “Initializing NCB Records” on page 187 for more information about the NCB initialization function.
4. Also ensure that the online file recoup function is not running.

If the online file recoup function is running, wait for it to be completed or end it before starting the NCB reorganization function. See *TPF Operations* for more information about the online file recoup function.

5. Enter **ZNNCB REORG START** to start the NCB reorganization function.
6. When you are prompted by the TPF system, enter **ZNNCB REORG SWITCH** to complete the NCB reorganization function.

A completion message is displayed when the NCB reorganization function has been completed.

### To End the NCB Reorganization Function

Once you start the NCB reorganization function, you can end it at any time by entering the ZNNCB REORG command with the ABORT parameter.

If you are ending the NCB reorganization function from a processor other than the processor where it was started, you must also specify the BP parameter for the ZNNCB REORG command.

### Additional Information

See *TPF Operations* for more information about the following commands:

- ZNNCB DISPLAY
- ZNNCB REORG.

## Performance Considerations for Accessing NCB Records

There are no performance considerations for accessing NCB records in the TPF system.

The TPF system uses the CSNB segment to access NCB records. This segment uses the NCB directory record to determine the file address of the NCB record for a dynamic LU resource only when the session is first started. The CSNB segment then saves the file address of that NCB record in the RVT, which is located in main storage. The next time the TPF system tries to access the NCB record, the CSNB segment uses the address stored in the RVT.

See “Retrieving NCB and SPA Data Records (CSNB)” on page 287 for more information about the CSNB segment.

## Developing Applications That Retrieve NCB Records

When developing your own applications, you **must** use the CSNB segment to access NCB records. This segment returns the file address of the appropriate 381-byte fixed file NCB record or 381-byte long-term pool file NCB record based on the RID that it is given.

See “Retrieving NCB and SPA Data Records (CSNB)” on page 287 for more information about the CSNB segment.

## Allocating or Retrieving a Scratch Pad Area (SPA) for a Dynamic LU

The TPF system does not allocate an SPA ordinal when a dynamic LU is created. You can, however, assign a spare SPA ordinal to a dynamic LU with user exits CDLX and CDLY and save the SPA ordinal at RV1ORDN in the RVT. The SPA ordinal can be saved at RV1ORDN for dynamically created resources. For static resources, the NCB and SPA ordinal (generated by OSTG) will be saved at RV1ORDN.

If you intend to retrieve the SPA for a dynamic LU using the CSNB segment, initialize the SPA fixed file record before entering the CSNB segment or the CSNB segment will set an error return code and return to the calling segment.



---

## Defining SNA Resources to the TPF System

You must define SNA resources to the TPF system for the TPF system to communicate with them. You can define SNA resources to the TPF system using the following:

- Offline ACF/SNA table generation (OSTG) program
- ZNDYN ADD command
- Dynamic LU support.

Define the SNA resources as follows:

- Use the OSTG program to define the local resources, such as TPF applications and system services control points (SSCPs).

**Note:** When you define the local applications to the TPF system, you must use the application name table (ANT) deck, which is generated by the MSGRTA macros, as input to OSTG.

- Use the OSTG program to define the shared printers.
- Use the OSTG program or the ZNDYN ADD command to define the following remote resources:
  - Cross-domain resource manager (CDRM) resources
  - Channel-to-channel (CTC) resources
  - Network control program (NCP) resources.
- If the TPF system is running in low entry networking (LEN) mode, use the OSTG program to define the adjacent link station (ALS) resources.  
Otherwise, if the TPF system is running in APPN mode, you can also use the ZNDYN ADD command or dynamic LU support, as well as the OSTG program, to define the ALS resources.

**Note:** LEN mode or APPN mode is specified using the ZNAPN command.

- Use the OSTG program or dynamic LU support to define the remote LU resources.

See the following for more information about defining resources to the TPF system:

- “Using the OSTG Program to Define SNA Resources”
- “Using the ZNDYN ADD Command to Define SNA Resources” on page 198
- “Using Dynamic LU Support to Define SNA Resources” on page 199.

---

## Using the OSTG Program to Define SNA Resources

You can use the OSTG program to define all of the SNA resources to the TPF system. See *TPF ACF/SNA Network Generation* for more information about the OSTG program.

Once you define SNA resources using the OSTG program, you must load the resource definitions to the TPF system using the fresh load function or the dynamic load function. See “Loading Resource Definitions by Performing a Fresh Load” on page 194 for more information about the fresh load function. See “Loading Resource Definitions by Performing a Dynamic Load” on page 195 for more information about the dynamic load function.

## Important Considerations

When you generate the TPF system, you must use the IODEV macro to define the symbolic device address (SDA) and system characteristics for **each** of the following SNA resources that you define to the TPF system using the OSTG program:

- ALS resources
- CTC resources
- NCP resources.

Otherwise, although you can create the resource definition for the SNA resource, you cannot activate it.

See *TPF System Generation* for more information about the IODEV macro.

## Loading Resource Definitions by Performing a Fresh Load

The fresh load function replaces the current resource definitions in the TPF system with an entirely new set of resource definitions. That is, the resources that were previously defined to the TPF system using the OSTG program, the ZNDYN ADD command, and dynamic LU support are replaced with the new resource definitions during a fresh load.

The fresh load function also initializes all of the SNA control block structures in the TPF system, including the resource vector table (RVT), node control block (NCB) records, and the subarea address table (SAT). Therefore, when you perform a fresh load, you **must** first deactivate the network because any existing sessions are destroyed.

The fresh load process involves the following:

1. When you enter the ZNOPL LOAD command and specify the FRESH parameter, the new resource definitions are loaded to the current resource resolution table (RRT) and the old resource definitions are copied to the alternate RRT. At this point, the new resource definitions are loaded to the TPF system; however, no processors in the TPF system are using the new definitions yet.
2. As you perform an initial program load (IPL) on each processor in the TPF system, the SNA control block structures, including the RVT, NCB records, and SAT, are initialized on that processor. After the IPL has been completed on a processor, that processor is using the new (or current) resource definitions.

### Important Considerations

- The fresh load function destroys all active sessions. Therefore, you must deactivate the SNA network before you perform a fresh load.
- The fresh load function removes from the TPF system all of the resource definitions that were created using the ZNDYN ADD command or dynamic LU support.
- The NCB records and NCB directory records are initialized by the first TPF processor on which an IPL is performed.

### To Perform a Fresh Load

Use the following procedure to perform a fresh load:

1. Define the SNA resources using the OSTG program. See *TPF ACF/SNA Network Generation* for more information about the OSTG program.
2. Load the pilot tape or general data set (GDS) that was created by the OSTG program to the TPF system.

3. Deactivate the active resources in the TPF system. For more information about deactivating the resources in the TPF system, see “Activating and Deactivating Resources” on page 201.
4. Enter the ZNOPL LOAD command and specify the FRESH parameter to load the new resource definitions to the TPF system.  
A message is displayed that prompts you to enter the ZRIPL command.
5. Enter the ZRIPL command from each processor in the TPF system to perform an IPL and incorporate the new resource definitions.

**Note:** You can enter the ZNOPL STATUS command to determine the processors that require an IPL to incorporate the new resource definitions.

### Additional Information

See *TPF Operations* for more information about the following commands:

- ZNOPL LOAD
- ZNOPL STATUS
- ZRIPL.

## Forcing a Fresh Load during the Next IPL

Enter the ZNOPL BUILD command if you want to force a fresh load of the current resource definitions on 1, and only 1, processor during the next IPL.

The next time you perform an IPL on that processor, the SNA control block structures in the TPF system, including the resource vector table and the SAT, are initialized using the resource definitions in the current RRT.

**Note:** The NCB records are *not* initialized during the build function.

See *TPF Operations* for more information about the ZNOPL BUILD command.

## Loading Resource Definitions by Performing a Dynamic Load

The dynamic load function allows you to load new resource definitions to the TPF system without requiring you to perform an IPL or interrupt network communications. The dynamic load function does *not* initialize the SNA control block structures in the TPF system. It simply updates them with the new resource definitions. Therefore, you do not need to deactivate the network before you perform a dynamic load and the existing sessions are not destroyed.

In addition, the dynamic load function does *not* delete the resource definitions that were created using the ZNDYN ADD command or dynamic LU support. These resource definitions remain in the RVT after a dynamic load is performed.

The dynamic load process involves the following:

1. When you enter the ZNOPL LOAD command and specify the DYNAMIC parameter, the new resource definitions are loaded to the alternate RRT.
2. When you enter the ZNOPL UPDATE command, the alternate RRT becomes the current RRT, and the current RRT becomes the alternate RRT. At this point, the new resource definitions are loaded to the TPF system; however, no processors in the TPF system are using the new definitions yet.
3. As you enter the ZNOPL MERGE command on each processor in the TPF system, the RVT and its related tables are updated on that processor with the

new resource definitions. After the online merge function has been completed on a processor, that processor is now using the new (or current) resource definitions.

## Restrictions

- You cannot delete or redefine local resources using the dynamic load function. You must perform a fresh load to delete or redefine these resources.
- You cannot delete or redefine active SNA resources using the dynamic load function. You must first deactivate those resources. See “Activating and Deactivating Resources” on page 201 for more information about activating and deactivating SNA resources.
- You cannot add ALC terminals using the dynamic load function because ALC terminals are not defined in the RVT; they are defined in the WGTA. The same restriction applies to the data terminal equipment (DTE) that is supported by AX.25 and XALCI LU resources.

## To Perform a Dynamic Load

Use the following procedure to perform a dynamic load:

1. Define the SNA resources using the OSTG program. See *TPF ACF/SNA Network Generation* for more information about the OSTG program.
2. Load the pilot tape or GDS that was created by the OSTG program to the TPF system.
3. Enter the ZNOPL LOAD command and specify the DYNAMIC parameter to load the new resource definitions to the alternate RRT.  
A message is displayed that prompts you to enter the ZNOPL UPDATE command.
4. Enter the ZNOPL UPDATE command from one processor in the complex to switch the current and alternate RRT definitions.  
A message is displayed that prompts you to enter the ZNOPL MERGE command.
5. Enter the ZNOPL MERGE command from each processor in the TPF system to update the RVT with the new resource definitions.

**Note:** You can enter the ZNOPL STATUS command to determine the processors where you must enter the ZNOPL MERGE command to incorporate the new resource definitions.

## Additional Information

- If a processor is inactive when you perform a dynamic load, the RVT is automatically updated with the new resource definitions when you perform an IPL on that processor. There is no need to enter the ZNOPL MERGE command once the processor is activated.
- See *TPF Operations* for more information about the following commands:
  - ZNOPL LOAD
  - ZNOPL MERGE
  - ZNOPL STATUS
  - ZNOPL UPDATE
  - ZRIPL.

## Falling Back to the Old Resource Definitions

After you perform a dynamic load, you can use the fallback function to reload to the TPF system the old resource definitions that were saved in the alternate RRT, if necessary.



**Note:** You cannot fall back to the old resource definitions after you perform a fresh load. Instead, you must reload those resource definitions by entering the ZNOPL LOAD command.

When you use the fallback function, you can specify whether you want to reload the old resource definitions using the dynamic load function or the fresh load function.

### To Fall Back using the Dynamic Load Function

You can use the dynamic load function to fall back to the old resource definitions without interrupting network communications or destroying existing sessions.

Use the following procedure to fall back to the old resource definitions using the dynamic load function:

1. Enter the ZNOPL FALLBACK command to reload the old resource definitions from the alternate RRT to the current RRT.  
A message is displayed that prompts you to enter the ZNOPL MERGE command.
2. Enter the ZNOPL MERGE command from each processor in the TPF system to update the RVT with the old resource definitions.

**Note:** You can enter the ZNOPL STATUS command to determine the processors where you must enter the ZNOPL MERGE command to incorporate the old resource definitions.

### To Fall Back using the Fresh Load Function

If you cannot fall back to the old resource definitions using the dynamic load function, you can fall back to the old resource definitions using the fresh load function.

Use the following procedure to fall back to the old resource definitions using the fresh load function:

1. Deactivate the active resources in the TPF system. For more information about deactivating resources in the TPF system, see “Activating and Deactivating Resources” on page 201.
2. Enter the ZNOPL FALLBACK command and specify the FRESH parameter to reload the old resource definitions from the alternate RRT to the current RRT.  
A message is displayed that prompts you to enter the ZRIPL command.
3. Enter the ZRIPL command from each processor in the TPF system to perform an IPL and incorporate the old resource definitions.

**Note:** You can enter the ZNOPL STATUS command to determine the processors that require an IPL to pick up the old resource definitions.

### Additional Information

- See *TPF Operations* for more information about the following commands:
  - ZNOPL FALLBACK
  - ZNOPL LOAD
  - ZNOPL MERGE
  - ZNOPL STATUS
  - ZRIPL.
- See “Loading Resource Definitions by Performing a Fresh Load” on page 194 for more information about the fresh load function.
- See “Loading Resource Definitions by Performing a Dynamic Load” on page 195 for more information about the dynamic load function.

## Displaying Status Information about the Load Functions

You can use the ZNOPL STATUS command to display the following information about the RRT, SNA resource definitions, and load functions:

- Which section in the RRT contains the current resource definitions and which section in the RRT contains the alternate (or new) resource definitions.
- Whether the resource definitions in the current and alternate section of the RRT were loaded to the TPF system using the fresh load function or the dynamic load function.
- The date and time that the resource definitions were loaded to the TPF system.
- A description of the resource definitions that were loaded to the TPF system. This is the description that was specified for the DESC parameter in the PARM field of the OSTG JCL EXEC statement.
- Whether the update, fallback, and merge functions are enabled or disabled.
- Whether the build function will be performed during the next IPL.
- A list of all the processors in the loosely coupled TPF system, if these processors are active, and if they are using the current resource definitions.

See *TPF Operations* for more information about the ZNOPL STATUS command.

---

## Using the ZNDYN ADD Command to Define SNA Resources

You can use the ZNDYN ADD command online to define the following resources:

- ALS resources (if the TPF system is running in APPN mode)
- CTC resources
- CDRM resources
- NCP resources.

See *TPF Operations* for more information about the ZNDYN ADD command.

## Restrictions

You cannot use the ZNDYN ADD command to define ALS resources if the TPF system is running in LEN mode.

## Important Considerations

- Resource definitions that are created using the ZNDYN ADD command are removed from the TPF system if a fresh load is performed.
- When you generate the TPF system, you must use the IODEV macro to define the symbolic device address (SDA) and system characteristics for **each** of the following SNA resources that you define to the TPF system using the ZNDYN ADD command:
  - ALS resources
  - CTC resources
  - NCP resources.

Otherwise, although you can create the resource definition for the SNA resource, you cannot activate it.

See *TPF System Generation* for more information about the IODEV macro.

---

## Using the ZNDYN CHANGE Command to Change SNA Resource Definitions

You can use the ZNDYN CHANGE command online to change the name of an ALS, CTC, CDRM, or NCP resource. You can also use the ZNDYN CHANGE command to change the subarea for a CTC, CDRM, or NCP resource.

See *TPF Operations* for more information about the ZNDYN CHANGE command.

---

## Using Dynamic LU Support to Define SNA Resources

You can use dynamic LU support to create resource definitions for remote LU resources and ALS resources.

### Defining Remote LU Resources

Remote LU resources can log on to the TPF system without first being defined using the OSTG program. Instead, when a new remote LU resource tries to log on to an application in the TPF system, the TPF system automatically creates a resource definition and a 381-byte pool file NCB record for that remote LU resource, which allows the session to be started.

#### Restrictions

- You cannot use dynamic LU support to define the following resources:
  - Local TPF resources
  - Shared printers.
- You cannot add ALC terminals using the dynamic load function because ALC terminals are not defined in the RVT; they are defined in the WGTA. The same restriction applies to the data terminal equipment (DTE) that is supported by AX.25 and XALCI LU resources.

#### Important Considerations

- You must update the Dynamic LU user exit before you can use dynamic LU support to define LU resources to the TPF system. This user exit specifies which LU resources can log on to the TPF system using dynamic LU support and defines certain characteristics for these resources.

**Note:** Initially, the dynamic LU Definition user exit is set up so that no LU resources can be defined to the TPF system using dynamic LU support.

See *TPF System Installation Support Reference* for more information about the Dynamic LU user exit.

- If you use dynamic LU support to define a remote LU resource that uses a PSV routine, that PSV routine **must** first be defined to the TPF system in at least 1 OSTG RSC statement. Once the PSV routine is defined in an OSTG RSC statement, it can be used by any number of remote LU resources.
- The TPF system does **not** create a scratch pad area (SPA) for LU resources that are defined using dynamic LU support. Only LU resources that are defined using the OSTG program have an SPA in the TPF system.
- Resource definitions that are created using dynamic LU support are removed from the TPF system if a fresh load is performed.

## Defining Remote ALS Resources

If the TPF system is running in APPN mode, ALS links can be activated without first defining the ALS resource to the TPF system using the OSTG program. When an ALS resource that was not previously known to the TPF system tries to activate the PU 2.1 link, the TPF system automatically creates a resource definition for that ALS resource, which allows the link to be activated.

**Note:** The TPF system does not create an NCB record for ALS resources that are defined using dynamic LU support.

### Restrictions

You cannot use dynamic LU support to define ALS resources if the TPF system is running in LEN mode.

### Important Considerations

Resource definitions that are created using dynamic LU support are removed from the TPF system if a fresh load is performed.

---

## Activating and Deactivating Resources

Resources in a TPF/SNA system are defined as active or inactive. TPF only uses active resources. The TPF operator controls activation and deactivation of resources. Resources include:

- Applications
- NCPs and logical units
- Cross-domain resource managers
- Cross-domain resources
- Channel-to-channel (CTC) links in the same network
- APPN control points (CPs).

Resources are activated and deactivated via the ZNETW command.

See *TPF Operations* for complete description of the ZNETW command format.

Other resources in the named resource's hierarchy may be affected indirectly.

For example, a request to activate a local SSCP implies the activation of the application logical units under the control of the SSCP. If the SSCP is already active, only its logical units will be activated. If both the SSCP and its logical units are active, then no action is taken.

The ZNKEY command permits you to alter and/or display fields in the SNA communications keypoint.

The ZNPOL command is used to initiate/stop polling of a NCP/CTC device.

---

## Activating and Deactivating a Shared NCP

TPF may *not* load an NCP. This function must be performed by an MVS/VTAM network owner (CMC). When running in this configuration, the following considerations apply:

- Define to the TPF system all of the LU resources attached to the NCP that can communicate with TPF system. See “Defining SNA Resources to the TPF System” on page 193 for more information about defining remote LU resources to the TPF system.

If an operator deactivates an NCP or its resources, it does not affect other users of the NCP.

---

## Starting and Stopping Application Programs

TPF applications are defined during generation of the TPF system. Issuing the ZROUT command starts the application. It is then available to non-SNA terminals. Applications are available to the SNA network only after ZNETW activation. Thus, applications must be started (ZROUT) before the non-SNA network can access them; started (ZROUT) and activated (ZNETW) before the SNA network can access them.

TPF provides support to establish cross-domain application to application sessions. This is necessary because TPF applications cannot request a session. Instead, the TPF operator issues a ZNETW command with the LOGON operand. If the

cross-domain application is a primary LU, the TPF application will act as a secondary LU. During TPF system generation, users can define up to 255 secondary LUs to access an application. TPF attempts to establish a session for each secondary LU defined.

If a VTAM system supports receiving NOTIFY PIUs, and a VTAM LU requests a session with a TPF application that is not active, TPF will send a NOTIFY PIU to the requesting VTAM system when the TPF application is activated. The VTAM system may redrive the session request if the LU is still available.

Issuing a ZNETW INACT command deactivates a TPF application. This makes the application unavailable to the SNA network and terminates all sessions with the application's logical units. The application is still available to the non-SNA network. Deactivation types are the same as for other resources: orderly, immediate, and forced.

---

## Activating and Deactivating Cross-Domain Resource Managers

In the PU 5 environment, a CDRM to CDRM session must be established before resources in a TPF domain can communicate with resources in another domain. Here, the CDRMs work in tandem to activate, maintain, and terminate sessions between resources. The network address of the T5 SNA Network Interconnection (SNI) gateway CDRM is not defined to the TPF system until the session is activated. Therefore, the TPF system is unaware of the path the session will take. This enables the TPF system to dynamically discover the network address of the CDRM, which allows an alternate path to be used if the CDRM-CDRM session breaks.

Define to the TPF system each CDRM that can connect to the TPF system as a T5 node. See "Defining SNA Resources to the TPF System" on page 193 for more information about defining CDRM resources to the TPF system.

When the TPF system is connected to a VTAM system using both PU 5 links through SNI NCPs and APPN links, you must define an alias name for the CDRM in VTAM. See "SNI and APPN Considerations" on page 234 for more information.

A CDRM-CDRM session over a channel-to-channel link can be activated by the operator of either domain.

The operator of either domain can deactivate the CDRM-CDRM session. A deactivation request between domains is called a cross-domain takedown (CDTAKED). Cross-domain takedown requests include:

<b>CDTAKED orderly:</b>	Equivalent to TPF orderly deactivation.
<b>CDTAKED forced:</b>	Equivalent to TPF immediate deactivation.
<b>CDTAKED cleanup:</b>	Equivalent to TPF forced deactivation.

Either domain manager can perform two types of deactivation:

- **Disruptive CDRM inactivation** takes down all LU-LU sessions that were established using a specified CDRM and is accomplished through a forced, normal, or immediate inactivation of the CDRM.
- **Nondisruptive CDRM inactivation** does not take down any LU-LU sessions that were established using a specified CDRM. Nondisruptive inactivation is performed if the operator specifies that sessions should be saved across the CDRM deactivation when initiating a forced or immediate inactivation request on

a *remote* CDRM. See the ZNETW command in the *TPF Operations* for details on issuing a nondisruptive inactivation request.

Support of nondisruptive CDRM inactivation and reactivation is specified during CDRM session establishment. Control vector 6 in the ACTCDRM request and response contains this information. Nondisruptive inactivation can be performed regardless of whether the adjacent SSCP supports nondisruptive inactivation.

When the adjacent SSCP supports nondisruptive inactivation, the same sequence of PIUs are used to perform both disruptive and nondisruptive inactivation. The only differences between nondisruptive and disruptive inactivation are the type of cross-domain takedown (CDTAKED) and deactivate CDRM (DACTCDRM) commands and the status of the LU-LU sessions after the inactivation is completed.

**Note:** During nondisruptive CDRM inactivation, the associated LU-LU sessions which remained in session across the inactivation are disassociated from their owning CDRM. The consequence of this disassociation is the LU-LU sessions are not reassociated with the CDRM session when it is re-established. In order to inactivate the LU-LU sessions that survived a nondisruptive CDRM inactivation, an inactivation request must be specified on either NAU involved in the LU-LU session.

When the adjacent SSCP does **not** support nondisruptive inactivation, TPF treats the nondisruptive inactivation request as a session outage notification (SON). The operator should be aware that only the LU-LU sessions that support SON remain active across the inactivation. Both a nondisruptive immediate and forced inactivation of a CDRM that does not support nondisruptive inactivation results in SON, with only a flow of a single PIU (a DACTCDRM type 3). A nondisruptive immediate inactivation is escalated to a forced inactivation to perform SON.

**Note:** When TPF initiates a nondisruptive inactivation, TPF always interprets the first ACTCDRM request from the remote SSCP as an attempt to automatically recover the CDRM session. TPF then sends an 0858 negative response to the ACTCDRM request. If an automatic recovery was not intended, the operator can reissue the CDRM activation request, and TPF completes the CDRM reactivation successfully.

Once the TPF domain manager is in session with another domain manager, resources within the domains can communicate. The ZNETW command activates and deactivates cross-domain resources.

Cross-domain resources are also deactivated if:

- The communication path to the resource malfunctions, or
- The owning cross-domain resource manager (CDRM) is deactivated by means of disruptive inactivation.

Path malfunctions can occur in the communication line, NCP, 37x5, and/or owning CDRM.

Activation of a CDRM is non-disruptive to any ongoing cross-domain LU-LU sessions that may exist at the time of the activation. This feature enables a VTAM CMC to restart a failed CDRM-CDRM session without affecting any cross-domain sessions that survived the failure.



---

## Activating and Deactivating Control LU-Logon Manager Sessions

To activate PU 2.1 sessions when the TPF system is connected to the network as a LEN node, you must first establish a session between the TPF control LU (CLU) and the VTAM logon manager. To activate the session across an ALS connection, the VTAM operator must issue the following command:

**VARY NET, ACT, ID=*clu\_name*, LOGON=*elmngr***

Where:

*clu\_name*

The name of the control LU (CLU)

*elmngr*

The name of the Logon Manager application.

A CLU-Logon Manager session must be activated across each ALS connection between TPF and VTAM.

The Logon Manager can also be used to mediate sessions across a PU 5 CTC connection. In this case, the CDRM-CDRM session across the CTC link must first be activated. Then, the TPF operator can issue:

**ZNETW ACT ID=*clu\_name***

**ZNETW ACT ID=*elmngr*, LOGON=*clu\_name*, CDRM=*vtam\_cdrm\_name***

Where:

*clu\_name*

The name of the control LU (CLU)

*elmngr*

The name of the Logon Manager application

*vtam\_cdrm\_name*

The name of the remote cross-domain resource manager.

To deactivate the sessions, the VTAM operator must issue:

**VARY NET, INACT, ID=*clu\_name***

Alternatively, the TPF operator can issue:

**ZNETW INACT ID=*clu\_name***

---

## Activating and Deactivating APPN CP-CP Sessions

When the TPF system is connected to the network as an APPN end node, special control sessions exist between the TPF system and its network node server (NNS). An NNS is a network node that provides services to an end node (the TPF system in this case). The control point (CP) in the TPF system exchanges data with the CP in the NNS using CP-CP sessions.

CP-CP sessions are used during the LU-LU session activation process to exchange data between the TPF system and the APPN network. You cannot start new LU-LU sessions if the CP-CP sessions are not active. CP-CP sessions provide services that are similar to the services that CDRM-CDRM sessions provide to a PU 5



network. One key difference is that the deactivation of the CP-CP sessions for any reason does **not** cause the active LU-LU sessions to be deactivated. Only the LU-LU sessions that are in the process of being activated when the CP-CP sessions fail will be cleaned up.

When the TPF system is connected to a PU 5 network, each processor in the TPF loosely coupled complex can have CDRM-CDRM sessions to many remote hosts. In an APPN network, an end node can be connected to many different network nodes but can have active CP-CP sessions to only 1 network node. If the active CP-CP sessions fail, new CP-CP sessions can be started with the same network node or a different network node. Because the TPF loosely coupled complex is 1 end node, only 1 TPF processor has CP-CP sessions.

CP-CP sessions are critical resources; therefore, the TPF system automatically attempts to activate CP-CP sessions when the network is started. If the active CP-CP sessions fail unexpectedly (for example, because of a path failure), the TPF processor that had the CP-CP sessions will attempt to activate new CP-CP sessions.

When CP-CP sessions are not active, they can be activated by a TPF operator or by a remote operator. To activate CP-CP sessions from the TPF operator console, use the ZNETW ACT command. See *TPF Operations* for more information about the ZNETW ACT command.

CP-CP sessions can be deactivated by a TPF operator or by a remote operator. To deactivate CP-CP sessions from the TPF operator console, use the ZNETW INACT command. See *TPF Operations* for more information about the ZNETW INACT command.

If an LU is defined to the TPF system as a remote CP, the only sessions it can have are CP-CP sessions. The TPF system does not support sessions between a remote CP and a regular TPF application (that is, an application that is not a local TPF control point LU). In addition to having CP-CP sessions, the LU that represents the TPF CP can be in session with remote LU 6.2 nodes that are not control point LUs.

## CP-CP Session Considerations When Running Loosely Coupled

Only 1 processor in the loosely coupled complex can have CP-CP sessions. You can use the ZNAPN STATUS command to determine if CP-CP sessions are active and, if so, on which processor.

Before a TPF processor attempts to automatically start CP-CP sessions, the UACP user exit segment is called to determine whether this TPF processor is allowed to start CP-CP sessions. UACP allows you to limit CP-CP session activation attempts to a subset of the TPF processors that reside in the loosely coupled complex. See *TPF System Installation Support Reference* for more information about the UACP user exit.

Cycling down the TPF processor with the CP-CP sessions to below CRAS state prevents new LU-LU sessions from being established for the entire loosely coupled complex until that host is cycled up again. Cycling down the processor with the CP-CP sessions is not recommended and will result in an attention message on the operator console. If you need to cycle down the processor with the CP-CP sessions, do the following:

1. Deactivate the CP-CP sessions using the ZNETW INACT command.

2. Activate new CP-CP sessions on a different processor in the loosely coupled complex using the ZNETW ACT command.
3. Cycle down the processor that had the original CP-CP sessions.

See *TPF Operations* for more information about the ZNETW ACT and ZNETW INACT commands.

Follow the same procedure if you want to take the processor with the CP-CP sessions out of the complex. However, if the processor with the CP-CP sessions fails causing an unplanned outage, you must follow a different procedure. If the links to the failing processor have not reached their automatic network shutdown (ANS) time-out value, the links are still considered active from the point of view of the network. This means that the network considers the CP-CP sessions to still be active. Until those CP-CP sessions are deactivated, new CP-CP sessions cannot be started. This is why the TPF system does not attempt to automatically start new CP-CP sessions when the processor with the active CP-CP sessions fails.

When the processor with the CP-CP sessions fails, do the following:

1. Deactivate the failing TPF processor using the ZPSMS command.
2. Deactivate the links to the failing TPF processor from a remote operator console if the links are still active. This will cause the network to clean up the CP-CP sessions.
3. Enter **ZNETW INACT ID-CPCP,F** from one of the surviving TPF processors. This cleans up the processor-shared structures to indicate that CP-CP sessions no longer exist.
4. Activate new CP-CP sessions from one of the surviving TPF processors using the ZNETW ACT command.

See *TPF Operations* for more information about the ZNETW ACT and ZNETW INACT commands.

If the TPF processor with the CP-CP sessions performs either a hardware IPL or software IPL, the TPF system deactivates the CP-CP sessions and then attempts to activate new CP-CP sessions during cycle up. Only LU-LU sessions that are being activated are affected by this action, causing the session activations to fail; LU-LU sessions that are active are not affected by the deactivation and reactivation of the CP-CP sessions.

In a loosely coupled complex, LU-LU sessions on all TPF processors that are being activated will fail if an IPL is performed on the TPF processor with the CP-CP sessions. If an IPL is performed on a TPF processor that does not own the CP-CP sessions, only the LU-LU sessions that are being activated on that TPF processor will fail; LU-LU sessions being activated on the other TPF processors are not affected.

---

## Activating LU-LU Sessions

To activate an LU-LU session, the TPF system needs to know the SNA command to send and across which session to send that command. The decision is based on the following factors:

- Whether the TPF system is connected to the network as a PU 5 node, PU 2.1 node, or both.
- How sessions with the remote LU were activated previously.

- What path information, if any, was provided on the request that caused the LU-LU session to be activated.

This section discusses how the TPF system activates LU-LU sessions, meaning that the TPF system sends the first command in the session activation sequence. In this context, activating an LU-LU session does **not** mean which side is the primary LU (PLU) and, therefore, sends the BIND request to start the actual LU-LU session.

The discussion is broken into two parts, LU 6.2 and other LU types, because of the different triggers that activate these sessions and the different connectivity capabilities.

## Activating LU-LU Sessions Other Than LU 6.2

The ZNETW ACT command activates LU-LU sessions for LU types other than LU 6.2. The CDRM parameter, which is optional and only applicable to the TPF system connected as a PU 5 node, specifies the owning cross-domain resource manager (CDRM) of the remote LU that is being activated. The owning CDRM is not necessarily the CDRM that owns the remote LU but the CDRM through which the TPF system can access the remote LU.

Processing of the ZNETW ACT request depends on whether the CDRM parameter is specified and, if so, is that CDRM-CDRM session active.

- If the CDRM parameter is specified and the corresponding CDRM-CDRM session is active, that CDRM-CDRM session will be used to start the LU-LU session.
- If the CDRM parameter is specified and the corresponding CDRM-CDRM session is not active, the ZNETW ACT request is rejected.
- If the CDRM parameter is not specified, processing is based on other factors that will be discussed in subsequent sections.

## Activating LU 6.2 Sessions

An application program that issues the ALLOCATE verb causes an LU 6.2 session to be activated (if there are no active and available sessions already). Before the ALLOCATE verb is issued, a change number of sessions (CNOS) operation must have been done to initialize the mode name and session limits for the remote LU. This process can be done by an operator who enters the ZNCNS INITIALIZE command or an application program that issues the CNOSC INITIALIZE macro. Both of these have a pair of optional parameters, CDRM and CP. The CDRM parameter is identical in function to the CDRM parameter on the ZNETW ACT command.

The CP parameter can only be used when the remote LU 6.2 resource resides in an Advanced Peer-to-Peer Networking (APPN) node that is adjacent to the TPF system. The CP parameter specifies the control point (CP) name of the adjacent APPN node. In this case, the TPF system bypasses the normal APPN search, meaning the TPF system does not send a LOCATE request on the CP-CP sessions and, instead, sends a BIND request directly to the adjacent APPN node. User exit UALS selects the adjacent link station (ALS) over which the TPF system will send the BIND request.

Processing of a CNOS INITIALIZE request depends on whether the CDRM or CP parameter is specified and if that resource is active.

- If the CDRM parameter is specified and the corresponding CDRM-CDRM session is active, that CDRM-CDRM session will be used to start the LU-LU session.

- If the CDRM parameter is specified and the corresponding CDRM-CDRM session is not active, the CNOS INITIALIZE request is rejected.
- If the CP parameter is specified and there is at least one active ALS connecting the TPF system to the adjacent APPN node whose CP name was specified, a BIND request will be sent to the adjacent APPN node.
- If the CP parameter is specified and there are no active ALS links connecting the TPF system to the adjacent APPN node whose CP name was specified, the CNOS INITIALIZE request is rejected.
- If neither the CDRM parameter or CP parameter is specified, processing is based on other factors that will be discussed in subsequent sections.

**Note:** The CNOS INITIALIZE request can be performed a long time before the application program issues the ALLOCATE verb and network conditions could have changed. For example, the CDRM-CDRM session was active when the CNOS INITIALIZE request was performed but then the CDRM-CDRM session fails and is no longer active when the ALLOCATE verb is issued. In this case, the ALLOCATE verb will fail.

## PU 5 Environment

When the TPF system is connected to the network as a PU 5 node only, a CDINIT request is the first command sent to start the LU-LU session. Because the TPF system can be connected to many hosts, each with a CDRM-CDRM session, the owning CDRM must be known for the CDINIT request to be sent across the correct CDRM-CDRM session.

When a session is activated with the remote LU for the first time, the owning CDRM must be specified (the CDRM parameter must be specified). On subsequent session activation requests for the remote LU, the owning CDRM does not need to be specified, in which case the TPF system will use the same owning CDRM that was used the first time.

## PU 2.1 LEN Environment

When the TPF system is connected to the network as a PU 2.1 node only and is running in LEN mode, the only LU-LU sessions that can be started by the TPF system are LU 6.2 sessions and the PLU must reside in the TPF system. Do not code the CDRM parameter on CNOS INITIALIZE requests. Each time that a session is started, the TPF system will select a CLU-CLU session over which to send a REQTAIL request to the VTAM logon manager.

## PU 2.1 APPN Environment

When the TPF system is connected to the network as a PU 2.1 node only and is running in APPN mode, sessions other than LU 6.2 can be started by the TPF system and the PLU is not required to reside in the TPF system. Unlike PU 5 where there can be many CDRM-CDRM sessions, there is only one pair of CP-CP sessions because the TPF system is an APPN end node (EN).

### PU 2.1 APPN Environment: Sessions Other Than LU 6.2

Do not code the CDRM parameter on the ZNETW ACT command. Each time that a session is started, the TPF system will send a LOCATE request on the CP-CP sessions.

## PU 2.1 APPN Environment: LU 6.2 Sessions

Do not code the CDRM parameter on the CNOS INITIALIZE request. If the remote LU does not reside in an adjacent APPN node, do not code the CP parameter either. Each time that a session is started, the TPF system will send a LOCATE request on the CP-CP sessions.

If the remote LU resides in an adjacent APPN node and the PLU resides in the TPF system, code the CP parameter on the first CNOS INITIALIZE request with the remote LU. On subsequent CNOS INITIALIZE requests, the CP parameter does not need to be specified. If the CP parameter is not specified, the TPF system uses the same owning CP that was used previously if there is a path to it; otherwise, a LOCATE request is sent on the CP-CP sessions.

### Notes:

1. If the remote LU resides in an adjacent APPN node and that node is connected to or defined to the network node server (NNS) of the TPF system, you do not need to code the CP parameter because the LOCATE search on the CP-CP sessions will determine that the remote LU resides in a node adjacent to the TPF system. However, code the CP parameter in this case so that the session can be activated more effectively.
2. If the remote LU resides in an adjacent APPN node and that node is not connected to and not defined to the NNS of the TPF system, you **must** code the CP parameter.
3. If the remote LU moves to a different APPN node or the CP name of the adjacent APPN node changes, you must specify the CP parameter.

## Mixed PU 5 and PU 2.1 Environment

When the TPF system is connected to the network as both a PU 5 node and PU 2.1 node, the question becomes: What type of network search do you use to find the remote LU? In this context a mixed PU 5 and PU 2.1 environment means that the TPF system has both PU 5 and PU 2.1 links active; it does **not** mean that part of the LU-LU session path is in the PU 5 network and the other part of the path is in the PU 2.1 network.

The question of whether to search the PU 5 or PU 2.1 network is only relevant if the CDRM parameter (and CP parameter if LU 6.2) was not specified. There are two key considerations:

- Choosing the most effective link protocol if multiple paths to the remote LU exist. For example, the TPF system is connected to a VTAM system through PU 5 channel-to-channel (CTC) and also to the PU 2.1 APPN network (through 3745 devices). In this example, to access an application (LU) in the VTAM system, use the PU 5 CTC path because it is more effective.
- Migrating 3745 NCPs from PU 5 to APPN. An LU that used to be accessed through the PU 5 network is now accessed through the APPN network, or the opposite in fallback scenarios. If only a partial migration is done, some LUs will remain in the PU 5 network and others are now in the APPN network.

When migrating 3745s from PU 5 to APPN, the following rules are in place:

- Remote LUs that are accessed through PU 5 CTC links are not affected by the migration and will continue to use the PU 5 CTC path.
- If CP-CP sessions are active, to activate a session with a remote LU that is still accessed through a PU 5 NCP, you must code the CDRM parameter **each time** (on the ZNETW ACT command or CNOS INITIALIZE request), not just the first time.

- To activate a session with a remote LU that used to be accessed through a PU 5 NCP and is now accessed through an APPN link, do not code the CDRM parameter. A LOCATE request will be sent on the CP-CP sessions even if the owning CDRM is known.

### **Mixed Environment: Sessions Other Than LU 6.2**

If you did not specify the CDRM parameter on the ZNETW ACT command, the order of preference is as follows:

1. Use PU 5 (send CDINIT) if the owning CDRM is known and the CDRM-CDRM session is across a CTC link.
2. Use APPN (send LOCATE) if the CP-CP sessions are active.
3. Use PU 5 (send CDINIT) if the owning CDRM is known and the CDRM-CDRM session is across an NCP link.

### **Mixed Environment: LU 6.2 Sessions**

If you did not specify the CDRM parameter or CP parameter on the CNOS INITIALIZE request, the order of preference is as follows:

1. Use APPN (send BIND) to the adjacent APPN node if the owning CP is known and ALS links to that CP are active.
2. Use PU 5 (send CDINIT) if the owning CDRM is known and the CDRM-CDRM session is across a CTC link.
3. Use APPN (send LOCATE) if the CP-CP sessions are active.
4. Use PU 5 (send CDINIT) if the owning CDRM is known and the CDRM-CDRM session is across an NCP link.
5. Use LEN (send REQTAIL) if you are running in LEN mode.



---

## LU-LU Sessions in an APPN Network

Applications (LUs) that reside in the TPF system need to be defined to the APPN network. Because the entire TPF loosely coupled complex is a single APPN node, LUs that reside in one or all processors in the complex need to be defined to the network. Each LU in an APPN network is associated with its owner, which is the control point (CP) name of the node where the LU resides.

The TPF LUs can be predefined to the APPN network, or discovered dynamically at network startup time using the APPN LU registration process. The UARG user exit in the TPF system indicates which TPF LUs, if any, to register with the APPN network.

---

### Predefining TPF LUs to the APPN Network

Predefining TPF LUs to an APPN network is similar in concept to predefining TPF LUs to a PU 5 network. In a PU 5 network, each TPF LU is defined to the VTAM system in a CDRM deck using CDRSC statements. The CDRM under which the CDRSC is defined, referred to as the owning CDRM, indicates where the LU resides. In an APPN network, the owner of an LU is a CP rather than a CDRM.

In the definition deck of the node that is to be the network node server (NNS) for the TPF system, define each TPF LU as owned by the CP name of the TPF system.

If you predefine the TPF LUs to the APPN network, there is no need for the TPF system to register its LUs at network start up time; therefore, you should code the UARG user exit to not register any of the TPF LUs. See *TPF System Installation Support Reference* for more information about the UARG user exit.

---

### APPN LU Registration Process

When CP-CP sessions are activated between an APPN end node (EN) and network node (NN), the EN is expected to register all of its resources (LUs) with the APPN network if the resources have not been predefined. Being an EN, the TPF system will register all its LUs after the CP-CP sessions are activated.

The LU registration user exit, UARG, allows you to control which LUs, if any, will be registered with the APPN network. If you have not predefined the TPF LUs to the APPN network, use the default logic for UARG, which will register all of the TPF LUs. See *TPF System Installation Support Reference* for more information about the UARG user exit.

Predefining LUs eliminates the overhead of having to do LU registration whenever CP-CP sessions are activated; however, the LU registration process is more dynamic and makes adding new LUs much easier. Moving an application (LU) from one node to another node is also much easier when LU registration is used as opposed to predefining LUs and their owner.

---

## Remote Initiated LU-LU Sessions

All session initiation requests from remote LUs that want to go into session with TPF LUs will be received by the TPF processor in the loosely coupled complex that has the CP-CP sessions. The select a host user exit, UAPN, allows you to load balance sessions across all TPF processors in the complex.

APPN session initiation requests are called LOCATEs. When the TPF system receives a LOCATE request, UAPN is called to determine which TPF processor will be assigned this new LU-LU session. The LOCATE request is passed to the selected TPF host for processing. UAPN is also valuable in a uniprocessor TPF environment because it allows you to select the path (which ALS to use) for the session.

For LU-LU sessions started by a remote primary LU (PLU), the network rather than the TPF system will do the load balancing in most cases. The BIND request from the remote PLU can be received by the TPF system without any previous LOCATE request being received. Unlike LOCATEs, the BIND does **not** always flow to the TPF processor with the CP-CP sessions.

For LU-LU sessions started by a remote secondary LU (SLU), the network might do load balancing by providing a suggested route for the LU-LU session on the LOCATE request. If a suggested route is present, the name of the suggested ALS and the CPU ID of the TPF processor connected to that ALS are passed as input to the UAPN user exit. You have the option to use the suggested route, select your own route (TPF host, ALS, or both), or inform the TPF system to select a TPF processor for this LU-LU session.

See *TPF System Installation Support Reference* for more information about the UAPN user exit.



---

## Network Services Application Interfaces

TPF applications that communicate with end points accessed through an SNA LU require application-to-endpoint session control and status information. TPF provides this information to the application via TPF **Session Status Awareness** and **Session Management Request** services. The ISHLL macro (TPF SNA SHELL) defines the interface parameters and fields used between these TPF SNA service routines and the TPF application. This does not apply to LU 6.2 sessions.

---

### ISHLL Macro

The format of the ISHLL macro shown in Figure 87.

Operation	Operands
ISHLL	REG=reg

Figure 87. The ISHLL Macro

#### **REG=reg**

This required operand indicates the register to be used to establish addressability to the parameter area.

The ISHLL macro and the labels generated by this macro are documented in the various interface tables in “Session Management Request Services” and in “Session Status Awareness Services” on page 215.

---

### Session Management Request Services

The following services may be requested:

- Session Resynchronization - This is valid only for an LU using PSV with DFC ownership.
- Session Termination

The TPF SNA system segment (CSU0) is provided for applications to request the session services.

### Starting Session Management Request Services

To start a particular session service, a TPF application:

- Sets up the corresponding ISHLL interface parameter list.
- Initializes register 1 with the address of the ISHLL parameter list.
- Issues ENTRC CSU0.

CSU0 resides in all subsystems.

The ISHLL macro defines the parameter area that should reside in working storage. The parameter area settings for each service call are described in detail in Table 7 on page 214, Table 8 on page 214, and Table 9 on page 215.

The service program (CSU0) sets the return code field (ISHLRC) in the parameter area and returns control to the calling program.

Table 7. CSU0 Interface

Interface	Input to CSU0	Returned from CSU0
Registers	R1: ISHLL Parameter List Address	R0-R7: Not changed
ECB Workarea	<ul style="list-style-type: none"> <li>EBX012-EBX041 reserved for system use</li> <li>Remaining ECB fields irrelevant</li> </ul>	<ul style="list-style-type: none"> <li>EBX012-EBX041 unpredictable</li> <li>Remaining ECB fields not changed</li> </ul>
Data Levels	Irrelevant	Not Changed
Protect Key	Working Storage	Working Storage

## Requesting Session Resynchronization

Table 8. Request Session Resynchronization Interface

Label	Length (In Bytes)	Description
ISHLREQ	1	X'01' RESYNCH SESSION REQUEST
ISHLTYP	1	Reserved
ISHLRC	1	Return Code (output only) <ol style="list-style-type: none"> <li>X'00' Request scheduled</li> <li>X'11' PLU name conversion error.</li> <li>X'12' ISHLPNAM is not defined as a primary LU.</li> <li>X'13' PLU not available.</li> <li>X'21' SLU name conversion error.</li> <li>X'22' SLU already in session.</li> <li>X'23' SLU is pending activation.</li> <li>X'24' ISHLSNAM is defined as a primary LU.</li> <li>X'26' SLU not available.</li> <li>X'27' PLU/SLU domain error.</li> <li>X'41' Invalid request code.</li> <li>X'42' System is not in NORM state.</li> <li>X'43' CSCD can not schedule the request.</li> <li>X'44' Invalid request type.</li> <li>X'45' Resynch valid only for LU using PSV with DFC ownership.</li> </ol>
ISHLSW	1	Indicator, reserved for TPF internal usage.
ISHLSNAM	4	Secondary LU Name Address  The address of a 16-byte field that contains the NETID and the Network Qualified LU name of the resource functioning as secondary LU (SLU) for the requested session. The NETID is an 8-character string left-justified and padded with blanks. The name, which immediately follows the NETID, is also an 8-character string left-justified and padded with blanks.
ISHLPNAM	4	Primary LU Name Address  The address of a 16-byte field that includes the NETID and the Network Qualified LU name of the resource functioning as primary LU (PLU) for the requested session. The NETID is an 8-character string left-justified and padded with blanks. The name, which immediately follows the NETID, is also an 8-character string left-justified and padded with blanks.

## Requesting Session Termination Interface

Table 9. Request Session Termination Interface

Label	Length (In Bytes)	Description
ISHLREQ	1	Deactivate Session Request (X'02')
ISHLTYP	1	Type of Deactivation 1. Normal (X'01') 2. Forced (X'02')
ISHLRC	1	Return Code (output only) 1. X'00' Request scheduled 2. X'11' PLU name conversion error. 3. X'12' ISHLPNAM is not defined as a primary LU. 4. X'21' SLU name conversion error. 5. X'24' SHLSNAM is defined as a primary LU. 6. X'28' SLU is not in session. 7. X'29' SLU is not in session with ISHLPNAM. 8. X'2A' A zero PSV index is found in PSV name table. 9. X'41' Invalid request code. 10. X'42' System is not in NORM state. 11. X'43' CSCD can not schedule the request. 12. X'44' Invalid deactivation type requested.
ISHLSW	1	Indicator, reserved for TPF internal usage.
SHLSNAM	4	Secondary LU Name Address  The address of a 16-byte field that contains the NETID and the Network Qualified LU name of the resource functioning as secondary LU (SLU) for the requested session. The NETID is an 8-character string left-justified and padded with blanks. The name, which immediately follows the NETID, is also an 8-character string left-justified and padded with blanks.
ISHLPNAM	4	Primary LU Name Address  The address of a 16-byte field that contains the NETID and the Network Qualified LU name of the resource functioning as primary LU (PLU) for the requested session. The NETID is an 8-character string left-justified and padded with blanks. The name, which immediately follows the NETID, is also an 8-character string left-justified and padded with blanks.

---

## Session Status Awareness Services

A TPF application may require knowledge of LU session establishment and/or termination in order to determine whether to transmit, queue, or purge the output for that resource. You can specify session awareness support for LU sessions in the RSC statement of the OSTG input data set or by using the Dynamic LU user exit.

A 3270 welcome screen is also available through session awareness support for 3270 sessions. The TPF system provides a sample 3270 welcome screen. You can modify this sample to meet your needs.

See *TPF System Installation Support Reference* for more information about the Dynamic LU user exit and the sample 3270 welcome screen. See *TPF ACF/SNA Network Generation* for more information about the RSC statement and the OSTG program.

## Activating Session Status Awareness Services

If this notification is implemented, ENTRC activates CSXD with notification information for the following events:

- SESSION STARTED, data flow enabled.
- SESSION ENDED, planned or unplanned.

To notify the application/user of these events, TPF code passes the information to CSXD with the address of the ISHLL area contained in register 1. The fields for each event parameter list are described in Table 10 on page 216.

The released CSXD module contains a BACKC instruction.

Table 10. CSXD Interface

Interface	Input to CSXD	Returned from CSXD
Registers	R1: ISHLL ADDR	R0-R7: Not changed
ECB Workarea	<ul style="list-style-type: none"> <li>• EBW000-EBW103: Reserved</li> <li>• EBSW01-3: Reserved</li> <li>• EBR01: Reserved</li> <li>• EBCM01-3: Reserved</li> <li>• EBER01: Reserved</li> <li>• EBX000-EBX023: ISHLL</li> <li>• EBX024-EBX103: Irrelevant</li> </ul>	<ul style="list-style-type: none"> <li>• EBW000-EBW103: Unchanged</li> <li>• EBSW01-3: Unchanged</li> <li>• EBR01: Unchanged</li> <li>• EBCM01-3: Unchanged</li> <li>• EBER01 Unchanged</li> <li>• EBX000-EBX023: Irrelevant</li> <li>• EBX024-EBX103: Irrelevant</li> </ul>
Data Levels	1. D0-D7: not available 2. D8-DF: available	Same as on input.
Protect Key	Working Storage	Working Storage

## Starting Session Started Notification

Notification of this event is driven whenever TPF data flow is enabled for an SNA session. The TPF application can now send messages on this session. Any messages already on the TPF OMT queue are also sent.

When data flow is enabled, OMT processing issues an ENTRC to start the user replaceable module (CSXD) with the address of the Session Started Notification Interface in register 1. (See Table 11 on page 216 and Table 10 on page 216 for detailed information.) Labels for this area are defined using the ISHLL macro.

Table 11. Session Started Notification Interface

Label	Length (In Bytes)	Description
ISHLNOT	1	X'05' Session Started Notification
ISHLSS	1	Reserved (X'00' only value defined).
ISHLRC	1	Reserved (X'00' only value defined).
ISHLSW	1	Reserved (X'00' only value defined).

Table 11. Session Started Notification Interface (continued)

Label	Length (In Bytes)	Description
ISHLSNAM	4	<p>Secondary LU Name Address</p> <p>The address of a 16-byte field that contains the NETID and the Network Qualified LU name of the resource functioning as secondary LU (SLU) for the requested session. The NETID is an 8-character string left-justified and padded with blanks. The name, which immediately follows the NETID, is also an 8-character string left-justified and padded with blanks.</p>
ISHLPNAM	4	<p>Primary LU Name Address</p> <p>The address of a 16-byte field that contains the NETID and the Network Qualified LU name of the resource functioning as primary LU (PLU) for the requested session. The NETID is an 8-character string left-justified and padded with blanks. The name, which immediately follows the NETID, is also an 8-character string left-justified and padded with blanks.</p>
ISHLPSV	6	<p>Process Selection Vector.</p> <p>Left-justified 6-character PSV name, padded with blanks, defined for the remote LU.</p>

## Starting Session Ended Notification

This event indicates that the session is no longer available for data transmission to or from the TPF application.

Notification of this event is driven whenever a session between a TPF application and a network resource is terminated. Possible reasons for this event and session outages are normal session termination and network failure.

When the session ends, Lost Terminal Processing issues an ENTRC to start the user replaceable module (CSXD) with the address of the Session Ended Notification Interface in register 1. (See Table 10 on page 216 and Table 12 on page 217 for additional information.) Labels for this area are defined using the ISHLL macro.

Table 12. Session Ended Notification Interface

Label	Length (In Bytes)	Description
ISHLNOT	1	X'06' Session Ended Notification
ISHLSS	1	<p>Session Status at time of failure.</p> <ol style="list-style-type: none"> <li>1. X'01' - Pending activation</li> <li>2. X'02' - In session and data flow allowed</li> </ol>
ISHLRC	1	Reserved (X'00' only value defined).
ISHLSW	1	Reserved (X'00' only value defined).

Table 12. Session Ended Notification Interface (continued)

Label	Length (In Bytes)	Description
ISHLSNAM	4	<p>Secondary LU Name Address</p> <p>The address of a 16-byte field that contains the NETID and the Network Qualified LU name of the resource functioning as secondary LU (SLU) for the requested session. The NETID is an 8-character string left-justified and padded with blanks. The name, which immediately follows the NETID, is also an 8-character string left-justified and padded with blanks.</p>
ISHLPNAM	4	<p>Primary LU Name Address</p> <p>The address of a 16-byte field that contains the NETID and the Network Qualified LU name of the resource functioning as primary LU (PLU) for the requested session. The NETID is an 8-character string left-justified and padded with blanks. The name, which immediately follows the NETID, is also an 8-character string left-justified and padded with blanks.</p>
ISHLPSV	6	<p>Process Selection Vector.</p> <p>Left-justified 6-character PSV name, padded with blanks, defined for the remote LU.</p>

---

## PU 5 FID4 Considerations

---

### NCP Support

In order to connect TPF to current level NCPs as a PU type 5, the TPF system must connect as a data host using exchange identification format 2 (XID) and use FID4 PIUs. In addition, the TPF system supports explicit route (ER) and virtual route (VR) protocols when using FID4. The TPF system supports ER0, VR0, and TP2 (transmission priority high) between TPF and a channel-attached SNI NCP.

### Channel Contact

A channel contact procedure is implemented between TPF and channel-attached NCPs in SNA 4.2; that is, an explicit PU type 5 to PU type 4 session separate and distinct from the SSCP-PU.T4 session. This PU-PU session is analogous to the PU-PU sessions that exists that is supported between NCPs, and is established and terminated using a formal channel protocol between the physical units. Such establishment and dis-establishment of PU-PU sessions is always initiated by the PU Type 5(TPF) at the direction of its SSCP (through CONTACT and DISCONTACT requests). This new procedure occurs as an integral part of the activation of a channel-attached NCP and is completely transparent to the network operator.

As part of the channel contact procedure, the 2 physical units exchange certain information using the XID Format 2 architected for PU-PU contact procedures. Among the information sent from TPF to the NCP is the channel transmission group (which will always be 1) and other items concerning the environment in which the 2 PUs agree to operate. Such information is supplied at PU generation (NCP generation for NCPs and when the resource definition is created for the TPF system). As a result of this, using certain parameters of the NCP's HOST macro changes.

INBFRS, the number of buffers allocated by an NCP to receive data from a host, and MAXBFRU, the number of buffers allocated by a host to receive data from an NCP, is sent to the NCP in the XID Format 2. TPF obtains the MAXBFRU value from the SNAKEY macro in CTK2.

The size of host buffers used to receive data from an NCP (formerly obtained by the NCP from UNITSZ) is sent to the NCP in the XID. The number of pad characters prefixing NCP transmissions to the host (formerly obtained by the NCP from BFRPAD) is always sent to the NCP as zero in the XID. TPF obtains the UNITSZ value from the SNAKEY macro value in CTK2.

For the case of a host subarea attempting channel contact for a subarea which the NCP already has active on another channel, the NCP normally rejects the XID. The host can prevent this rejection and force acceptance (with the consequent breaking of contact on that other channel) by setting a bit in the XID Format 2. TPF does not support a host backup over a different channel adapter. Since TPF is designed to keep the network up and running, the backup host connects on the same channel adapter with the same subarea and does not issue the channel contact to break the connection. Whenever possible TPF continues from the point of failure. However, when the failing host was not able to checkpoint the fact that the NCP had been in contact, TPF issues a channel contact with the bit set in the XID to allow acceptance of the channel contact.

## XID Format 2 Sent by TPF

This section describes the format and contents of the XID format 2 command sent by TPF. The values TPF uses are shown. Values reserved or not used are not shown.

- 0 bits 0-3, Format of XID I-Field.
  - X'2' Format 2 (For T5 to T4 Node exchanges), bytes 0-41 are included.
  - bits 4-7, Type of the XID-Sending Node.
    - X'4' Subarea node
- 1 Length, in binary, of variable-format XID I-field (bytes 0-41).
- 2-5 Node Identification
  - bits 0-11, BLOCK NUMBER: The IBM Product Number for TPF. TPF will set block number to X'000' to indicate a node identification that is not unique.
  - bits 12-31, ID number: a binary value that, together with the block X'0' number, identifies a specific station uniquely within a customer network installation. TPF does not support ID number and will set this field to zero (0).
- 6-p Format 2 Continuation
- 6-7 Reserved
- 8 Characteristics of TPF node:
  - bit 0, TG Status:
    - 0 TG inactive
  - bit 1, multiple-link TG support:
    - 0 not supported
  - bits 2-3 segment assembly capability:
    - 10 segments are assembled on a session basis.
  - bits 4-7 reserved
- 9 FID types supported
  - bit 0
    - 0 FID 0 not supported
  - bit 1
    - 0 FID 1 not supported
  - bits 2-3, Reserved
  - bit 4
    - 1 FID 4 supported
  - bits 5-7, Reserved
- 10 Reserved
- 11-12 Length in binary of maximum PIU XID sender can receive  
MAXBFRU\*UNITSZ
- 13 Transmission Group Number (TGN)
  - 1
- 14-17 Subarea address of XID sender
- 18
  - bit 0 Reserved



bits 1-4 error status

bits 5-7 Reserved

19           CONTACT or load status of XID sender

          00 CONTACT has been received by an XID command sender

20-27       IPL Load module name

          40....40 no information conveyed

28-29       Reserved

31           Number of buffers suggested by primary

          00 No suggestion made.

32-33       Number of Read command (MAXBFRU)

34-35       Number of Bytes allocated per Read (UNITSZ)

36           Number of pad bytes

          00 No pad

37

          bit 0   Reserved for primary

          bit 1   Reserved

          bit 2   1, accept XID if TG in contacted state on another channel

38-39       Reserved for primary

40-41       Reserved for primary

## NCP Considerations

NCP can operate with TPF as a PU 5 SSCP, where FID4 PIUs are used. The old values for the MAXBFRU and UNITSZ parameters (13 and 106 respectively) can be used in TPF for FID4 support. However, MAXBFRU and UNITSZ can be increased to support larger message sizes in the network. The only necessary requirement is to ensure the class of service (COS) table pointed to on the NETWORK macro for the TPF network has entries for ISTVTCOS and the default entry that matches TPF's requirements.

The following items must be specified in an NCP gen to define a channel-attached TPF system as a PU Type 5:

- The Auto Network Shutdown keyword should be coded as ANS=CONT on each PU statement in order that a CMC failure will not affect LU sessions with TPF.
- On the LINE statement defining a channel-adapter, an Attention Timeout value greater than the time necessary for TPF to detect and recover from an error condition should be coded. Typically, this is 1 minute for a software-initiated IPL of TPF.
- For NCP Version 5.3 or higher, on the LINE statement defining a channel adapter, code the MONLINK parameter as 'MONLINK=NO'.

### Sample Definition for an NCP Gen with TPF Channel-Attached as a PU Type 5 Node

See Figure 88 on page 222 for a sample definition for an NCP gen with TPF channel-attached as a PU Type 5 node.

**Note:** This sample definition only shows the specified keywords that are required to define a PU Type 5 node. User-supplied keywords are not included here.

```
*****
*      BUILD MACRO FOR N30H521                      * 00620003
***** 00630003
N30H521  BUILD MODEL=3745,                          *
          SUBAREA=511,                              *
          TYPGEN=NCP,                               *
          BFRS=80,                                  *
          MAXSESS=1000,                             *
          NEWNAME=N30H521,                          *
          PUNAME=N30H521,                          *
          NETID=VTAMNET,                             *
          SALIMIT=1023,                             *
          SLODOWN=12
***** 01320003
*      HOSTS MACRO                                  * 01330003
***** 01340003
TPFBH   HOST INBFRS=4,          TPF INPUT BUFFERS *
          MAXBFRU=13,          *
          UNITSZ=106,          TPF UNIT SIZE      *
          BFRPAD=0,           TPF  REQUIRES NO PAD *
          NETID=TPFNET,       *
          SUBAREA=11
*
* 06280003
***** 21950003
*      GROUP MACRO FOR CHANNEL ADAPTERS            * 21960003
***** 21970003
GR30CA  GROUP LNCTL=CA,
          CA=TYPE6,
          NCPCA=ACTIVE,
          NPACOLL=YES
*
* 23210003
***** 23220003
*      CHANNEL DEFINITION FOR VTAM PU5 *          23230003
***** 23240003
L30CA3  LINE ADDRESS=(10),
          CASDL=0,
          TRANSFR=52,
          DELAY=0,
          INBFRS=6,
          ISTATUS=INACTIVE,
          TIMEOUT=120
*
* 23320003
P30CA3  PU PUTYPE=5
*
* 23690003
```

Figure 88. NCP Gen with TPF Channel-Attached as a PU Type 5 Node

## CTC Support

With CTC support, TPF connects to an MVS/VTAM, VM/VTAM, or another TPF system. TPF appears to a remote host as a PU type 5 (T5) node using FID4 PIUs. TPF and the remote system exchange identification (XID) and establish sessions across a virtual route using the CTC connection.

## Pre-Channel Contact/Priming

TPF supports a CTC “priming” procedure that allows links (across devices that support the CTC architecture) to be activated without operator intervention. SNA restart places each attached CTC device in extended mode and ready state; this

generates an asynchronous interrupt to the other side of the CTC adapter. The remote system initiates XID processing, if the link is pending active.

## Channel Contact

A channel contact procedure is implemented between TPF and channel-attached T5 nodes. In other words, there exists an explicit PU Type 5-to-PU Type 5 session. This PU-PU session is analogous to the PU-PU sessions supported between NCPs, and it is established using a formal channel protocol between the physical units.

As part of the channel contact procedure, the 2 physical units exchange information using the XID Format 2 architected for PU-PU contact procedures. Among the information sent from TPF to the T5 is the transmission group number and environmental characteristics under which the 2 nodes agree to operate. Such information is supplied at network generation (VTAM generation and when the resource definition is created for the TPF system).

The XID process for SNA CTC connection consists of the initiating host (X-side) sending an XID0 (XID format 2 with byte 19 set to 0) to the receiving host (Y-side). See "XID Format 2" on page 225 for detailed information about XID format 2. If the other host is not able to connect at this time, the operation times out and the initiating host will wait for the other host to activate the connection. XID0 is then allowed 30 seconds to complete.

If the other host is able to connect, it receives an attention interrupt from the write control (WCTL) CCW that began the initiating host's (X-side) channel program. The receiving host (Y-side) then issues its XID channel program, beginning with a sense command byte (SCB) CCW. The Y-side channel program consists of a read CCW for the X-side's XID0 and a write CCW for the Y-side's XID0.

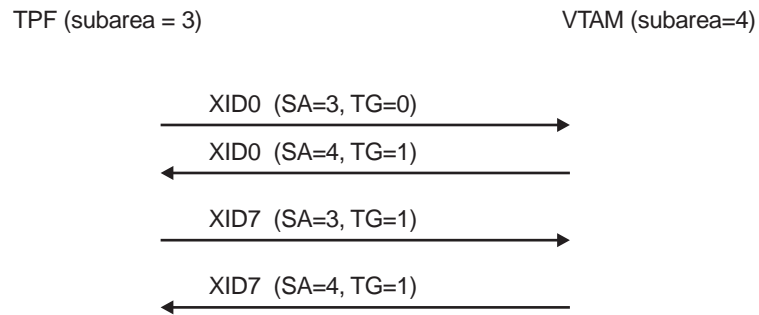
TPF checks the XID returned by the remote node for any error status set (byte 18, bits 1-4) and if the value is non-zero, TPF terminates the channel contact procedure.

The side with the lower subarea will send XID7 (byte 19 set to 07) if it can accept the CTC connection. When the other side sends its XID7, the CONTACT procedure is complete and the explicit route operatives (ER.OPs) are exchanged. The CTC connection is now available for sessions, subject to the virtual route availability.

The transmission group (TG) number that TPF passes in the XID0 data is TG=0 (TG=ANY). If the remote system is VTAM, the TG number is specified in VTAM's PU definition. TPF requires that the TG number specified be a value of 1 or 2. If the remote system is TPF, then the system with the low subarea selects either TG=1 or TG=2.

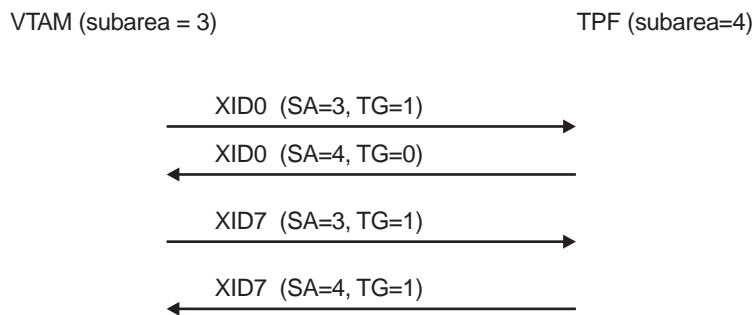
When XID processing has completed, a completion message is then sent to the originating operator.

See Figure 89 on page 224, Figure 90 on page 224, and Figure 91 on page 225 for illustrations of XID processing flows.



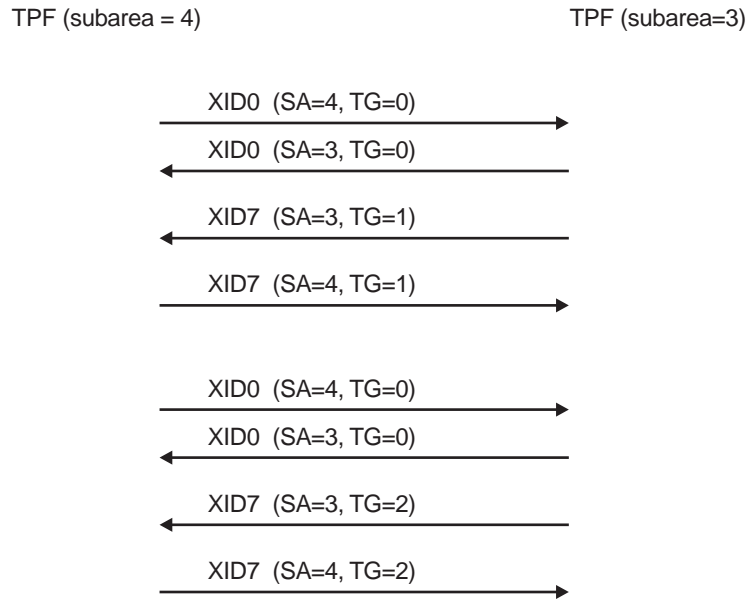
SA = subarea  
TG = transmission group

*Figure 89. XID Processing Example 1. TPF is the low subarea.*



SA = subarea  
TG = transmission group

*Figure 90. XID Processing Example 2. TPF is the high subarea.*



SA = subarea  
TG = transmission group

Figure 91. XID Processing Example 3. There are 2 links between TPF systems.

The following 2 parameters in the VTAM definition for the channel-attached major nodes must be correctly specified in order to attach to TPF. These parameters are sent to VTAM in the XID format 2.

- DELAY - The delay before a send I/O operation is initiated after a PIU of priority 0 or 1 is sent.
- MAXBFRU - The number of buffers allocated by a host to receive data.

**Note:** You should be aware of these parameters when specifying the value for the VTAM DELAY operand.

The number of 4K read buffers used by TPF for a CTC connection is specified in the CTCRBFR operand of the SNAKEY macro. The number of 4K write buffers in a buffer pool is specified on the CTCWBFRS operand of the SNAKEY macro. The buffer pool is used to allocate buffers for CTC write operations. TPF dynamically allocates write buffers to match the VTAM MAXBFRU value discovered at XID time.

## XID Format 2

This section describes the format and contents of the XID format 2 information sent by TPF. The values used by TPF are shown in the following XID header data example. Values reserved or not used are not shown.

The following is an example of the header that precedes XID format 2 for CTC links.

Byte	
0-1	Number of buffers transmitted
XL2'0001'	XID data occupies 1 4K buffer
2	Indicator byte
1.....	XID transfer
.1.....	Error Indicator
..000000	Reserved
3	CTC format indicator

XL1'01'	New protocol used
4-5	Number of 4K pages in input buffer
6-7	Reserved

The following is an example of the XID format 2.

Byte	
0	XID format and node type
0010....	Format 2
....0101	Type 5 node
1	Length, in binary, of variable-format XID I-field
2-5	Node identification and ID number
XL4'FFF00000'	Block number x'FFF0' and ID number 0
6-7	Reserved
8	Characteristics of TPF node:
0.....	TG status inactive
.0.....	Multiple link transmission group (TG) not supported
.10....	Segments reassembled on session basis
....0000	Reserved
9	FID types supported
0.....	FID 0 not supported
.0.....	FID 1 not supported
..00....	Reserved
....1...	FID 4 supported
.....000	Reserved
10	Reserved
11-12	Length in binary of maximum PIU XID sender can receive
X'0FF5'	4086 byte PIU
13	TG number
X'00'	TG0 (TG=ANY)
X'01'	TG1
X'02'	TG2
14-17	Subarea address of XID sender
18	Error status
0.....	Reserved
.1000...	XID parameters are incompatible
.1001...	Multi-link TG is incompatible
.1010...	TG number not defined
.1100...	Multi-link TG is not supported
.....000	Reserved
19	CONTACT or load status of XID sender
X'00'	CONTACT has been received by an XID sender
X'07'	XID response sender is already loaded
20-27	IPL load module name
XL8' '	No information conveyed
28-29	Reserved
30	DLC type
X'03'	System/370 to S370
31-36	Reserved

## Loosely Coupled Considerations

When multiple TPF processors are coupled together to share a common database, there are some important considerations to keep in mind: the IODEV addresses defined in SIP include all the SDAs for processors in the complex; that is, 1 SDA per TPF processor for each CTC connection to a VTAM or TPF host.

For SNA CTC connections between members of a loosely coupled complex, only CDRM-CDRM session and TPF/APPC sessions are supported.

## Session Initiation

Sessions can be established across a PU type 5 CTC connection using either a CDRM-CDRM session or a session between a Control LU (CLU) and the VTAM Logon Manager. When TPF connects to the subarea network, a CDRM-CDRM session is established. Then, an optional CLU session may be established with the Logon Manager. The CLU session is required if TPF/APPC sessions are to be brought up. In any case, there is a CTC configuration restriction on sessions that can be established: TPFs' subarea routing is limited to a single **hop**. (A hop is a link crossed on a path from 1 host or NCP subarea to another host or NCP subarea.) Therefore, the session partner must be in a host adjacent to TPF.

---

## VTAM Considerations

This section describes considerations for VTAM.

### Class of Service, Virtual Routes, and Transmission Priority

The capability exists in an SNA 4.2 level network to have three transmission priority levels (0, 1, and 2) and the traffic flowing in the network will be assigned 1 of the levels. TPF only supports high priority (2) for sessions across an SNI NCP. For SNA CTC, TPF only supports medium priority (1). This is because priority 2 traffic is not queued by VTAM, but is sent immediately.

TPF does not provide a class of service (COS) table; however, for TPF/APPC parallel sessions you can specify user-defined COS name with the PARACOS parameter on the SNAKEY macro. (See the *TPF ACF/SNA Network Generation* for more information on the SNAKEY macro.)

For communications with VTAM for a gateway NCP, it is recommended that a COS table contain VR0, TP2 as the first virtual route in any COS entries used where VTAM will be the primary side of a session with TPF (for example, CDRM). Otherwise, when activating sessions, VTAM will needlessly attempt to activate virtual routes that TPF will reject. TPF only supports 1 VR (VR0, TP2) and TPF has to be the primary side of the virtual route.

For SNA CTC, the COS name specified by VTAM must contain the following routes:

- VR0, TP1
- VR1, TP1

When VTAM requests a session for which it is the primary side (for example, ACTCDRM, CICS to SLU-P), VTAM chooses the VR. Because TPF only supports these routes, it is recommended that the ISTVTCOS and default COS table entries have the NCP or SNA CTC virtual routes as the first entry in the VR list.

On request for a session, TPF only selects the routes defined. Immediately after the XID exchange, TPF attempts to activate ERs and VRs. TPF responds negatively to

any attempt by VTAM or another TPF to activate a VR that does not request VR0 on TG1 or VR1 on TG2 because other VRs are not defined to TPF.

TPF requires that the virtual route use pacing windows with equal minimum and maximum values; there is no adaptive VR pacing. TPF enforces these restrictions on path definition:

- ER/VR0 **must** use TG1.
- ER/VR1 **must** use TG2.

It is recommended that TPF be the primary side of the virtual route to set pacing windows. If VTAM is allowed to be the primary side of the virtual route, the VRPSW01 in the PATH statement for the TPF subarea **must** set the minimum and maximum window size to be of equal values. This is because TPF does not support adaptive VR pacing.

See *VTAM Network Implementation Guide* and *NCP/SSP/EP Resource Definition Reference* for more information about transmission priorities and virtual routes.

## Other VTAM Considerations

When TPF is PU type 5 node:

- TPF will no longer perform application authorization for SNA resources. The VTAM session management exit must be used to ensure that the terminal is permitted to LOGON to the TPF application.
- VTAM USS definition tables should be used to transform the TPF-oriented LOGI requests to a VTAM equivalent (LOGON APPLID).
- TPF supports nondisruptive CDRM inactivation and reactivation. This support permits LU-LU sessions that were established using a specified CDRM to remain active and bound when the CDRM is deactivated through a nondisruptive inactivation request.

For CTC, with the definition of MAXBFRU, TPF accepts a dynamic buffer size from VTAM at XID time subject to available buffers from the pool. When coding the DELAY operand on the VTAM side, you should be aware that the TPF value is determined by the SNAPOLL parameter of the SNAKEY macro. This SNA polling time interval can be from 10 milliseconds to 50 milliseconds.

## VTAM Considerations for NCP 5.3

In the CDRM deck for TPF for NCP Version 5 Release 3 or higher, all GWPATH statements should be coded with the ADJNETEL parameter omitted. This causes VTAM to use the default value of 1 for the CDRM representing TPF. If any 37X5 with NCP 5.3 or higher is channel-attached to TPF, ADJNETEL=1 is required.

---

## LU-LU Session VR Assignment for CTC

The SNA Communication Route Selection Exit, CSJV, selects a virtual route for an LU-LU session during CDCINIT processing when the local LU is the PLU. For additional information on this exit, see *TPF System Installation Support Reference*.

---

## Virtual Route (VR) Activation

TPF attempts to activate VR0 to the destination subarea when the NCP is activated after explicit route 0 has been reported operative and activated.



If the explicit route associated with the virtual route being activated is operational and active, then an “activate virtual route” request (ACTVR) is sent from this subarea to the other end of the virtual route. On receipt of a positive response, the virtual route is marked active. It is now available for sessions to be assigned to it.

The virtual route deactivation (DACTVR) can only be requested by the primary virtual route end. This is the end that initiated the activation. However, if the other end denies the deactivation, then it becomes the primary.

To activate a selected virtual route, TPF first insures that the associated explicit route is active as described above. Then, an ACTVR request, which includes the maximum VR pacing window size, is sent to the physical unit at the other end of the route. On receipt of a positive response to the ACTVR request, the virtual route is “active,” but in a “hold” state as the network flow control mechanism has not yet opened the route for traffic. The positive ACTVR response includes a “VR Pacing Response” which drives the TPF flow control logic and causes the route to be opened for traffic and so able to support a session. If the ACTVR response does not include a “VR Pacing Response” then the VR remains in “Hold” state until an isolated response arrives. Upon receiving the ACTVR response, TPF sends an isolated VRPRS to open the route for traffic from the other side. The route remains active unless TPF deactivates the virtual route, or the associated explicit route becomes inoperative.

TPF checks pacing window sizes, enabling it to set the window sizes necessary for TPF to connect to the network. Remember, TPF only supports ER0, VR0 to a channel attached SNI NCP.

With SNA CTC, activation of the virtual route is slightly different. Here, TPF supports VR0, VR1, and TP1 (transmission priority 1).

After the channel contact procedure is completed, but before sessions can be established, the virtual route(s) must be activated. TPF initiates sending ER\_OP and ER\_ACT immediately after channel contact. ACTVR is then sent after ER activation. Next, TPF checks an incoming ACTVR to ensure that:

- The ACTVR is for a defined VR
- The VR is for a single hop
- The ACTVR specifies that the minimum and maximum window values are the same (min=max).

This checking permits TPF-to-TPF CTC connections. TPF can now be the secondary side of the VR and any ACTVR meeting the previously described checks is accepted, if the VR is not already active.

Contention can occur if both sides send ACTVRs simultaneously. The contention winner is the node with the higher subarea (SA) value.

## Virtual Route Deactivation

TPF does not take down a virtual route when the session count becomes zero (as VTAM does). Further, it responds negatively to any attempt to deactivate the virtual route by an orderly DACTVR from NCP. Only the primary end of the virtual route can deactivate the route. TPF ensures that it is the primary side of the virtual route by denying any ACTVR request. However, in accordance with the architecture, TPF does respond positively to a DACTVR forced, and cleans up any sessions associated with the virtual route.

When the explicit route associated with a virtual route becomes inoperative, the associated virtual route becomes inactive regardless of the number of sessions assigned to it. Appropriate notification is given to the endpoints of sessions using the virtual route.

When the operator requests deactivation of an NCP, after all the sessions through the NCP have been brought down, a channel disconnect is sent for the normal, immediate and force options of ZNETW INACT. This causes the ER and VR to be reset by NCP.

In SNA CTC, VTAM deactivates the VR when the session count reaches zero. TPF denies the request with a sense indicating that a session is waiting to use the VR, and thereby becomes the primary side of the virtual route.

---

## Network Flow Control

The initial state of the virtual route once the ACTVR has been issued is “held waiting for a virtual route pacing response.” On receipt of the positive response to ACTVR then the virtual route is active and, if that positive response contains a VR pacing response, is open as well (for example, not **held**).

Once the virtual route is **open** each PIU is transmitted as a singly segmented PIU. TPF sets the pacing request bit on in the first PIU of the window and suspends transmission on the virtual route if a pacing response is outstanding and the window size has been exhausted. This should be an extremely rare case, as TPF sets the minimum and maximum window size to a user-defined value. The recommended value is 42 for NCPs, but the user can set them up to the maximum (255). The tradeoff is NCP buffer utilization versus TPF performance. NCP allocates buffers equal to 3 times the minimum value, so that a large value may cause buffer depletion. TPF queues messages waiting on a blocked virtual route in main storage until the VRPRS is received. A small value may cause extra delay in response time and cause congestion in TPF due to the length of time TPF buffers are committed to an entry.

Because TPF sets the minimum and maximum window size to the same value, TPF requires that the adjacent NCP contain the SNA network Interconnect (SNI) function and that all TPF traffic that is routed through intermediate nodes cross the gateway in the adjacent NCP. TPF checks that any ER activated is limited to a hop count of 1 to ensure the SNI connection.

Virtual route pacing responses received by TPF open the VR. When a VR pacing request is received, TPF must generate a VRPRS to the other end of the virtual route.

After receiving a VR pacing request TPF returns an isolated VRPRS PIU on that virtual route, including the transmission priority and virtual route number in the TH. TPF does not set “congestion detected” indicators.

TPF explicitly withholds the VR pacing response, based upon the values you can set in CTK2 for the different block types. The values indicate the percentage of available blocks that are needed for a VR pacing response to be sent. VR pacing responses can be withheld based upon 4K, 1055, 381, and ECB block types. A value of 0 indicates that VR pacing should not be withheld based upon this block type. (See *TPF ACF/SNA Network Generation* for additional details.)

For example, if there are 1000 4K blocks in the system and the SNAKEY parameter ILWP4 is set to 35, there must be at least 350 4K blocks available for the system to respond to the VR pacing request. When setting these values, the input list shutdown values for the different block types should be the values you want to keep from reaching. Using the previous example of ILWP4 set to 35, if the input list shutdown value for 4K blocks is 200, the difference is 150 4K blocks. This should be enough 4K blocks to handle the next window's worth of PIUs. If the amount of remaining 4K blocks is not correct, the specified value for ILWP4 should be changed.

While a virtual route is "blocked" (waiting for a pacing response) the output message data is held in storage buffers that were associated with the application sending the message.

---

## Software IPL Considerations

In consideration of the following facts:

1. PIUs with a VR sequence number higher than the next 1 expected are discarded.
2. PIUs with a VR sequence number lower than the next 1 expected are discarded until the expected 1 arrives.
3. No traffic will flow inbound to TPF when a VRPRQ is lost by TPF. (TPF owes a VRPRS).
4. Messages sent by TPF outbound are either discarded or the route taken down if TPF exceeds the window size (TPF owes a VRPRQ).

TPF must take steps to guard against these occurrences and recover from them should the condition occur.

The VR sequence number problem is addressed by having TPF record 2 sets of VR sequence numbers. 1 is the VR sequence numbers that are being assigned by TPF outbound. The assigned sequence numbers are updated by the SNA output routine after the message is taken off the VR queue and is ready for transmission. The second is a recording of the VR sequence numbers at interrupt time (CE, DE) that the NCP has actually received. Following a software IPL, TPF sends out dummy messages that flow on the VR starting from the last 1 that we know NCP has received up to the 1 that TPF last assigned. The number of these dummy messages will be the largest number TPF can send in 1 I/O operation.

The problem of VR pacing is addressed by TPF setting VRPRS received during processing of the read interrupt. TPF uses existing logic that parses the PIUs to record the first 64 bytes in the in-core trace table to remember that a VRPRS was received. This closes the window where TPF could lose track of whether a VRPRS was received.

On output, TPF sets VRPRQ sent and resets the indicator in the write interrupt routine that records VR sequence numbers sent. This indicator is checked following a software IPL and if on, the window size is adjusted to indicate that a full window remains and that a VRPRQ should be set on the first PIU of the window.

If a VRPRQ was received but a VRPRS not sent by TPF, it is handled in a similar fashion and TPF sends out a VRPRS during restart. This is placed on the top of the VR queue and sent out as soon as TPF resumes communication with the NCP.

---

## Hardware IPL Considerations

If HARDREC=YES is coded in SNAKEY on restart, TPF attempts to resynchronize the VR (virtual route) sequence number for each channel-attached NCP. This is due to the VR sequence number not being keypointed before the hardware IPL. The NCP must be version 4 release 2 or above to support VR resynchronization.

When VR resynchronization is in progress, TPF continues to receive inbound PIUs and hold all outbound PIUs on VR0 in core blocks until VR Resync is completed. Meanwhile, TPF sends a few SNA commands on VR1 in an attempt to recover the next VR sequence number from the NCP. It is critical that the amount of inbound traffic should be minimized, because TPF must queue all outbound PIUs on VR0 in core blocks. The VR PIU Pool threshold in the gateway NCP is used to control the network traffic between TPF and NCP.

To alleviate TPF resource problems, TPF performs VR resynchronization serialization across all channel-attached NCPs. The SNAKEY parameter VRRT0 is used to specify the time-out value for VR resynchronization for each NCP. Upon timeout, TPF either declares that VR resynchronization is complete for the NCP if the response to DACTVR is outstanding, or TPF shuts down the NCP because of VR resynchronization failure.

The VR PIU Pool threshold should be set properly to prevent the NCP from depleting its buffer pool and to minimize TPF congestion from queuing core blocks.

The NCP generation parameter, VRPOOL, is the parameter for the VR PIU pool. See *NCP/SSP/EP Resource Definition Reference* for more information about VR PIU Pool:

Virtual route resynchronization does not address the problem of LU-LU session synchronization following a soft or hard IPL. For NEF type terminals connecting using NEF V2 or ALCI, no session resynchronization is necessary. This is because NEF does not check session sequence numbers, nor does it use brackets or change direction. For X.25 terminals, NPSI does not use brackets or change direction, but it does check session sequence numbers. Session resynchronization (CLEAR/SDT) is required to recover the NPSI LU\_1 sessions. For SNA-type terminals, because the session control block (RVT2) is not keypointed on a hard IPL, the LU session may be hung; that is, TPF status does not agree with LU status. TPF is aware that a session may exist. (LU control block, RVT1, is keypointed on a time-initiated basis, and LU information is preserved.) However, TPF is not aware of the status of the session; that is, in brackets, response pending, or next expected sequence number.

TPF does provide time-out utilities to detect hung sessions and attempt resynchronization based on LU type (CLEAR/SDT or CLEAR/STSN/SDT) which should recover most, though not all sessions; for example, LU6.2 sessions receiving unexpected sequence numbers are terminated.

If HARDREC=NO is coded in SNAKEY on restart, TPF issues a DISCONTACT request for each NCP using FID4, if the NCP shows active in the RVT entry. As a result of DISCONTACT, automatic network shutdown (ANS) takes place. The network has to be restarted. The DISCONTACT is issued because TPF cannot determine the correct VR sequence numbers, as the control block containing the sequence numbers (in the subarea address table) was not keypointed prior to the soft or hard IPL.

---

## Extended Network Addressing

TPF requires the extended network addressing feature available in NCP. This means that the network address contained in a FID4 header is a 1 byte subarea number and 2 byte element address.

As a result, the TPF network address table (NAT) has been expanded to allow for 3 byte network addresses. The Resource Vector Table now treats the RV1NASA field as the subarea and the RV1NA field (which used to be the 2 byte network address) as the element address.

In addition, TPF builds and accepts CDINIT format 3 requests with control vector X'1A', which contains the larger network addresses. TPF indicates this support on the ACTCDRM response to VTAM (bits 9, 12, 13, 16 and 18 of the CDRM usage field in control vector 6 must be set on). CDCINIT format 1 uses session key 15 (network qualified address pair). Session key 15 contains the 6 byte network addresses with the length field set to 12 to indicate no network identifiers.

---

## Network Definition

For SNA CTC, the IODEV macro is used in the System Initialization Process to define the addresses of the CTC connections. In SNAKEY, this IODEV macro produces the symbolic device address table (SDAT), which defines the size of the physical network, including the number of CTC connections. After you define the addresses of the CTC connections using the IODEV macro, define the NCP and CTC links to the TPF system using the OSTG program or the ZNDYN command. See “Defining SNA Resources to the TPF System” on page 193 for more information about defining these resources to the TPF system.

When you create a network definition for NCP or CTC resources, you must define the VR window size for VR pacing using the WINSIZE parameter. This number is saved in the MAXDT field in the RRT/RVT and passed to the adjacent node on the ACTVR. The recommended value is 100 to minimize the overhead of pacing but can be specified as high as 255. You are expected to tune this parameter to suit his network. Be aware that picking too high a number will impact NCP performance as NCP allocates 3 times this number of buffers for the virtual route.

The Subarea Address Table includes the ER and VR information.

TPF shares 1 network address for the SSCP and the CDRM (element address 0). With FID4 support (TPF as a data host), the SSCP is replaced by the PU Manager. TPF uses the same RVT entry for both the PU Manager and the CDRM, since no information on the PU5-PU4 session is kept in the RVT. ER and VR control information is kept in the Subarea Address Table.

## SNA Network Interconnection (SNI) Considerations

When operating as a PU 5 type node (SSCP), with TPF's support of the SNA network interconnection (SNI) facility of NCP and VTAM, there are some special considerations relative to network definition. A VTAM CMC must own all resources attached to a gateway (SNI) NCP. For more information on PU 5 nodes, see “Cross-Domain Resource Manager (CDRM)” on page 14.

A gateway NCP is defined from the viewpoint of the gateway SSCP that owns it. Therefore, to represent the TPF view of the network from within TPFs' own network, you must specify a subarea for the CDRM when the CDRM is defined to the TPF

system. See “Defining SNA Resources to the TPF System” on page 193 for more information about defining CDRM resources to the TPF system.

The name of the TPF systems’ network ID stored in the TPF systems’ SNA keypoint (CTK2) is the name as it is known by the owning VTAM system.

Because the design of SNI requires the VTAM CDRM that owns the gateway resources to appear in the subarea of the gateway NCP as an element in the NCPs subarea, some restrictions exist regarding TPF’s attachment to an SNI network. TPF’s table structure assumes that NCPs and VTAM hosts are each a unique subarea. Therefore, the subarea address table (SAT) has 1 entry for each subarea node. The VTAM SSCP and its associated CDRM are assumed to reside in the same subarea. With SNI, this is not true. To circumvent this problem, define a virtual or pseudo-subarea for the VTAM CDRM. The TPF system will create an SAT entry for this subarea and point it to the resource vector table entry for the VTAM CDRM. The RVT will contain the real network address for the CDRM. The actual network address of the VTAM CDRM is discovered dynamically when the CDRM-CDRM session is established. This virtual subarea will be used by TPF internally as the **owning** subarea for all resources that request sessions with TPF through this CDRM.

### SNI and CTC Considerations

A CTC connection to VTAM requires TPF to appear as a node in the VTAM network and use a CDRM-CDRM session or Control LU (CLU) - Logon Manager session to establish LU-LU sessions between the VTAM and TPF hosts.

SNA CTC support requires the channel-attached major node to reside in the same network (for example, VTAMNET) as VTAM. Because the TPF system must be connected to the NCP network through SNI, the TPF CDRM resides in a different network (for example, TPFNET). In order to establish a session across CTC, TPF must appear to VTAM as a CDRM in the VTAM network so an additional TPF CDRM must be defined. See the *TPF ACF/SNA Network Generation* for examples of SNA CTC configurations.

Therefore, if both a CTC connection and an SNI connection exist, VTAM must have 4 separate definitions for the TPF CDRM:

- 1 definition for the SNI connection
- 1 definition for the CTC connection
- 2 VTAM CDRM definitions for TPF.

### SNI and APPN Considerations

When the TPF system is connected to a VTAM system using both PU 5 NCP links across SNI and APPN links, an alias name for the CDRM to the VTAM system must be defined to the TPF system. In this environment, the VTAM system is defined as an *interchange node*, allowing it to act as a PU 5 node and an APPN network node. VTAM requires that its SSCP name and CP name must be identical. The problem is that this name cannot be defined to the TPF system as both a CDRM and a CP. The CP must be defined to the TPF system using its real name and the CDRM must be defined using an alias name.

As an example, the SSCP and CP name for a VTAM system is VTM2. The alias name for the CDRM will be VTM2CDRM. The subarea of the VTAM system across the SNI boundary is 5. The following are the OSTG definitions needed to define this CP and CDRM:

```
VTM2      RSC      LUTYPE=REMCP
VTM2CDRM CDRM     SUBAREA=5,ELEMENT=1,REALNAME=VTM2
```



In the previous example, the real name of the VTAM system, VTM2, is defined to the TPF system as a remote APPN control point using the OSTG RSC statement. You can also use dynamic LU support to define VTM2 to the TPF system.

You must define the alias name, VTM2CDRM, to the TPF system as a CDRM using the OSTG CDRM statement. The REALNAME parameter on the CDRM statement must be the real name of the VTAM system. To display or deactivate the CDRM from the TPF operator console, you must use the alias name.

See “Defining SNA Resources to the TPF System” on page 193 for more information about defining SNA resources to the TPF system.





---

## PU 2.1 Considerations

See “Low-Entry Networking (PUT 2.1) Environment” on page 15 and *IBM Systems Network Architecture Advanced Peer-to-Peer Networking Architecture Reference* for additional information about PU 2.1.

---

### General Information

In a PU 2.1 network, the TPF system can be a LEN node or an APPN node. The TPF system can be channel-attached to devices like 37x5 NCP, 3174 APPN controller, RISC System/6000 system, and other PU 2.1 nodes.

The following information provides general information about the TPF system in a PU 2.1 environment. It also provides more specific information based on the TPF node type (LEN or APPN) and which PU 2.1 devices are connected to the TPF system.

### Session Control

The considerations for sessions in a PU 2.1 environment are as follows:

- The LENNETID parameter in the SNAKEY macro is used as the network identifier when the TPF system is connected to the network as a PU 2.1 node.
- An LU-LU session is associated with a session identifier (SID). The SID is assigned by the node sending the BIND command and is used to route session traffic between the two LUs. The SID provides the same function as a network address pair in a subarea network, but is dynamically assigned and reused when the session ends.
- A TPF PU 2.1 LU does not have a local address known to VTAM or NCP and, consequently, local addresses are not carried in the Transmission Header (TH). Local addresses have been replaced with a session identifier (SID) assigned when the Bind requesting the LU-LU session is sent. All PIUs that flow between a TPF PU 2.1 and NCP use the session identifier to identify the LU-LU session.
- Each session is uniquely identified by a Procedure Correlation Identifier (PCID). The PCID is assigned at session activation, by the node sending the BIND, and identifies the session for its duration. The PCID provides a common identifier that can be used by the operator, of any node, to display or request information about a particular LU-LU session.
- The PCID is the only unique identifier for a PU 2.1 LU-LU session throughout the entire network. PU 5 LU-LU sessions are uniquely identified by the network address pair of the two LUs. If the route for a PU 2.1 LU-LU session traverses multiple hops, the SID cannot be used to identify the session throughout the network because the SID value is unique only between two adjacent nodes.
- LU names, as carried in the BIND, are optionally prefixed with the network identification (NETID) and called network qualified names (NQN). Qualifying an LU name with its NETID guarantees the name is unique across networks and allows cross-network communication without the need to use aliases.
- An adaptive pacing technique has been designed which allows greater flexibility and prevents deadlock. The use of this technique replaces current window sizes and prevents the possibility of deadlock for LU-LU sessions.

### Session Identifiers

The routing of message traffic between the TPF system, as a PU 2.1 node, and NCP (or other channel-attached PU 2.1 device) is based on session identifiers

(SIDs), as contrasted with network addresses used for subarea routing when TPF is a PU 5 node. The SID uniquely identifies a session between a TPF LU and another LU and is associated with a link. A SID is a 17 bit number with the following structure:

- 1 bit for an origin destination assignment indicator (ODAI). The ODAI specifies which node, TPF or NCP, assigned the SID.
- 16 bits for an index, where the lowest index is X'0101' and X'FEFF' the highest; for a total of 65,022 indexes.

Which node, TPF or NCP, assigned the SID is denoted by the ODAI. TPF always assigns an ODAI value of zero (0), and the NCP assigns an ODAI value of one (1). How a node, TPF or NCP, sets the ODAI value is determined at XID. That is, the primary side of the link uses the ODAI value zero (0), and the secondary side uses one (1). Since TPF is considered the primary side of the channel, it always uses zero (0) as its ODAI value.

SIDs are assigned by the BIND sender from a pool of available numbers. Assignment starts from the lowest numbers available number (X'0101') and proceeds to the higher numbers. When a session ends, its SID is available for re-assignment. This procedure ensures that the address space is not fragmented and limits the main storage the TPF system needs to correlate SIDs to session control blocks (RVTs or SCBs).

## Extended BIND

The SNA architecture requires that a TPF system as a PU 2.1 node supports extended BINDs, which implies support of the following:

- Negotiable and non-negotiable BINDs
- Control vectors in the BIND
- Adaptive session level pacing
- Network qualified LU names (NQNs).

A negotiable BIND allows the PLU, SLU, and NCP boundary function (BF) to negotiate the BIND parameters to be used for the session. The SLU and its boundary function can change the BIND parameters it disagrees with. The changed parameters are then reviewed by the PLU. If the PLU agrees, then normal data traffic can begin. If the PLU disagrees, it must UNBIND the session.

For certain LU-LU session types, the TPF system uses negotiable BINDs to allow the NCP BF and VTAM to negotiate the maximum request unit (MAXRU) sizes and pacing values for the session. If a BIND receiver changes any of the BIND parameters whose negotiated value is unacceptable to the PLU, the session is unbound.

The TPF system supports negotiable and nonnegotiable BINDs that contain the following control vectors:

- **PCID**—The PCID uniquely identifies a session. It contains the following:
  - Network identifier (NETID) of the control point (CP)
  - CP name of the node that initiated the session
  - 8-byte number to make the PCID unique among all sessions started by the given CP.

The inclusion of NETID and CP name in a PCID eliminates the current gateway problem of requiring an alias PCID each time a session crosses a network

boundary. The fully qualified PCID, therefore, uniquely identifies an LU-LU session regardless of the number of networks it passes through. The PCID is primarily intended for use by network management applications as a means to collect and report on specific LU-LU sessions.

- **Mode name**—The mode name used for a session defines properties of the connection, such as virtual route number and priority.
- **Class of service / transmission priority field (COS/TPF)**—The COS/TPF specifies the transmission priority through the network of data traffic flowing on the session. This value should not be confused with virtual route priority, which is the transmission priority through the subarea network.
- **Route selection**—The route selection defines the route a session uses through the network.

The TPF system will always send an extended BIND for an LU-LU session over a PU 2.1 link. The BIND response received by the TPF system may or may not be extended.

## SNA Restart and ALS Discovery

During SNA restart, the TPF system verifies that ALS connections that were active at the time of the IPL are still active. In addition, the TPF system discovers new ALS connections that have become active and automatically creates new resource definitions for these ALS connections, if necessary.

The restart portion consists of identifying ALS connections that were active when TPF went down, and verifying their status by issuing a non-activation XID.

The discovery portion is identifying all ALSs that have become active since the TPF system went down. This function is only for ALSs (and PU 5 CTC links); PU 5 NCPs become active via the ZNETW activate command.

An overview of the SNA restart logic for ALSs is as follows:

1. Scan the RVT entries to find ALS connections that were active before the IPL. For each active ALS found, issue a non-activation XID and rebuild the session index table (SIT) entry for that ALS.
2. Scan the symbolic device address table (SDAT) in keypoint 2 (CTK2) to find SDAs that are not currently assigned to active ALS connections. For each such SDA found, issue a pre-negotiation XID to check if the ALS is ready to become active. (This is known as *self discovery* of an ALS.)

Before issuing the pre-negotiation XID, a sense is issued to determine if the SDA is connected to a PU 5 NCP. If it is, a pre-negotiation XID is not issued because this is not a PU 2.1 device.

When an XID is sent by the TPF system, the information provided must include the fully qualified control point (CP) name of the TPF system. The network identifier portion of the fully qualified CP name is the value of the LENNETID parameter on the SNAKEY macro in keypoint 2 (CTK2). The name portion of the fully qualified CP name is the name of the application that was defined to the TPF system as the local CP. This is done by coding ASNA=LOCP on the MSGRTA statement that defines the application to the TPF system during the SIP process.

Because an ALS becomes active dynamically, there is no way of knowing which ALS links will be active when the TPF system comes up; therefore, it is not possible to allocate SNA CCW areas at OSTG time. Instead, the TPF system allocates a CCW area as follows:

- When the ZNETW activate command is issued for a PU 5 NCP.
- When an ALS becomes active during SNA restart or an attention interrupt is received from an ALS to initiate the link activation process.

---

## 37x5 Considerations

Define the ALS name of the 37x5 channel adapter to the TPF system. The name of the ALS resource in the TPF resource definition must be identical to the name on the PU statement in the NCP generation deck.

See “Defining SNA Resources to the TPF System” on page 193 for more information about defining ALS resources to the TPF system.

The following items must be specified in an NCP gen to define a channel-attached TPF system as a PU 2.1 node:

- The Auto Network Shutdown keyword should be coded as ANS=CONT on each PU statement in order that a CMC failure will not affect LU sessions with TPF.
- On the LINE statement defining a channel-adapter, an Attention Timeout value greater than the time necessary for TPF to detect and recover from an error condition should be coded. Typically, this is one minute for a software-initiated IPL of TPF.
- For NCP Version 5.3 or higher, on the LINE statement defining a channel adapter, the MONLINK parameter, should either be omitted or should be coded as 'MONLINK=CONTINUOUS'. If it is omitted, then the default value of 'CONTINUOUS' is assumed for the channel adapter.
- The NCP gen must be modified to start sessions with AX.25 LUs using NCP Version 5.3 or higher. The AX.25 LU definition generated by the NCP/EP Definition Facility (NDF) for the NPSI virtual circuit must have VPACING set to zero.
- When defining the independent LU for a TPF application, ensure that the logon mode (DLOGMODE) contains the same pacing value defined for other network LUs that log on to the TPF application. If the pacing value of the independent LU for the TPF application is less than the pacing value of the other LUs in the network, an error occurs during session startup; a sense 08210006 is received in response to the bind.

**Note:** This error occurs only for LU-LU sessions that use non-negotiable BINDs.

- A remote PS/2 can send a BIND request if it is defined as a PU T2.1 node with an independent LU. Specify XID=YES in the PU statement of the NCP generation to make the PS/2 a PU T2.1 node, and specify LOCADDR=0 in the LU statement to make it an independent LU.

**Note:** When the TPF SLU resources are defined to VTAM, VTAM will direct all sessions for those resources through that NCP. Dynamic Reconfiguration allows the user to move TPF SLU resources to another NCP channel attached to TPF.

## 37x5 and APPN Considerations

You must define the control point (CP) name of the VTAM system that owns the 37x5 to the TPF system. If you have backup VTAM systems capable of owning the 37x5, define to the TPF system the CP names of these VTAM systems using the LUTYPE=REMCP option in the OSTG RSC statement or using dynamic LU support. The CP name in the TPF resource definition must be identical to the value of the SSCPNAME start option in the VTAM system.

See *TPF ACF/SNA Network Generation* for more information about the OSTG RSC statement. See “Using Dynamic LU Support to Define SNA Resources” on page 199 for more information about dynamic LU support.

When the TPF system is operating in APPN mode, you do not need to predefine TPF resources (LUs) to the NCP or to VTAM. Instead, the TPF system registers its resources with the APPN network when you bring up the network.

## 37x5 and LEN Considerations

When the TPF system is operating in LEN mode, all TPF SLU resources (TPF SLU threads, FMMR, and CLU resources) must be defined in the NCP generation deck of the channel-attached NCP. The NCP generation deck is used by VTAM to define TPF SLU resources. For an example of the definitions of SLU resources, see Figure 92 on page 242.

When the TPF system is operating in LEN mode, you must define all TPF PLU resources (TPF PLU applications) in the NCP generation deck of the channel-attached NCP. These names must have a 1-character CPU ID suffix of the TPF host with which the names are associated and the 2-byte hexadecimal number (X'00'–X'FF') associated with this ALS. This number must match the ALSQN parameter on the ALS macro in OSTG because of the VTAM restriction that all PLU names must be defined before they can issue a BIND request. VTAM does not support dynamic discovery of independent PLUs. The reason for the suffixing is that there is no multiple peripheral node support in VTAM or NCP; to make the PLU name unique per host per ALS, the 4-character PLU name must have a suffix. Figure 92 on page 242 contains sample alias names for PLU resources. Resource GMT1B30 is an example of how TPF application GMT1 is defined with suffix B30, where B is the CPUID and 30 is the ALSQN.

### Sample NCP Definitions with TPF Connected as a LEN Node

See Figure 92 on page 242 for a sample definition of an NCP generation deck with the TPF system channel-attached as a PU type 2.1 LEN node.

**Note:** This sample definition shows only the specified keywords that are required to define a PU type 2.1 LEN node. User-supplied keywords are not included.

```

*****
*      BUILD MACRO FOR N30H521      *
*****
N30H521  BUILD MODEL=3745,          *
        SUBAREA=511,                *
        TYPGEN=NCP,                  *
        BFRS=80,                     *
        MAXSESS=1000,                *
        NEWNAME=N30H521,             *
        PUNAME=N30H521,              *
        NETID=VTAMNET,                *
        SALIMIT=1023,                 *
        SLODOWN=12                    *
*
*****
*      LINE FOR REMOTE PS/2 FOR LU6.2      *
*****
*
L3027S  LINE ADDRESS=27,            *
        CLOCKNG=EXT,                 *
        SPEED=9600,                   *
        DUPLEX=FULL,                  *
        ISTATUS=INACTIVE,              *
        SSCPFM=USSSCS,                 *
        MODETAB=LU62MOD1,              *
        DLOGMOD=TPFLU62                *
*
        SERVICE ORDER=(P3027A)
P3027A  PU ADDR=C1,                  *
        PUTYPE=2,                      *
        ANS=STOP,                      *
        MAXDATA=265,                   *
        MAXOUT=7,                      *
        PASSLIM=8,                     *
        PACING=7,                      *
        XID=YES,                       *
        RETRIES=(,1,4)                 *
*
T3027A01 LU LOCADDR=0,ISTATUS=ACTIVE,RESSCB=10
*
*****
*      CHANNEL DEFINITION FOR TPF PROCESSOR B PU2.1      *
*****
L30CA4B  LINE ADDRESS=(11),          *
        CASDL=420,                     *
        TRANSFR=52,                     *
        DELAY=0,                        *
        INBFRS=4,                       *
        ISTATUS=INACTIVE,                *
        TIMEOUT=120                      *
*
P30CA4B  PU PUTYPE=2,                  *
        ANS=CONT                          *
*
CLUBB004 LU LOCADDR=0,RESSCB=1,ISTATUS=ACTIVE
GMT1B30  LU LOCADDR=0,RESSCB=100,ISTATUS=ACTIVE,DLOGMOD=TPF3270
NEFB30  LU LOCADDR=0,RESSCB=100,ISTATUS=ACTIVE
ZZZB004 LU LOCADDR=0,RESSCB=1,ISTATUS=ACTIVE
ZZZB006 LU LOCADDR=0,RESSCB=1,ISTATUS=ACTIVE
PPPB001 LU LOCADDR=0,RESSCB=1,ISTATUS=ACTIVE,          *
        MODETAB=LU62MOD1,DLOGMOD=TPFLU62
PPPB002 LU LOCADDR=0,RESSCB=1,ISTATUS=ACTIVE,          *
        MODETAB=LU62MOD1,DLOGMOD=TPFLU62
*

```

Figure 92. NCP Gen with TPF Channel-Attached as a PU Type 2.1 LEN Node

## VTAM and Logon Manager Considerations

The VTAM logon manager application is used when the TPF system is operating in LEN mode. When you migrate the TPF system to APPN mode, the VTAM logon manager application is no longer used; therefore, if the TPF system is operating in APPN mode, the following information does not apply.

- When TPF is a PU type 2.1 LEN node:
  - TPF will no longer perform application authorization for SNA resources. The VTAM session management exit must be used to ensure the terminal is permitted to LOGON to the TPF application.
  - VTAM USS definition tables should be used to transform the TPF-oriented LOGI requests to a VTAM equivalent (LOGON APPLID).
  - All TPF applications LUs (4-character name) must be defined to VTAM as APPLs in a VTAM application deck. These definitions are required by the Logon Manager.
  - TPF applications should be predefined to Logon Manager, if possible; however, Logon Manager does have the ability to dynamically discover application names. For a description and examples of how resources are defined to Logon Manager, see the *VTAM Network Implementation Guide*.
- When TPF is a PU 5 node across a CTC connection:
  - Use the VTAM Logon Manager for initiating sessions between VTAM host applications or VTAM local terminals and TPF applications. This allows for load balancing across multiple CTC connections.
  - Define to VTAM the TPF applications to be accessed across CTC connections using the Logon Manager. Specify these applications as APPLs using the 4-character generic name. The alias names (the application names suffixed by the TPF CPU ID and CTC qualifier number) must be defined to VTAM as CDRSCs associated with the TPF CTC CDRM in the VTAM network.
  - In most cases, predefine to the Logon Manager the TPF applications to be accessed across CTC connections. See the *VTAM Network Implementation Guide* for specific details of when this is required. When predefinition is not required, the Logon Manager does have the ability to dynamically discover the TPF application names.
- When TPF is a PU 5 node attached to a gateway NCP and also a PU 2.1 node or PU 5 node across a CTC connection:
  - The terminal operator attached to a gateway NCP will attempt to logon to a TPF application. In this case, the use of interpret tables and aliasing eliminates the problem of duplicate names. Users on an NCP subarea connection can LOGON RES0 and have that translated by VTAM to LOGON RES1, which is a TPF PLU defined to VTAM as belonging to the TPF CDRM. Through the use of gateway aliasing, the name RES1 can be translated back to RES0 and the owning SSCP (TPFx) determined. The CDINIT would be sent on the CDRM-CDRM session between VTAM and TPFx.

Users on a PU 2.1 or PU 5 CTC connection can LOGON RES0 and have RES0 as an application controlled by the Logon Manager. The Logon Manager then forwards the SESINIT to the TPF CLU on behalf of the SLU for the TPF PLU RES0. TPF suffixes the PLU name in the BIND to make it unique within the network.



---

## Non-37x5 Considerations

The TPF system connects to PU 2.1 devices other than 37x5 devices. Some of these devices, like a 3174 APPN controller or RISC System/6000 system, do not provide their ALS name in the XID during link activation. The TPF system creates an ALS name using the following format:

TPF $xnnnn$

Where:

$x$  is the 1-character CPU ID of the TPF processor.

$nnnn$

is the 4-character SDA number of the ALS channel adapter.

You must define the ALS names to the TPF system using the TPF $xnnnn$  format described previously. See “Defining SNA Resources to the TPF System” on page 193 for more information about defining ALS resources to the TPF system.

If a new symbolic device address (SDA) is required for a non-37x5 PU 2.1 device, you must add an IODEV macro with the parameter DVTYP=37x5 to the SIP deck. See the *TPF System Generation* for more information about the IODEV macro.

A VTAM system is not required for non-37x5 PU 2.1 devices that are channel-attached to the TPF system; however, consider a VTAM system for network management purposes.

## Non-37x5 and APPN Considerations

You must define the control point (CP) name of the channel-attached PU 2.1 node to the TPF system using the LUTYPE=REMCP option in the OSTG RSC statement or using dynamic LU support.

See *TPF ACF/SNA Network Generation* for more information about the OSTG RSC statement. See “Using Dynamic LU Support to Define SNA Resources” on page 199 for more information about dynamic LU support.

A VTAM system or other APPN network node that is a dependent LU server is required if there are dependent LUs attached to the non-37x5 device, and this device is only a dependent LU requester (not a server).

## Non-37x5 and LEN Considerations

When the TPF system is connected to non-37x5 devices as a LEN node, only LU 6.2 sessions are supported and these sessions must be started by the remote LU, not by the TPF system. The remote LU 6.2 devices must, therefore, be independent LUs.

## 3174 Considerations

- The TPF system supports as many as 255 LU 6.2 sessions per controller and all sessions are started by PS/2 workstations, or other intelligent workstations, on the token ring that is managed by the 3174 controller.
- The network identifier defined in the 3174 APPN configuration must match the LENNETID parameter in the SNAKEY macro.



- It is recommended that you use the wildcard option in the 3174 APPN configuration so that BIND requests are sent to the TPF system when the 3174 does not know the target resource.
- Define the mode names used for LU 6.2 sessions in the 3174 APPN configuration.
- Define the link in the 3174 APPN configuration as supporting both PU 2.0 and PU 2.1 traffic.
- For 3174 connections, SIDs X'0101' to X'01FF' are reserved by the 3174. To compensate for this, you must add an additional 255 SIDs when specifying the MAXSID parameter on the SNAKEY macro. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro and MAXSID parameter.



---

## SNA Message Protocol

Messages transmitted between TPF and a logical unit must adhere to SNA message protocol. In an SNA environment, messages are prefixed with a request-response header (RH). Within this header is a field that indicates to the receiver of the message the type of response the sender requires. SNA defines three response types:

**Definite response:**

Directs the receiver to respond without regard to the nature of the response. A response, either positive or negative, must be returned. A positive response is referred to as a +RSP. A negative response is referred to as a —RSP.

**Exception response:**

Directs the receiver to respond only if the request is unacceptable as received or cannot be processed. Only a negative response is returned.

**No response:** Indicates to the receiver that no response is to be returned.

---

## Response Protocol

The response for logical units sending messages to TPF depends upon the LU device type. Input messages received from logical units attached to 3601, 3602, 3274 or 3276 cluster controllers must request exception response (-RSP, that is, negative response only). Otherwise, the data returned in response to a properly sent request serves as request confirmation. Messages received in error from LUs attached to 3274 or 3276 controllers result in TPF sending a Clear/Start Data Traffic (SDT) SNA command sequence. TPF also informs the operator that an error has occurred.

Input messages received from logical units attached to a 3271 control unit request “no response.” When an error is detected, TPF discards the input message and sends a CLEAR command. The terminal operator must press the RESET key, which is located on the keyboard, and reenter the message.

---

## Recoverable and Non-Recoverable Messages

Recoverable input messages may be defined as saved messages. The sender of a message saves the message until a response is received. This practice enables the sender to retransmit the message should a malfunction occur. Non-recoverable messages are not saved; a malfunction results in the loss of the message.

TPF requests **definite response** when it sends a message defined as recoverable; “exception response” when it sends a message defined as non-recoverable. TPF saves recoverable messages until a positive response is received.

## Single-Segment Messages

Single-segment messages are messages that take only one I/O transmission. TPF-initiated single-segment messages are sent to all logical unit types except for: 1) messages sent to 3270 SDLC terminals attached to a 3271 control unit, and 2) batch logical units. In the following single-segment illustration, arrows indicate the direction of data transmission.

```

LU                TPF

Input  ----- 1 ----->

      <----- 1 -----      (Recoverable reply)
Response
      -----+RSP(1)----->

Input
      ----- 2 ----->
      <----- 2 -----      (Non-Recoverable Reply)
      (-RSP or none)

```

## Chained and Segmented Messages

Chained messages are messages connected via forward and/or backward pointers. Functional associations may or may not exist between chained messages. For example, output messages to the same device are often chained together pending delivery. Here, the only association is the device to which the messages are sent. Most often, however, chaining is used to mean segmenting. A segmented message is one that exceeds the buffer-size limitation of a particular link. Thus, the message is broken into segments. Segments may be viewed as chained messages that have a functional association. Each chained message is identified with either a first-in-chain, middle-in-chain, or last-in-chain indicator. TPF assembles a chained and/or segmented message before presenting it to an application. The example below illustrates a segmented (chained) message transmission:

<i>LU</i>	<i>TPF</i>	<i>Comments</i>
Input Message		
Segment 1	----->	First-in-chain
Segment 2	----->	Middle-in-chain
Segment 3	----->	Last-in-chain
	<----- Reply	

TPF output messages are:

- Chained when the message is larger than the buffer of the receiving logical unit
- Sent in segments of 337 characters or less for LUs attached to 3271 control units

An SNA technique called "session pacing" controls the flow of messages throughout a network by limiting the introduction of new traffic to the rate at which it can be accepted. TPF supports both adaptive pacing and non-adaptive (fixed) pacing. Pacing window size is passed to TPF by VTAM in CDCINIT and SESINIT. TPF performs adaptive pacing on all PU 2.1 sessions and fixed pacing on all PU 5 sessions. AX.25/NEF sessions, however, are not paced.

TPF sends a number of message segments and waits for a pacing response from the NCP. The pacing response from the NCP resets the window size which allows TPF to send more segments. TPF responds to NCP's pacing requests by sending out pacing responses. These pacing responses allow the NCP to send TPF more segments.

```

LU                TPF

      <----- Segment 1

```

```

<----- Segment 2      (Request pacing response)
Cluster controller
pacing response
----->
<----- Segment 3      (Last-in-chain)
LU sends
response
----+RSP----->

```

In this example, the output message is recoverable. Thus, a definite response (+RSP) is requested for the last segment.

TPF processing for a chained output message, received in error, is as follows:

<i>LU</i>	<i>TPF</i>	<i>Comments</i>
<----- Segment 1		
<----- Segment 2		(Request pacing response)
Pacing Response		
----->		
<----- Segment 3		
<----- Segment 4		
LU sends negative response		
----- -RSP ----->		(Sequence number error)
<----- Clear Command		
<----- STSN Command		(TPF resynchronizes sequence numbers)
LU responds with		
----- +RSP ----->		(Text of the response contains the sequence number of the last successfully received message)
<----- SDT command		
<----- Segment 1		(Message is resent beginning with the first segment)
<----- Segment 2		
Pacing response		
----->		
<----- Segment 3		
<----- Segment 4		
----- +RSP ----->		(Last message was successfully received)

Here, a segment is received with an invalid SNA sequence number. The logical unit returns a negative response. TPF then initiates a message recovery/resynchronization dialogue. This determines the last completed message. TPF then transmits the message a second time. For a non-recoverable message received in error, TPF resynchronizes the sequence numbers but does not resend the message.

Large output messages to logical units attached to 3271 control units require special processing. The 3271 cannot process chained or segmented messages.

Processing of single block messages is unchanged. After the application issues the ROUTC macro, the output message transmission (OMT) program reblocks the message and issues a SOUTC macro. See *TPF General Macros* for more information about the ROUTC macro. See *TPF System Macros* for more information about the SOUTC macro.

---

## Multithread Processing

Multithread processing is a TPF option. It is intended primarily for logical units that support financial service terminals (Models 3606 and 3608). However, it is available for all 3600-type logical units. In multithread processing, either side can send the next message before it receives a response from the previous message. The multithread option is specified during generation of the network. However, TPF does not support 3600 multithread processing concurrent with message recovery. Message recovery is also specified during generation of the network. The two features are mutually exclusive because of constraints in the routing control parameter list (RCPL) introduced by 3-byte RID support.

---

## Bracket Support

Bracket support is provided primarily for use with devices attached to the 3274/3276 cluster controllers. Brackets identify the beginning and end of a conversation between a logical unit and a host application. A bracketed conversation continues, uninterrupted, until the host application ends it with an “end bracket” indicator. The logical unit initiates the conversation with a begin bracket indicator in the input message. Bracket support is not available for:

- Multithread logical units
- Logical units attached to 3271 control units
- IBM 3614/3624 Consumer Transaction Facility terminals

<i>LU</i>	<i>TPF</i>	<i>Comments</i>
Message 1 -----BB----->		Begin bracket = BB
<-----	Reply	
Message 2 ----->		
<-----EB-----	Reply	End bracket = EB

---

## Unsolicited Messages

An unsolicited message is an output message not associated with an input message. An unsolicited message results from a host application sending an unsolicited message. TPF SNA supports a ZSNDU interface only for SNA resources in session with a non-SNA application (that is, addressing via LNIATA/LEID rather than RID).

### Unsolicited Messages Destined for a LNIATA/LEID

For an unsolicited message destined for a LNIATA/LEID, TPF notifies the terminal of the pending unsolicited message. The operator must request delivery of the message. Unsolicited messages are non-recoverable.

### Unsolicited Messages Destined for a RID

An unsolicited message destined for a RID is treated as a normal data message. The operator **cannot** request delivery of the message. Unsolicited messages are non-recoverable.

---

## 3614/3624 Message Processing

The 3614 and 3624 Consumer Transaction Facilities are unattended self-service banking terminals. Both issue money, accept deposits, and perform various banking transactions. These units can be attached to TPF via a 37x5 communications controller or “loop” attached via a finance communication controller. Loop attachment refers to an intermediate connection (in this case, the finance controller) in the communications path. TPF cannot distinguish direct attachments from loop attachments. Therefore, applications residing in the finance controller must present a directly attached appearance.

The protocol for a 3614/3624 logical unit is unique. These logical units do not request “exception response” as do other units. Prior SNA terminology defined the 3614/3624 response as “exception reached recovery node (RRN).” Today, however, this response is generally accepted to mean a positive response (+RSP). Input messages from and output messages to a 3614 or 3624 may not be chained.

Communication between a 3614/3624 and the host typically consists of three messages. Such a sequence ends with the 3614/3624 status being sent to the host application.

<i>3614 or 3624</i>	<i>TPF</i>
Transaction Request ----->	
	<----- Transaction Reply
Transaction Status ----->	
	<---+RRN---

3614/3624 logical units are identified during network definition. These units must be defined as multi-thread, non-recoverable. Bracket protocol is not supported for 3614/3624 logical units, nor are these units permitted to receive unsolicited messages.

## 3614/3624 Session Initiation and Application Considerations

TPF and 3614/3624 logical units must establish a session before data can be transmitted. Prior to session initiation, the 3614/3624 must be loaded with the “configuration image.” This is a component of the IBM 3600 finance communication system. The configuration image is a combination of: 1) formatted configuration data, and 2) selected modules of financial controller data. When loaded, the configuration image defines the operations of the 3614/3624. Once the 3614/3624 is loaded, a session can be initiated.

Host applications use the ROUTC macro to send data to the 3614/3624. TPF does not examine output data; the data content and format is not validated. Data transmitted between a 3614/3624 and a TPF application are encrypted. The CIFRC macro is used to encrypt and/or decrypt data. TPF applications issue these 3614/3624 commands:

Change key -	Changes the encryption key
Close -	Closes the 3614/3624
Inquiry -	Retrieves the stored status of 3614/3624
Open -	Opens the 3614/3624
Request load -	Loads a new 3614/3624 configuration image
Request recovery -	Retrieves the current transaction status
Set key -	Sets a new encryption key

See the *3600 Finance Communication System 3614 Programmer's Guide and Reference* and *3600 Finance Communication System 3624 Programmer's Guide* for additional information.



## User Routines

TPF provides a set of replaceable exit modules to allow users to adapt their system to their specific requirements. The use of an exit or replacement of a TPF supplied module is optional and is provided to allow the user to amend or supplement the standard TPF support. Every reasonable attempt will be made to keep the interfaces to the routines the same in future TPF releases; however, future requirements may dictate changes. The exit modules operate as extensions to the TPF operating system and are part of a continuing effort to provide a clear separation of operating system and application functions.

## Process Selection Vector (PSV)

TPF provides optional exits, called Process Selection Vectors (PSVs), to allow the user to extend the TPF communication support. These exits can be used to add support for additional terminal types without modifying the user's application programs. These optional exit programs operate as an extension to the TPF system; the exits are started when a message is received or sent. PSV exits are started in the main I-stream.

The PSV exits are used to convert from the various X.25 protocols to the TPF protocol and visa versa. These exits may also be used for conversion to and from 3270 and 3600 protocols. In addition to protocol conversion, users now have the ability to supply their own form of message recovery support. The TPF Message Recovery support (the System Recovery Table and the safe store of the message) is bypassed when using these exits for message processing. The user may also provide SNA-architected Data Flow Control (DFC) layer function with PSVs. For additional information on the DFC layer, see "Data Flow Control (DFC) Considerations" on page 260. For a high-level summary of PSVs, see Appendix F, "TPF Message Processing Flow User Extensions" on page 317.

## Input Message PSV Processing

Figure 93 shows the principal that the user-written exit routine can transform the terminal protocol into the standard TPF Application Program Interface.

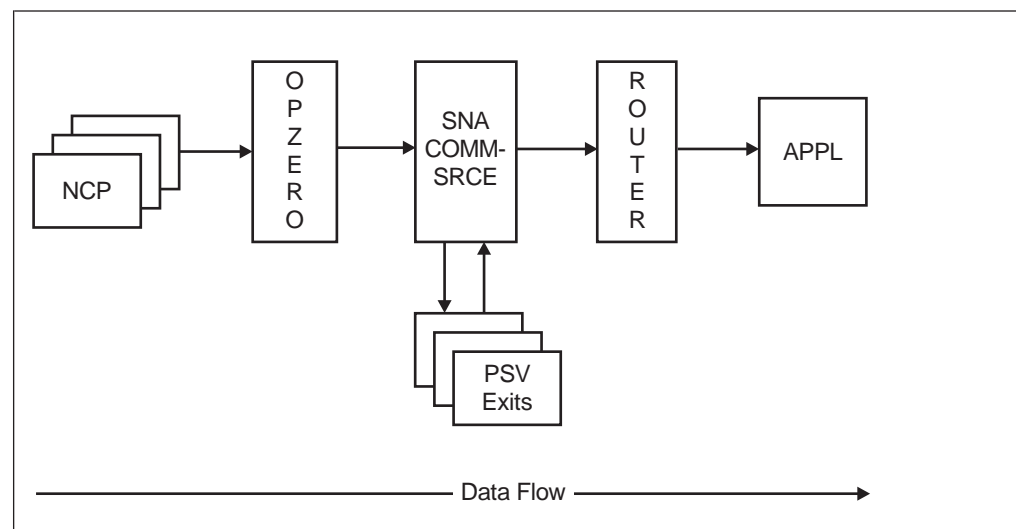


Figure 93. Input Message Processing

The requirement for a user exit routine to be started for a specific LU or terminal is specified at network definition. For SNA terminals (LUs), exits are defined via the Offline SNA Table Generation (OSTG) package. By coding a 6-character Process Selection Vector (PSV) name, each PSV name is associated with an ECB-controlled program that receives control when a message is received.

TPF users have the opportunity to define a maximum of 96 unique PSV names that can be correlated to a maximum of 96 user exit routines. Because TPF passes the PSV name to the ECB-controlled user exit program, each user-supplied program can uniquely service a given message (by PSV name) or multiple message types (using the name to identify the type of service required). The type of pre-processing message services include the following categories:

- Protocol Conversion - For example:
  - Data Transformation - Converting from 3270 to ALC format.
  - Data Compression and Compaction
  - Translation - Converting the character set used by the terminal to EBCDIC.
  - Address Transformation - Converting the address used by the network to the internal address used by TPF.
- Processing X.25 commands - Accepting a Call Request or Clear Request.
- Causing TPF to start the Log Processor to either logon or logoff a terminal.
- Causing TPF to start the Unsolicited Message Package with a request for an unsolicited message.
- Authorization and Decryption - Normally considered application functions, these may be handled in the exit.
- Provide inbound DFC layer.

## Output Message PSV Processing

Figure 94 on page 255 illustrates the user-written exit program to transform the protocol used by the application to the protocol required by the terminal.

Identifying which specific exit program to start is part of the network definition procedure. A PSV name can be defined for each LU that requires customized output message handling. Optionally, the user can specify that messages routed to a terminal addressed by line, interchange and terminal (LIT) or Logical End-Point Identifier (LEID) should be intercepted and passed to a PSV routine. See the *TPF System Installation Support Reference* for additional information about user exits.

The general function categories that an exit can provide are:

- Protocol Conversion - This is identical to the Protocol Conversion information previously detailed in the “Input Message PSV Processing” on page 253.
- Connection Establishment - This includes sending a Call Request command to NPSI to request a virtual circuit.
- Encryption - Normally considered an application function, this may be handled in the exit.
- Message Queueing - This is normally handled by the system in OMT or LMT, but it may be handled in the exit. See “Queue Manager” on page 269 for more details.
- Provide output DFC layer support. For additional information, see “Data Flow Control (DFC) Considerations” on page 260.

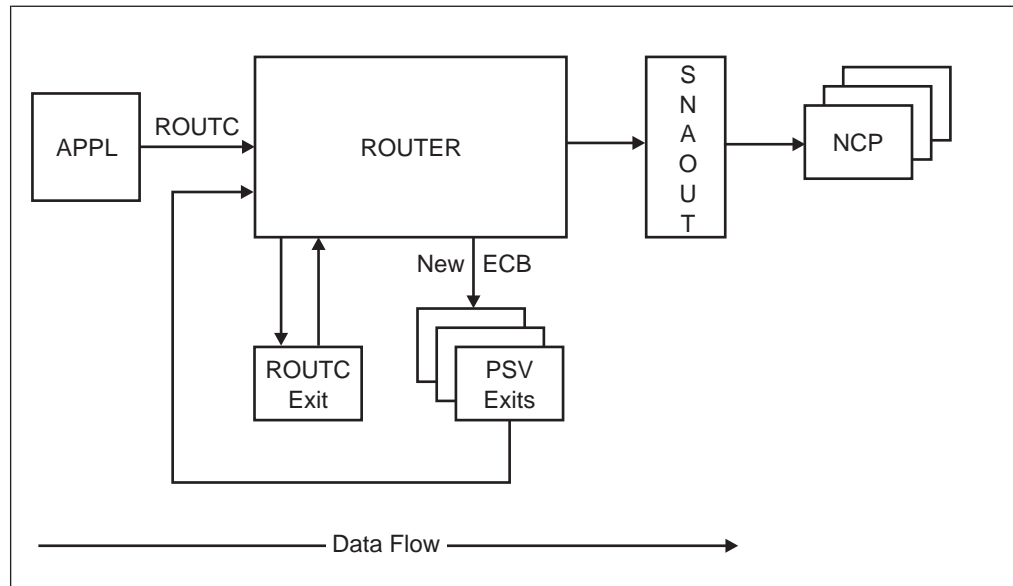


Figure 94. Extended TPF Outbound Message Flow, ROUTC and PSV Exits.

## PSV Interface

When a message is received from an LU, SNA Comm Source starts the user exit routine by using an ENTRC macro (Figure 93 on page 253). When a message is sent via the ROUTC macro, and the destination LU uses a PSV routine, the ROUTC macro code starts the associated PSV routine via control transfer (Figure 94). When the PSV exit routine is entered, the PSW storage protection key is set to the protect key of working storage. The information passed to the PSV exit routine includes:

<b>ECB Level 0</b>	The address of message block.
<b>EBW000-EBW0nn</b>	The RCPL.
<b>EBX012-EBX017</b>	The 6-character PSV name.
<b>EBX018</b>	The length of the LEID field (on entry from SNA Comm Source only).
<b>EBX019-EBX021</b>	The LEID defined for the LU (on entry from SNA Comm Source only).
<b>EBX022-EBX027</b>	The NETID of the remote LU, left-justified and padded with blanks (on entry from SNA Comm Source only).
<b>EBX028-EBX035</b>	The NAME of the remote LU, left-justified and padded with blanks (on entry from SNA Comm Source only).
<b>EBX036</b>	The device type of the remote LU as in NODEQ (on entry from SNA Comm Source only).
<b>EBCM01 bits 0-5</b>	Reserved for system use only.
<b>EBCM01 bit 6</b>	Input/Output indicator: 0 (Input message handling) 1 (Output message handling)

**EBCM01 bit 7**

Return/Exit indicator:

- 0 (caller expects return, issue BACKC)
- 1 (caller does not expect return, issue EXITC)

**EBCM02-EBCM04**

User data:

- If input, set to zeros
- If output, user information (see the ROUTC exit in the *TPF System Installation Support Reference*).

The previously described PSV interface included within the EBX fields and the equates for the EBC fields is contained in the IPSVI DSECT as shown in Table 13.

Table 13. IPSVI - PSV Interface Dsect

Label	Length (In Bytes)	Description
IPSINAME	6	PSV name
IPSIPNTX	1	Index to PSV name table
IPSILEIL	1	LEID length
IPSILEID	3	LEID
IPSINETI	8	Network Id of remote resource
IPSINAUN	8	Name of remote resource
IPSIDVTY	1	Device type of remote resource
EBCM01 Equates		<ul style="list-style-type: none"> <li>• IPSIEXIT X'80' discard and exit</li> <li>• IPSIINDX X'04' index is passed (COBT)</li> <li>• IPSIOUTP X'02' output processing</li> <li>• IPSINRET X'01' do not return to caller</li> </ul>

The exit routine passes the following information to SNA Comm Source:

**EBCM01 bit 0**

Continue/Exit indicator:

- 0 (continue processing input message)
- 1 (discard message, issue EXITC)

The user may activate PSV routines by entering segment COBT with the following interface:

**ECB Level 0**

The address of the message block.

**EBW000-EBW0nn**

The RCPL.

**EBX012-EBX017**

The 6-character PSV name (if EBCM01 bit 5 is off).

**EBX012**

The 1 byte PSV name index (if EBCM01 bit 5 is on).

**EBX018**

The length of the LEID field (0 if no LEID).

**EBX019-EBX021**

The LEID defined for the LU (irrelevant if length is 0).

**EBCM01 bits 0-4**

Reserved for system use only.

**EBCM01 bit 5**

PSV activation type indicator:

- 0 (EBX012-17 contains name of PSV to be activated)
- 1 (EBX012 contains the RVT name index of the PSV to be activated)

<b>EBCM01 bit 6</b>	Input/Output indicator: 0 (Input message handling) 1 (Output message handling)
<b>EBCM01 bit 7</b>	Return/Exit indicator: 0 (caller expects return, issue BACKC) 1 (caller does not expect return, issue EXITC)
<b>EBCM02-EBCM04</b>	Available for user use (passed to PSV).

If COBT cannot locate the requested PSV routine, system error CE9022 is issued and the ECB exited. In addition, if the destination in the RCPL is a RID, and a session exists, an UNBIND is scheduled.

### Input Considerations

The RCPL passed by SNA Comm Source to the exit routine contains:

- The RID of the origin LU.
- The 4-character application name to receive the message.

ECB area EBW048-EBW051 is reserved for system use only.

The exit routine can change any of the following information:

- Message Text - The message may be:
  - reformatted to remove protocol headers,
  - reformatted to remove an imbedded device address, or
  - translated to EBCDIC.
- The origin specified in the origin field of the RCPL may be changed to an LEID, an application name, or another RID.
- The destination specified in the RCPL may be changed to:
  - another TPF application,
  - the TPF Log Processor, or
  - the Unsolicited Message Package.

On return, the exit routine must:

- Pass the message block back on level 0.
- Return a valid RCPL in EBW000 of the ECB.

On return, the exit routine may optionally pass information to the application by:

- Placing data or file records on levels 4 through 15 of the ECB.
- Extending the RCPL to include control information in the RCPL General Data Area (GDA).

Comm Source receives control from the exit routine and:

- Sets the ECB flag to indicate either a SNA or non-SNA message.
- Sets ECB field EBROUT to:
  - X'000000' if RCPL Origin is an application name.
  - RID (right-justified) or LEID if RCPL Origin is a terminal.

This field is used by the System Error routines to return a message to the terminal if an error occurs in delivering or processing the message.

- Starts the trace and Data Collection routines to capture information for debugging and performance analysis.
- Delivers the message to the destination set in the RCPL.

## Output Considerations

The RCPL passed by ROUTC to the exit routine contains:

- the RID or LEID of the destination, and
- the 4-character origin application name.

The exit routine can change any of the following information:

- Message Text may be changed to:
  - Add protocol headers
  - Imbed device addressing
  - Compress or compact the data stream
  - Translate the character stream
  - Reformat the message
- The destination field of the RCPL may be changed to an:
  - LEID - to direct the output to another terminal
  - RID - to send the output to an SNA resource
  - Application name - to send the message to an application in this processor or another processor.

The origin field of the RCPL contains the name of the application that issued ROUTC. The origin may be:

- The application that sent the message.
- The application to which the message is returned if it cannot be delivered.
- The 4 characters 'CPSE' if the message was sent by the System Error program. This program responds to the terminal when an ECB terminates abnormally. The text of the message sent by the System Error program is 'CHECK DATA AND CALL SUPERVISOR'.
- The 3 characters 'SNW' followed by a one character CPU identifier for the processor, if the message was sent by the SNA Output Message Writer (CSNW). The text of these messages indicate why an input message from the terminal could not be delivered to the application.
- The 3 characters 'SMP' followed by a one character CPU identifier when the message was sent by the TPF system. There are three classes sent with an origin of SMPc:
  1. Solicited and unsolicited console messages. This class of message includes replies to operator commands and error messages sent by the TPF system.
  2. System replies to a request to logon or logoff.
  3. Error messages generated by the system when the input message from a terminal addressed by an LEID cannot be delivered to the application. The text of the messages that can be returned by the system are:
    - APPLIC appl NOT ACTIVE
    - SYS.ERR ON MSG FROM appl MSG LOST
    - ACP CANNOT ACCESS appl LOGOFF
    - APPLIC appl NOT AVAIL. LOGOFF
    - CANNOT ACCESS appl SYS IN 1052
    - RESTRICTED. CRAT TERM INPUT ONLY
    - FILE ERR. CANNOT DELIVER MSG TO appl
    - CANNOT ACCESS appl SYS IN RESTART
    - CANNOT ACCESS appl PROCESSOR INACTIVE

The lower case letters 'appl' in this description are replaced with the actual 4-character TPF application name when the message is returned to the originator.

The PSV routine issues ROUTC to send the message to its final destination and issues EXITC macro to return control to the system.

## PSV Output Message Queueing

Optionally, the PSV routine may need to queue the message on DASD prior to transmission. See "Queue Manager" on page 269 for detailed information about queueing. A PSV routine may queue a message prior to delivery because:

- The message must be safe-stored to allow re-transmission in case of an error.
- The class of output is delivered only at specific times of day. For example, airline tickets or bulk data.
- The terminal is inactive. In this case the PSV routine queues the output message and requests a session. When the session is established, the output is dequeued and sent.

TPF provides a generalized queuing package to allow the user to queue messages. The queuing package provides the primitives to:

- Enqueue messages
- Dequeue messages
- Re-enqueue the last message dequeued, and
- Purge the queue

If the queuing package is needed, then the user must implement the following facilities:

- Response handling - Typically messages are queued until they are successfully delivered. If delivery is successful, then the next message on queue can be sent. If delivery is unsuccessful, then the message should be repositioned at the top of the queue and the message resent.
- Queue swing - Messages destined for an inoperable terminal can often be re-directed to another terminal in the same vicinity. This requires the user to write an operator command causing messages on the queue for an inoperable terminal to be enqueued to another destination.
- Timeout - When a response is lost or a message garbled in transmission, then there is no signal to send the next message or retransmit the last message. Typically, this problem is handled by a time initiated program that checks for lost responses and stalled queues. For each lost response or stalled queue, the PSV routine is started to resend the last message. If a failure persists, then the queue is declared inoperable.

## Defining PSV Routines

A PSV routine has a 6-character name and is associated with an ECB controlled program. There can be 128 PSV routines (96 user routines and 32 routines reserved for use by IBM). Each PSV name must be 1 to 6 characters with the characters limited to the letters A through Z and the numbers 0 through 9. The PSV names reserved for IBM use start with the letter I. When the PSV name is passed to the user program, the name is 6 characters, left-justified and padded with blanks (X'40's). The name chosen may imply a specific network (for example, TYMNET), a transmission protocol, or a specific device type. The ECB controlled program associated with a PSV name may be unique to the name or shared among several PSV names. Each program must be allocated and loaded before it can be used. The Online Loader is used to load a new version of a PSV program. See *TPF ACF/SNA Network Generation* for detailed information on PSVs.



## Logical End-Point Identifiers, Terminals, and PSV Routines

Logical End-Point Identifiers (LEIDs) give a resource a pseudo line, interchange and terminal address. The resource may be an LU, a group of terminals attached via an LU, or any entity the user chooses. An LEID is a short hand notation for the actual resource. The PSV routines map LEIDs to devices in the following ways:

**One-to-One** This is the simplest case where an LU is associated with an LEID. For example, an LU is associated with an LEID value of 123. In this case, every message from the LU is passed to the application as if it came from terminal 123, and every message sent by the application to terminal 123 is forwarded to the LU.

**One-to-Many** This implies that a single LEID could represent multiple destinations, for example, a distribution list. In this case, the PSV routine intercepts output messages destined for an LEID and forwards a copy of the message to each terminal implied by the LEID.

Similarly, which LEID to assign to a message from an LU is determined when the person signs in. For example, when a dial-in terminal is connected to the system, the LEID associated with the LU indicates the end user, geographical location, or authority.

**Many-to-One** This implies that several LEIDs could represent the same destination. For example, a hard copy terminal uses multiple LEIDs to imply multiple forms. A single terminal uses one LEID for administrative traffic, one for batch data and one for tickets. The PSV routine handling these LEIDs needs to queue and schedule output traffic based upon the particular printer forms that are loaded.

---

## Data Flow Control (DFC) Considerations

The SNA architecture defines the Data Flow Control (DFC) layer to provide the following services for LU-LU sessions:

- Assign sequence numbers.
- Correlate requests and responses.
- Group related request units into chains.
- Group related series of chains into brackets.
- Enforce session request and response mode protocols.
- Coordinate session send and receive modes.

Although DFC may provide all of these services for a particular LU-LU session, all may not be supported for that session. The options to be supported (for example BRACKETS) are agreed upon at session establishment time (BIND).

Associated with each RSC defined at TPF SNA generation time, certain services are available and/or precluded depending upon the device type specified. For example, BRACKET and Definite Response protocols are precluded for NPSI (MCH and VC) LUs, but available for 3270s. Thus, TPF supports a specific set of BIND values (images) for each device type and the generation process forces the device definition to conform to one of the supported images for that device.

The TPF SNA DFC layer communicates with the network through the TPF-provided lower level Transmission Control (TC) layer. The TC layer provides the following functions:



- Transmit/receive when data flow is enabled.
- Verify sequence numbers.
- Manage session-level pacing.

The TPF SNA Output Message Transmission (OMT) package, acting as owner and manager of the message queue, provides the DFC services. As an alternative to using OMT, the user is able to define and implement PSV exit routines which own and manage the message queue via Queue Manager, thus assuming the DFC responsibility. Although the user should provide the full complement of DFC functions, only those specified in the associated TPF BIND image can be used for that session. For example, the TPF BIND image for NPSI (MCH and VC) LU resources precludes the support of BRACKET protocols, and any use of BRACKET protocols for that session will cause errors.

## User DFC Interface

The DFC functions are closely related to the Request/Response Header (RH) indicator settings and DFC RU commands of the PIU. Upon receipt of a PIU, the RH and RU dictate which DFC services must be performed; similarly, the DFC functions that are used dictate the RH and RU of the outbound PIU. Because of this interdependence, TPF provides the user DFC with:

- The inbound sequence number and RH indicators
- The ability to set the outbound sequence number and RH indicators.

The ROUTC user exit and PSV routines provide the user interface to the SNA DFC layer of TPF SNA. The interface is described in Table 14 on page 261.

Table 14. DFC ROUTC Interface

RU Type	AM0SG Fields	RCPL Setting
1. Any DFC Command 2. Data	1. DFC Command <ul style="list-style-type: none"> <li>• AM0FCH = 0 (Chaining not permitted)</li> <li>• AM0CCT = length of DFC Command RU + 5</li> <li>• AM0TXT = DFC Command RU</li> </ul> 2. Data <ul style="list-style-type: none"> <li>• AM0FCH = 0 (Chaining not permitted)</li> <li>• AM0CCT = length of Data RU + 5</li> <li>• AM0TXT = Data RU</li> </ul>	<ul style="list-style-type: none"> <li>• RCPLORG &amp; RCPLDES fields</li> <li>• RCPLCTL0 (See Note 1.)</li> <li>• RCPLCTL1 = 0 (Reserved)</li> <li>• RCPLCTL2 = 0 (Reserved)</li> <li>• RCPLCTL3 = 0 (Reserved)</li> <li>• RCPLCTL4 = 0 (Reserved)</li> <li>• RCPLCTL5 = 0 (Reserved)</li> <li>• RCPLGDD = RCPLDRH (X'40')</li> <li>• RCPLCTR = X'07' (length of GDA)</li> <li>• RCPLGDA               <ul style="list-style-type: none"> <li>– X'06' - length of GDA data</li> <li>– 2 Byte PIU Sequence Number</li> <li>– 3 Byte SNA RH (See Note 2.)</li> <li>– 1 Byte Indicator (X'00')</li> </ul> </li> </ul>

### Notes:

1. RCPLCTL0 indicators:
 

<b>RCPL0DTY (X'80')</b>	indicates input (Destination = TPF Application)
<b>RCPL0OTY (X'40')</b>	indicates output (Origin = TPF Application)
<b>RCPL0TRM (X'02')</b>	indicates RID addressing used.
<b>RCPL0FMT (X'01')</b>	indicates expanded RCPL used.
2. TPF ensures that the setting of the following RH indicators (as defined in PIUEQ) are set to zero:

<b>RH0CIB (Byte 0, X'20')</b>	Session Control and Network Services.
<b>RH0AWM (Byte 0, X'10')</b>	Reserved
<b>RH1PEM (Byte 1, X'40')</b>	Reserved
<b>RH1RBM (Byte 1, X'08')</b>	Reserved
<b>RH1RLWIM (Byte 1, X'04')</b>	Reserved
<b>RH1QRI (Byte 1, X'02')</b>	Queued Response Indicator
<b>RH1PCEM (Byte 1, X'01')</b>	Pacing Indicator
<b>RH0B2CE (Byte 2, X'01')</b>	Conditional End Bracket Indicator (LU6.2 only)

## Outbound Message Interface

An application or PSV (Process Selection Vector) sends output messages using the ROUTC macro. If an output message cannot be delivered to its destination, the originating application/PSV may have requested that:

1. The output message is returned to the originator. The originator indicates this to TPF by setting the RCPL Returned Message Indicator off (RCPL0RET = 0).
2. The output message is queued by TPF for later transmission. The originator indicates this to TPF by setting the RCPL Returned Message Indicator on (RCPL0RET = 1).

The user DFC interface (see Table 14 on page 261) precludes the application/PSV from requesting the message that is queued. Outbound messages across the user DFC interface are to be considered as having been transmitted to the network. Queueing of the message for later transmission is the user DFC's responsibility.

Output messages that cannot be sent, such as when no session exists, are passed to the originator as input. In this situation, the RCPL Returned Message Indicator is set on (RCPL0RET = 1). Applications and PSV exit routines should be designed with the capability of processing returned output messages.

## Pacing Considerations

The TPF TC layer enforces the session pacing protocols. When the user provides the DFC layer, the TPF TC layer queues messages when the session is at the pacing limit or if a pacing queue exists. Upon the return from the ROUTC request, the user DFC should consider the message to have been transmitted. Messages stalled by pacing limits are not returned to the originator, but are discarded during session end processing.

## Undeliverable Message Considerations

Messages being returned by TPF to the application may differ from those sent by the application as the original messages may have been altered by the ROUTC Exit or a PSV routine. Thus, when using these facilities, regardless of the DFC layer implementation and/or use of PSV routines, TPF must return messages to the originator (application/PSV) via the same path (ROUTC Exit / PSV), rather than directly to the application via the RCAT enter expansion.

## Response Protocols/Error Handling

The user DFC is given all PIU responses (positive or negative) during the session. A TPF sample negative response analysis program per LU type is provided to perform the analysis for each -RSP sense code received:

- CMJ0: LU0 -RSP Handler Analysis. See Table 16 on page 263.
- CMJ1: LU1, LU2, LU3 -RSP Handler Analysis. See Table 17 on page 263.
- CMJ2: LUSTAT -RSP Handler Analysis. See Table 18 on page 264.

The PSV may use the TPF sample or a user modified version. The sample analysis programs are activated via ENTRC using the interface described in Table 15 on page 263.

Table 15. Sample Negative Response Handler Interface

Interface	Input	Returned
Registers	Irrelevant	<ul style="list-style-type: none"> <li>R0-R6: Not changed</li> <li>R7: Recommended Action (Reference Table 19 on page 264)</li> </ul>
ECB Workarea	Irrelevant	Not Changed
Data Levels	D0 = PIU	Not Changed
Protect Key	Working Storage	Working Storage

Upon return from the analysis segment, the user should start the processing corresponding to the action code(s) specified in register 7. See Table 19 on page 264 for detailed information.

**Note:** These do not include BATCH\_LU and SLU-P conditions.

Table 16. LU0 (3600, NPSI) -RSP Analysis

SENSE	Description	User Action
X'0802'	Intervention Required	HOLDQ
X'0813'	Bracket BID Reject - No RTR forthcoming	HOLDQ + WAIT
X'0814'	Bracket BID Reject - RTR forthcoming	HOLDQ + WAIT
X'0817'	XIO unsuccessful (GATE/PAD) MCH Inactive	HOLDQ
X'081B'	Receiver in Transmit	RESYNC
X'1xxx'	Request Error	TERMINATE
X'2xxx'	State Error	RESYNC
X'4xxx'	Request Header (RH) Usage Error	TERMINATE
X'8xxx'	PATH Error	TERMINATE

Table 17. 3274/76 (LU1, LU2, LU3) Sense Code Table

SENSE	Description	User Action
X'0802'	Intervention Required	HOLDQ + WAIT
X'0807'	Resource Not Available - LUSTAT Forthcoming	HOLDQ + WAIT
X'0811'	BREAK	RETRY
X'0813'	Bracket Bid Reject - No RTR Forthcoming	HOLDQ + WAIT
X'0814'	Bracket Bid Reject - RTR Forthcoming	HOLDQ + WAIT
X'081B'	Receiver in Transmit Mode	SIGNAL
X'0829'	Change Direction Required	RESYNC
X'082A'	Presentation Space Alteration	REFORMAT
X'082B'	Presentation Space Integrity Lost	REFORMAT
X'082D'	LU Busy	HOLDQ + WAIT

Table 17. 3274/76 (LU1, LU2, LU3) Sense Code Table (continued)

SENSE	Description	User Action
X'082E'	Intervention Required at LU Subsidiary Device	HOLDQ + WAIT
X'082F'	Permanent Error at LU Subsidiary Device	RETRY
X'0831'	LU Disconnected	HOLDQ + WAIT
X'1xxx'	Request Error	TERMINATE
X'2xxx'	State Error	RESYNC
X'4xxx'	Request Header (RH) Usage Error	TERMINATE
X'8xxx'	PATH Error	TERMINATE
X'FFFF'	DEFAULT	Use actions shown in Table 16 on page 263 if sense not found here.

Table 18. 3274/76 (LU1, LU2, LU3) LUSTAT SENSE TABLE

SENSE	Description	User Action
X'00010000'	LU Available (T1/T3)	WAKEUP
X'0001B000'	Printer - Available (T2)	WAKEUP
X'0001D000'	Display - Available (T2)	WAKEUP
X'00020000'	No Data to Send	WAKEUP
X'082B0000'	Available, Presentation Space Integrity Lost, Formatting Required.	REFORMAT
X'08310000'	Powered Off or Disconnected	HOLDQ + CD
X'FFFF0000'	DEFAULT	TERMINATE

Table 19. Negative Response Processing Actions

Action	Value	Processing Description
CD	X'00000001'	Yield direction to the remote. Set change direction indicator of the RH (RH0B2CD = 1) in the RCPL GDA for the next ROUTC request.
HOLDQ	X'00000010'	Suspend processing of messages queued for transmission. Queue all ensuing transmit requests.
REFORMAT	X'00000020'	Application must format the screen.
RETRY	X'00000030'	Resend message up to some user defined limit before starting HOLDQ and awaiting LUSTAT with available sense data.
RESYNC	X'00000040'	Initiate resynchronization process. Using the Session Services Interface request 'RESYNC' activation of this session.
SIGNAL	X'00000050'	Send SIGNAL DFC request to obtain direction.
TERMINATE	X'00000060'	No recovery possible for this error. Using the Session Services Interface request 'FORCED' termination of this session.
WAIT	X'00000002'	Do not initiate any further transmissions. Wait for an indication from the remote resource (that is, LUSTAT or RTR) before resuming transmit processing.

Table 19. Negative Response Processing Actions (continued)

Action	Value	Processing Description
WAKEUP	X'00000070'	Resource now available. Treat as Session Started Notification.

## Input Message Router Exit

Specifying COMEXIT=YES on the SIP MSGRT macro provides the user exit routine (segment COBC) to be included in the generated TPF system. The sample code released with TPF consists of nothing more than a 'BACKC' macro. The user should replace this with the desired logic.

This routine is given control whenever a message is received by the Message Router or the Communication Source package and the destination is an application. The message includes both those from terminals and those generated by another application.

It should be noted that upon return from this routine, TPF no longer examines the input message and all subsequent processing is based entirely upon the RCPL information. If the destination specified in the RCPL has been changed by this routine to anything other than a local TPF application, TPF issues a 'ROUTC' upon receiving control from this routine.

The INPUT/OUTPUT interface between the TPF packages (Message Router and Communication Source) and the user coded COBC segment is as follows:

### INPUT to COBC

- ECB WORK AREA:

```
EBW000-EBWXXX = The RCPL of the destination message
                  (The length depends on whether it is an expanded
                  RCPL and the size of the RCPL General Data Area.)
EBSW01-EBSW03 = 0
EBCM01-EBCM03 = 0
EBER01         = 0
EBX000-EBX063 = Unpredictable and may be used by COBC.
EBX064-EBX103 = Reserved and must not be altered.
EBXSW0-EBXSW7 = Unpredictable and may be used by COBC.
EBROUT        = Reserved and must not be altered.
CE1DBI        = Database ID of Basic Subsystem
CE1PBI        = Program Base ID of Basic Subsystem
CE1SSU        = SubSystem User ID of Basic Subsystem
CE1USA        = Unpredictable and may be used by COBC.
```

### ALL OTHER ECB FIELDS ARE RESERVED

- ECB DATA LEVELS:

```
D0 = INPUT MESSAGE IN AMSG FORMAT
D1 = If a core block exists, it is the AAA of the terminal in the
     Basic Subsystem. This level should not be altered by the user.
D2 = Reserved and may not be used by COBC.
D3 = If a core block exists, it is the RCB of the terminal in the
     Basic Subsystem. This level should not be altered by the user.
```

### ALL OTHER LEVELS ARE AVAILABLE FOR USE BY THIS ROUTINE

- REGISTERS:

R0-R7 = Unpredictable and available for use.  
 R8 = Base of program COBC  
 R9 = Base of the ECB  
 R14-R15 = Unpredictable and available for use.

## OUTPUT from COBC

- **ECB WORK AREA:**

EBW000-EBWXXX = The RCPL of the destination message  
 (The length depends on whether it is an expanded  
 RCPL and the size of the RCPL General Data Area.)  
 EBSW01-EBER01 = Unpredictable, but will be set to zero by TPF prior  
 activating the application.  
 EBX000-EBX063 = Unpredictable and TPF may use it.  
 EBX064-EBX103 = reserved and must not be altered.  
 EBXSW0-EBXSW7 = Unpredictable.  
 EBR0UT = Reserved.  
 CE1DBI = Must not be altered  
 CE1PBI = Must not be altered  
 CE1SSU = Must not be altered  
 CE1USA = Unpredictable

- **ECB DATA LEVELS:**

D0 = INPUT MESSAGE IN AM0SG FORMAT  
 (USER MAY MODIFY THE CONTENT OF THE MESSAGE)  
 D1 = MUST NOT BE ALTERED BY USER  
 D2 = MUST NOT BE ALTERED BY USER  
 D3 = MUST NOT BE ALTERED BY USER

## ALL OTHER LEVELS MUST NOT BE HOLDING ANY CORE BLOCK

- **REGISTERS:**

R0-R7 UNPREDICTABLE  
 R14-R15 UNPREDICTABLE

- ***EXIT from COBC must be VIA 'BACKC'***

The TPF services and facilities that may be started by the routine are:

The Non-SNA Log Processor.  
 The Unsolicited Message Package.  
 The SMP Prefixing Facility.  
 All TPF macros.

The following are examples of the services that could be implemented in a user's input edit routine.

### **Data Transformation**

The input message text may be modified to a format acceptable to the application. This allows users to attach new terminals to existing applications and non-SNA terminals to new applications.

### **Data De-compaction and De-compression**

Data, encoded for transmission efficiency, is decoded.

### **Message Verification**

An error message may be returned to the message originator and the message discarded.

### **Prefixing**

This special case of transaction analysis allows system operators to denote the application, subsystem or subsystem\_user to process the command entered. For example, TPF supports operator command prefixing, where the command is prefixed with the application, subsystem or subsystem\_user name followed by the character /. For many users this

format is unacceptable and operationally awkward. Therefore, for those users that wish to use an alternate prefixing technique, the input edit routine may translate an installation defined prefix to the prefix format required by TPF.

### **Transaction Analysis**

The input text may be analyzed to determine the application to process the input message.

### **User Written Transaction Routing**

Some users may wish to bypass the TPF Log Processor and create their own. For example, assume a network where terminal users frequently access multiple TPF applications and the TPF requirement for a terminal user to logon is too costly in terms of operator efficiency. Here the input edit routine could be used to implicitly logon the terminal based on the input message text. The character \$ may be used to signify an implicit logon and the character immediately following the \$ signifies the application. Further assume that the presence of text following \$aaaa determines whether the connection is just for this message or all following input. Then, in a network where terminal users frequently require access to multiple applications, the terminal user could simply prefix their input rather than logging into the required application.

### **Unsolicited Message**

The user edit routine may request the next unsolicited message queued to the terminal and transform the input into a LOGU request, setting the input destination to the 4 characters, LOGI.

### **ECB Set Up**

The user edit may set or change the following:

- Information describing the input or its source may be passed to the application in the General Data Area of the expanded RCPL.
- The input RCPL could be copied to an installation defined area in the ECB. The saved RCPL would provide a backup copy of the input RCPL allowing pre-Message Router applications, say RES0, to create an output RCPL and issue ROUTC.
- ECB levels 0, and 4 through 15 may be used to pass data.
- Change the format of the data from AM0SG to OM0SG. Note, the appropriate RCPL indicator must be set to reflect this change.

Input from the following sources is passed to the user Input Edit routine:

- SNA/SDLC input
- ALC input
- SLC pseudo terminal input
- Binary Synchronous Station Names
- Application to application routing within the processor
- Undeliverable output messages (Returned Messages).





---

## Queue Manager

The Queue Manager package is a basic set of tools to perform queueing functions. The primitives provided by this package are:

- **ENQUEUE** queues a message FIFO.
- **GET**
  - **Get Message** retrieves a copy of the entire message, that was previously enqueued, FIFO.
  - **Get N Bytes** retrieves part of a message, up to the number of bytes specified by the user, FIFO.
- **DEQUEUE** removes the previously retrieved message from the queue, and if requested, releases all file addresses that are associated with the previously retrieved message.
- **WASH** repositions the previously retrieved message at the top of the queue.
- **PURGE ALL** purges all messages from the queue.

The control block used for queue management is called a Queue Control Element (QCE). This block is created and maintained by the user. When interfacing with the Queue Manager, the QCE must reside in **main storage** and must have a protect key of working storage. The user must have exclusive control over the QCE when requesting queueing services. As illustrated in Table 20 on page 269, the user must adhere to the layout of a QCE.

Table 20. Queue Control Element Layout

Field Label	Displacement	Byte	Bit	Comments
IQCEFOQ	0	4		First of Queue
IQCELOQ	4	4		Last of Queue
IQCESCUR	8	4		Start of Current Message
IQCECUR	12	4		Current Block
IQCEOFF	16	2		Next Available Byte for GET N
IQCEPARM	18	16		User Parameter Area

Data messages queued off a QCE must be in the standard AMSG format. The message text should be queued in a device-independent format to facilitate potential queue swing operations. That is, reformatting of the data should occur after the message has been dequeued. The message block size can be of 381, 1055, or 4K type. The maximum amount of text that can fit within a 381 byte block is 342, within a 1055 byte block is 1016, and within a 4K byte block is 4027.

When queueing a message that consists of multiple blocks, the prime block should be in main storage and the remaining blocks should be file chained, using the AMSG chaining scheme. The AM0BCH field is reserved for system usage. All chained blocks within the same message must be of the same size.

When retrieving a message the user has the following options:

- Get Message** The user requests to receive the message as given to the Queue Manager at the time of enqueueing.
- Get N Bytes** The user requests to receive a portion, up to N bytes, of the

message. If the number of bytes specified spans more than one block, the Queue Manager assembles it into one block.

## Queue Manager Interface

The Queue Manager parameter list, IQPRM (Table 21), is used for both input and output when interfacing with the Queue Manager. The user must provide this parameter list, load its address in register one (R1), and enter (ENTRC) segment CMX0 when issuing a queueing request. After completing the user-requested service, the Queue Manager returns the parameter list and sets indicators to identify successful or unsuccessful queue management results.

**Note:** The Queue Manager package is subsystem independent. It is the user's responsibility to ensure the subsystem environment is initialized when entering the Queue Manager package.

Table 21. Queue Manager Parameter List

Field Label	Displacement	Byte	Value	Comments
IQPRFUNC	0	1	X'00' X'01' X'02' X'03' X'04' X'05'	ENQUEUE GET Message GET N Bytes DEQUEUE Message WASH PURGE All
IQPRINDC	1	1	X'80'	Release File
IQPRGENR	2	1	X'00' X'01' X'02'	Successful Completion Unsuccessful Completion Parameter List Error
IQPRDETL	3	1	X'01' X'02' X'03' X'04' X'05' X'06' X'07' X'08' X'09' X'0A' X'0B' X'0C' X'0D' X'0E'	End of Message Queue Empty I/O Error No Message Available for Wash GET Message Function Not Valid or Not Performed Consecutive GETs Issued DEQUEUE Function Not Valid Function Not Supported GET N Byte Function Not Valid Invalid Target Offset for GET Invalid Record ID No Block Attached On D0 Invalid QCE Address Invalid Block Attached On D0 or Can't Queue the Message
IQPRQCE	4	4		QCE core address
IQPRRET	8	2		Number of bytes for GET N Bytes
IQPROFF	10	2		Target offset for GET N Bytes
IQPRID	12	2		2-character record ID
IQPRPARM	14	16		User parameter area

## Input Interface

The Queue Manager Input Interface is detailed in Table 22 on page 271.

Table 22. Queue Manager Input Interface

Registers	ECB Workarea	ECB Data Levels
R1 : Parameter List Addr - IQPRM	EBW - Irrelevant EBX - Irrelevant Protect Key of Working Storage	<ul style="list-style-type: none"> <li>• ENQUEUE Message D0: Prime block of the message DD-DF: Available D1-DC: Reserved</li> <li>• GET Message D0, DD-DF: Available D1-DC: Reserved</li> <li>• GET N Bytes D0: Prime block of the message DD-DF: Available D1-DC: reserved</li> <li>• DEQUEUE Message D0, DD-DF: Available D1-DC: Reserved</li> <li>• WASH/PURGE All DD-DF: Available D0-DC: Reserved</li> </ul>

For all queueing services, the QCE core address must be present in the parameter list. The user is responsible for initializing the QCE only before the first queueing request.

#### Enqueue

When the Queue Manager is started with the Enqueue function, the prime block of the message must be on data level zero (D0) of the ECB. Additional blocks should be file chained using standard AMSG format.

The user must include within the parameter list the record ID that the Queue Manager should use. Prime record attributes are used by the Queue Manager on the specified record ID. The user may also include within the parameter list a 16-byte parameter area. This area is saved by the Queue Manager and returned to the user in the QCE when the message is dequeued.

The prime block on level D0, provided by the user, is released by the Queue Manager.

#### Get Message

When the Queue Manager is started with the Get Message function, the message at the top of the QCE queue is returned to the user as **originally enqueued**. The prime block, returned by the Queue Manager, resides on data level zero (D0) of the ECB and any additional blocks are file chained using standard AMSG format. The 16-byte user parameter area is returned in the QCE.

Upon issuing the Get Message request, the user must include within the parameter list the record ID that the Queue Manager should use.

#### Get N Bytes

When the Queue Manager is started with the Get N Bytes function, the user must provide sufficient space within a single target core block on data level zero (D0) of the ECB. It is at this ECB location where the Queue Manager returns up to N bytes from the current message, source block, in the QCE.

The user may also specify a target offset (M bytes), within the parameter list. M indicates that the Queue Manager should start filling the target block on level zero (D0) M bytes after the 18th byte of the block. If M is equal to zero, the Queue Manager fills the target block starting at location 18.

**Note:** The text is extracted from the source block starting at location 18 within the prime block and location 23 within the overflow blocks.

The Queue Manager does not impose any values within the first 18 bytes of the returned block.

Upon return, the Queue Manager places the actual number of bytes returned in the IQPRRET field of the input parameter list.

The user must include within the input parameter list the record ID that the Queue Manager should use.

If the Get N Bytes function has been used to retrieve a message, the Get Message function cannot be used for retrieving any remaining bytes of the same message. The "End of Message" return code indicates that the message has been fully dequeued.

<b>Dequeue</b>	When the Queue Manager is started with the Dequeue function, the previously returned message is removed from the queue. The user may also specify the Release File option with the Dequeue function. In that case, all file addresses of the previously returned message are released.  The Dequeue function must be specified between two consecutive "Get" operations.
<b>Wash</b>	When the Queue Manager is started with the Wash function, the QCE structure is rearranged where the previously retrieved message is positioned at the top of the QCE queue.
<b>Purge All</b>	When the Queue Manager is started with the Purge All function, the QCE queue becomes empty. All file addresses are released.

## Output Interface

The Queue Manager Output Interface is detailed in Table 23 on page 273.

Table 23. Queue Manager Output Interface

Registers	ECB Workarea	ECB Data Levels
<ul style="list-style-type: none"> <li>• R1 : Parm List addr - IQPRM</li> <li>• R0, R2-R7: Unchanged</li> </ul>	EBW - Unchanged EBX - Unchanged Protect Key of Working Storage	<ul style="list-style-type: none"> <li>• ENQUEUE Message D0-DC: Unchanged DD-DF: Unpredictable</li> <li>• GET Message D0: Prime block of the message D1-DC: Unchanged DD-DF: Unpredictable</li> <li>• GET N Byte D0: Prime block of the message D1-DC: Unchanged DD-DF: Unpredictable</li> <li>• DEQUEUE Message D0: Prime block of the message D1-DC: Unchanged DD-DF: Unpredictable</li> <li>• WASH/PURGE All D0-DF: Unchanged DD-DF: Unpredictable</li> </ul>

The input/output parameter list is returned to the user upon completion of every queue service request. The Queue Manager also sets two types of indicators (general and detailed) to communicate the actions taken (or attempted) by the Queue Manager to the user as detailed in “General Return Codes” and “Detailed Return Codes”.

## General Return Codes

The general indicator returns one of these results:

- Successful completion - The type of information that exists within the detailed indicator is:
  - End of Message.
- Unsuccessful completion - The detailed indicator must be examined to further understand the type of error incurred. The type of information that exists within the detailed indicator is:
  - Queue Empty
  - I/O Error
  - No Messages Available for Wash
  - Get Message Function Not Valid
  - Dequeue Function Not Specified
  - Insufficient Core for Enqueue
- Parameter List Error

A parameter in the user supplied parameter list is in error. The detailed indicator byte must be examined for one of the following indications:

- Function Not Supported
- Invalid Number of Bytes for Get
- Invalid Target Offset for Get
- Invalid Record ID

## Detailed Return Codes

- End of Message

The End of Message return code is related to the Get N Bytes function. When the last part of the message is returned, the Queue Manager posts successful completion in the general indicator and end of message in the detailed indicator.

- Queue Empty

The Queue Empty return code is posted in the detailed indicator byte when the Queue Manager attempts to get a message from an empty QCE.

- I/O Error

The I/O Error return code is posted in the detailed indicator byte when an I/O error has been detected during manipulation of file records.

- No Messages Available for Wash

The No Messages Available for Wash return code is posted in the detailed indicator byte when the Wash function is requested and the file address of the previously returned message is not present in the QCE.

- Get Message Function Not Valid

The Get Message Function Not Valid return code is posted in the detailed indicator byte when the user requested to retrieve a message via Get N Bytes, and before receiving the end of message indicator the user requested further dequeuing via Get Message.

- Dequeue Function Not Specified

The Dequeue Function Not Specified return code is posted in the detailed indicator byte when the user requested two consecutive Get operations without issuing a Dequeue operation in between.

- Insufficient Core for Enqueue

The core block that corresponds to the record ID that the user specified is not large enough to fit the prime block of the message being enqueued.

- Function Not Supported

The user has requested a function that is not supported by the Queue Manager.

- Invalid Number of Bytes for Get

The Invalid Number of Bytes for Get return code is posted in the detailed indicator byte when the user has requested the Get N Bytes function with a non-positive number of bytes, or insufficient space has been provided within the core block on D0 for dequeuing.

- Invalid Target Offset for Get

The Invalid Target Offset for Get return code is posted in the detailed indicator byte when the user has requested the Get N Bytes Function with a target offset either negative or too large for dequeue.

- Invalid Record ID

The Invalid Record ID return code is posted in the detailed indicator byte when the user has requested the Enqueue function with an invalid record ID.

---

## Diagnostic Aids

TPF/SNA diagnostic and error processing is consistent with TPF philosophy; recover from errors as quickly as possible and save as much information as possible. TPF automatically records error information. This helps identify the causes of frequent temporary errors. Users can optionally request TPF tracing and testing services. Whenever possible, TPF avoids terminating sessions or deactivating resources.

The TPF operator receives a message when a permanent error is detected. The message specifies the name of the failing resource, the time that the error occurred, and text to describe the error. These steps should be followed to accumulate adequate diagnostic documentation:

- Save all console logs that pertain to the error. These logs reflect TPF's actions up to and including the error message.
- Obtain printouts of any traces initiated because of the error.
- Request and save status displays of the resources involved in the error.
- Initiate online testing of the devices involved in the error.

Deactivating the affected part of the network may temporarily circumvent the problem. However, users should maintain a complete description of the network as it existed before the error.

---

## Trace Functions

TPF supports the following SNA communication trace functions:

- Operating system traces
- Path information unit (PIU) trace facility
- SNA I/O trace facility.

### Operating System Traces

The TPF operating system traces monitor supervisor calls (SVCs) and input/output (I/O) interrupts. These traces produce a detailed image of the environment at the time the interrupt occurred. For more information about the realtime trace (RTT), see the *TPF Program Development Support Reference*.

### Path Information Unit (PIU) Trace Facility

The path information unit (PIU) trace facility provides a detailed trace of the data transferred between the TPF system and remote resources. See Appendix D, "Using the Path Information Unit (PIU) Trace Facility" on page 291 for more information about the PIU trace facility.

### SNA I/O Trace Facility

The SNA I/O trace facility provides a detailed I/O trace for SNA I/O interrupts that occur during Network Control Program (NCP) XID exchanges, adjacent link station (ALS) XID exchanges, and channel-to-channel (CTC) XID exchanges. Information about significant steps in XID7 processing for CTC devices is also logged in this table. It also provides a detailed I/O trace for NCP, ALS, and CTC asynchronous interrupts. Normal data transfer operations and attention-only interrupts that occur during NCP and ALS data transfer operations are not traced.

The I/O interrupts and the asynchronous interrupts are stored in a 4K-byte table that operates in wrap-around mode. This SNA I/O trace table contains header data and a maximum of 63 entries. Each entry in the trace table has a length of 64 bytes, which includes one byte that is used for status information. The following data is stored in the SNA I/O trace table entries:

- Channel command word (CCW) index
- Condition code (CC)
- Channel status word (CSW)
- Command associated with the first CCW for NCP and ALS XID exchanges, or the commands associated with the first two CCWs for CTC XID exchanges
- CTC status
- XID information field (I-field) contents.

The SNA I/O trace facility is initially active for each device defined in the symbolic device address table (SDAT). The SDAT is part of the SNA keypoint that is defined by the CK2SN data macro.

To display the contents of the SNA I/O trace table online, issue the ZDDCA command using the XID dump tag. The SNA I/O trace table is also included in system error dumps that have the control program keyword (ICP) specified in the dump override table. For more information about the XID dump tag and the dump override table, see the *TPF Program Development Support Reference*. For more information about the ZDDCA command, see *TPF Operations*.

---

## Reliability and Serviceability

TPF's reliability and serviceability support is provided to maintain operation of the network. It includes:

- Error Detection and Feedback:  
Before acting upon any request, TPF analyzes the request for errors. If an error is detected, TPF returns the request along with an error indicator.
- Hardware Error Recovery:  
When possible, TPF support for I/O handlers and NCP attempts automatic recovery from an error. If recovery is possible, processing continues. Otherwise, a record of the error is written to the realtime log tape.
- NCP Slowdown:  
TPF detects an NCP slowdown condition and assists the NCP in return to normal operation.

## Error Detection and Feedback

All TPF operator and application program-initiated requests are tested for errors. Errors detected in an operator request result in the return to the CRAS terminal of the request and an indication of the error. If an invalid condition is detected while processing a request, TPF: 1) stops processing the request, 2) informs the operator, and 3) resets the affected resources. TPF discards invalid application requests and starts a system error dump to document the error.

In addition to invalid request notices, TPF also sends discrete error messages to the system console. These messages describe unusual conditions that affect operation of the network such as permanent hardware errors, communication line failures, or negative responses to SNA commands.



## Hardware Error Recovery

When an I/O interrupt occurs, TPF (via the appropriate error recovery procedure) tries to determine the type of error and take the appropriate recovery action. When a non-recoverable error is encountered, a message is written to the system console and an error record is written to the realtime log tape.

The NCP provides recovery for the devices attached to it. When an NCP completes error-recovery processing, it forwards a Maintenance Data Record (MDR) to TPF. The NCP also maintains temporary error counters for each device. These counters are sent to TPF (as MDRs) whenever the counters overflow or the NCP is deactivated. MDRs are written to the realtime log tape.

The record of hardware errors on the realtime log tape is formatted and printed for analysis offline.

## NCP Slowdown

When an NCP's message buffer is filled, the NCP automatically enters slowdown mode. In slowdown mode, the NCP reduces the number of messages it accepts from both the communication lines and host processors. Further, the NCP tries to increase its output message rate. When NCP notifies TPF of a slowdown condition, TPF stops sending data to the NCP. TPF continues to accept requests from application programs, but queues those requests directed to a slowed NCP. When an NCP exits slowdown mode, the queued requests are processed and sent. TPF data collection programs record each occurrence of an NCP slowdown.

If an NCP remains in slowdown for an extended period of time, a large output queue can build up in the TPF system. The SLOWTIME parameter on the SNAKEY macro in keypoint 2 (CTK2) determines how long an NCP is allowed to be in slowdown before the TPF system breaks the connection to the NCP.

See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro.

## CTC Slowdown

TPF recognizes buffer shortage conditions on the host on the other side of the CTC connection. VTAM indicates slowdown in one of the control fields passed on a data transfer operation. TPF honors this indication and stops initiating writes until a VTAM write operation is received with the slowdown indicator reset. TPF responds to attention interrupts by issuing the channel program with read buffers, but the write count only includes the 8 bytes of control information.

To the receiver, the slowdown indicator means that the data that has been written is now rejected. In slowdown, TPF includes the total data size in the write count. VTAM only reads 8 bytes. TPF supports slowdown as a receiving system; that is, it sets the slowdown bit if a read buffer is not available.



---

## Data Collection/Reduction and Test Tools

TPF data collection/reduction programs provide data about the SNA network. Specifically, these programs delineate message activity from:

- An NCP, CTC or ALS
- Application programs

TPF data collection and reduction is described in *TPF System Performance and Measurement Reference*.

TPF provides the support functions listed below to test application programs. See *TPF Program Development Support Reference* for additional information about these functions.

- Realtime Trace (RTT)

Realtime Trace (RTT) permits users to trace TPF macros issued by programs that process logical unit input messages. Users can trace one logical unit and/or all logical units located on an NCP.

- Diagnostic Output Formatter

The DOF is used to format the RTT output. DOF provides a listing of: 1) input and output messages for each logical unit, and 2) macros issued by application programs.



---

## Appendix A. Logical Unit Status (LUSTAT)

**Note:** The logical unit status information described in this section does not apply to LU 6.2; see *IBM Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2* for this information.

A logical unit sends status information to its session partner via a mechanism called logical unit status (LUSTAT). The format of the SNA request unit (RU) when status information is sent is as follows:

### Byte

0	=	X'04'	Indicates this is LUSTAT
1-2	=	X'0001'	Indicates component is now available
	=	X'0002'	Component failure: Permanent error
	=	X'0003'	Component failure: Insufficient resource
3-4			User field; these two bytes may be used as an extension to bytes 1-2.

TPF only receives LUSTAT messages; it does not send them. Further, TPF does not receive LUSTAT messages from the following logical units:

- IBM 3614/3624 Consumer Transaction Facility
- Logical units attached to the 3271 control units

LUSTAT messages from 3274/3276 attached logical units are passed to TPF's output message transmission (OMT) program. TPF needs this status information to perform error processing. LUSTAT messages received from these logical units are listed below. The term primary, as used here, refers to a display station. The term secondary refers to a printer.

X'00010000'	Component now available
X'0001B000'	Secondary component now available
X'0001D000'	Primary component now available
X'00020000'	Component has no data to send
X'081CD000'	Component failure: Permanent error in primary device
X'081CB000'	Component failure: Permanent error in secondary device
X'082B0000'	Temporary error; component available
X'08310000'	Component powered off or disconnected

LUSTAT messages from all other logical units are passed to the application in the associated message block on data level 0. In order to let the application program distinguish between an SNA command and user data, the Expanded RCPL GDA area will contain the RH combined with the Sequence Number. The RH contains the RU category to distinguish between user data (FMD) and SNA commands (NC, DFC, SC). SIGNAL messages are handled in the same manner.



## Appendix B. General Format of the SNA BIND Command

### BIND Images for TPF Supported Secondary Logical Units

Valid RU indicator settings for each supported logical unit are listed below. See the *IBM Systems Network Architecture Format and Protocol Reference Manual: Architectural Logic* for an explanation of a particular indicator. See “Acceptable BIND Image For a Local Host Node SLU” on page 284, for bind images for a local host node SLU.

Table 24. BIND Images for TPF-Supported Secondary Logical Units

Secondary Logical Unit	BIND TYPE	FM PROF	TS PROF	PRI PROT	SEC PROT	COMM PROT	S-PACE	MAXRU	P-PACE
CLU	X'01'	X'03'	X'03'	X'B1'	X'A0'	X'3040'	X'0001'	X'0000'	X'0200'
3614 directly attached to a 37x5	X'01'	X'03'	X'04'	X'30'	X'30'	X'0040'	X'0101'	X'8585'	X'0000'
3614 attached to a 3601/3602	X'01'	X'03'	X'04'	X'30'	X'30'	X'0040'	X'0101'	X'8585'	X'0000'
Terminals attached to 3271	X'01'	X'02'	X'02'	X'00'	X'40'	X'0000'	X'0000'	X'0000'	X'0000'
Terminals attached to 3274/3276 (See note 1 on page 284 and note 2 on page 284.)	X'01'	X'03'	X'03'	X'B1'	X'90'	X'3080'	X'0000'	X'8585'	X'0000'
Remote host node SLU	X'01'	X'04'	X'04'	X'B1'	X'B1'	X'7080'	X'3F3F'	X'8785'	X'0000'
ALC/NEF logical units	X'00'	X'02'	X'03'	X'F0'	X'90'	X'0800'	X'0000'	X'8787'	X'0000'
3600 LU single thread, non-recoverable, brackets not used	X'01'	X'03'	X'04'	X'D0'	X'90'	X'4040'	X'0000'	X'8785'	X'0000'
3600 LU single thread, recoverable, brackets not used	X'01'	X'03'	X'04'	X'F0'	X'90'	X'4040'	X'0000'	X'8785'	X'0000'
LU single thread, recoverable, brackets are used	X'01'	X'03'	X'04'	X'F1'	X'91'	X'7080'	X'0000'	X'8785'	X'0000'
3600 LU single thread, non-recoverable, brackets are used	X'01'	X'03'	X'04'	X'D1'	X'91'	X'7080'	X'0000'	X'8785'	X'0000'
3600 LU multi-thread, non-recoverable	X'01'	X'03'	X'04'	X'D0'	X'90'	X'4000'	X'0000'	X'8785'	X'0000'
Batch transfer LU	X'01'	X'03'	X'04'	X'F0'	X'F0'	X'4080'	X'0000'	X'8785'	X'0000'
FTPI	X'01'	X'03'	X'03'	X'90'	X'90'	X'0040'	X'0000'	X'8989'	X'0000'
FMMR	X'00' or X'01'	X'02'	X'03'	X'80'	X'80'	X'0000'	X'3F00'	X'8787'	X'0000'

Table 24. BIND Images for TPF-Supported Secondary Logical Units (continued)

Secondary Logical Unit	BIND TYPE	FM PROF	TS PROF	PRI PROT	SEC PROT	COMM PROT	S-PACE	MAXRU	P-PACE
LU 6.2	X'00'	X'13'	X'07'	X'B0'	X'B0'	X'D0B1'	X'3F3F'	X'F8F8'	X'3F3F'
NPSI (See note 3.)	X'01'	X'03'	X'03'	X'90'	X'90'	X'0040'	X'0000'	X'8989'	X'0000'
NEF (ALCI)	X'01'	X'03'	X'03'	X'B0'	X'90'	X'0800'	X'0000'	X'8787'	X'0000'

**Notes:**

- For SLU type 1 (printer in SCS mode):
  - Byte 18 = X'F9'
  - Bit 0 = 1 BS, CR, INP, END, LF, HT, VT May Be Sent
  - 1 = 1 SHF May Be Sent
  - 2 = 1 SVF May Be Sent
  - 3 = 1 SVF (Channel) And SEL May Be Sent
  - 4 = 1 SLD May Be Sent
  - 5 = 0 Reserved
  - 6 = 0 Reserved
  - 7 = 1 TRN, IRS May Be Used
- For SLU type 2 (CRT) Or SLU type 3 (Printer):
  - For a 3277/3284/3286/3288:
    - Byte 24 = X'01' For Model 11 (12 By 40 Screen Only)
    - = X'02' For Model 12 (24 By Screen Only)
  - For a 3278/3287/3289:
    - Byte 20 = X'18' Default Line Number Of 24 For Mod 2,3,4
    - 21 = X'50' Default Column Number Of 80 For Mod 2,3,4
    - 22 = X'18' Alternate Line Number of 24 For Mod 2
    - 23 = X'50' Alternate Col Number Of 80 For Mod 2,3,4
    - 24 = X'7F' Refer To Bytes 20-23 For Screen Definition
- For NPSI, the defined TPF BIND images using TS Profile 4 are dynamically modified during session initialization processing to indicate that TS Profile 3 is currently being used. This modification applies only to resources defined as using PSV routines that supply the user DFC layer. This precludes the use of the 'Set and Test Sequence Numbers (STSN)' session control command.

## Acceptable BIND Image For a Local Host Node SLU

**Note:** This BIND image does not apply to LU 6.2; see *IBM Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2* for this information.

Byte	Value	Meaning
0	X'31'	BIND Request Code
1	0000.... ....0001	BIND Format 0 BIND Type 1 (non-negotiable)
2	X'04'	FM Profile 4 (LU-LU)
3	X'04'	TS Profile 4 (LU-LU)



Byte	Value	Meaning
4	1..... .0..... ..11.... ...0... ....0.. .....0. .....1	Primary LU Protocol Multiple RU Chains Immediate Request Mode Definite/Exception Response No 2 Phase Commit Reserved No Compression Primary may send end bracket (EB)
5	1..... .0..... ..11.... ...0... ....0.. .....0. .....X	Secondary LU Protocol Multiple RU Chains Immediate Request Mode Definite/Exception Response No 2 Phase Commit Reserved No Compression Secondary may/may not send EB (See Note 1)
6	0..... .1..... ..1..... ...X.... ....0... .....0.. .....0. .....0	Common Protocol No Segmentation FMH Allowed Bracket Protocol Used Bracket Termination Rule (See Note 2) Alternate Code Not Used Sequence Numbers Not Available For Sync Point Resynch BIS Not Sent BIND Cannot Be Queued
7	10..... ..0..... ...0.... ....00.. .....0. .....0	Common Protocol Half Duplex Flip/Flop PLU has Recovery Responsibility SLU is Contention Winner Alternate Code Processing Identifier No Control Vectors After SLU Name Reset State is SEND for SLU
8	..xxxxxx	SLU Send Pacing Count
9	..xxxxxx	SLU Receive Pacing Count
10	xxxx.... ....xxxx	SLU to PLU RU Size (See Note 3) Mantissa Exponent
11	xxxx.... ....xxxx	PLU to SLU RU Size (See Note 4) Mantissa Exponent
12	..xxxxxx	PLU Send Pacing Count
13	..xxxxxx	PLU Receive Pacing Count
14	00000000	LU Type LU Type 0

#### Notes:

1. This indicator may be set to zero or 1. Zero tells the SLU not to send the end bracket indicator. One tells the SLU to send the indicator. If begin bracket is sent, TPF sends both the begin and end bracket indicators.
2. Either conditional (rule 1) or unconditional (rule 2) termination is used.
3. TPF sends a request unit size of 3840 or the value specified in the PLU BIND command; TPF sends the smaller of the 2 values.

4. The request unit size must not exceed 3840.

---

## Appendix C. Interface Requirements for System Utility Programs

---

### RID and RVT Conversions

Use the INQRC and RIDCC macros to perform RID and RVT conversions. For more information about these macros, see *TPF General Macros*.

The NAU conversion program (COVX) is still provided for migration purposes. You can still use COVX for RID and RVT conversions, however; it is recommended that you use INQRC and RIDCC instead.

You can no longer use your own programs for RID and RVT conversions. Any references to such programs must be changed to use the INQRC or RIDCC macro.

---

### Retrieving NCB and SPA Data Records (CSNB)

You **must** use the CSNB segment to retrieve node control block (NCB) or scratch pad area (SPA) records in the applications that you develop. For more information refer to “Allocating or Retrieving a Scratch Pad Area (SPA) for a Dynamic LU” on page 190 or Migration Guide. You can also use this segment to retrieve the following:

- The file address of an NCB record or SPA record
- An NCB record or an SPA record
- An NCB record or SPA record with either the Find and Wait (FINWC) macro, or the Find, Wait and Hold (FIWHC) macro.

If an error occurs, you can return to the calling program or exit using the EXITC macro.

The CSNB segment is activated using the ENTRC macro.

Input requirements are as follows:

Register 1 must point to a 5-byte work area aligned on a halfword boundary. The format is:

Bytes 0-1	Contains the 2-byte resource identifier (RID) for the requested NCB or SPA.
0-2	Contains the 3-byte resource identifier (RID) for the requested NCB or SPA.
3	Available data level for which the data record may be retrieved. This must be a multiple of 8. For example, X'00' means data level 0, X'08' means data level 1 and X'10' means data level 2, etc.
4	Option Byte
Bit 0	= 0. The NCB is requested
	= 1. The SPA is requested
1	= 0. Retrieve the data record
	= 1. Return the file address of the data record (no retrieval is required).
2	= 0. Issue a FIWHC macro to retrieve the record

- = 1. Issue a FINWC macro to retrieve the record
- 3 = 0. Return to caller on an error condition
- 4 = 0. Retrieve 2-byte RID from bytes 0-1.
- 5-7 = 1. Retrieve 3-byte RID from bytes 0-2.
- Reserved and should be set to zero.

CNSB output is as follows:

- Registers:

- R1 = 0 If the requested function was performed successfully
- = 1 If an error was encountered in calculating the file address of the record
- = 2 If an error was encountered on FINWC or FIWHC macro
- = 3 If the input parameter is invalid
- R1 - 5 = Same as on entry
- 6 - 7 = Contents unpredictable

- Data Levels:

- CE1CRx = Address of the main storage block containing the requested data record
- CE1FAx = File address of the requested record

- Entry Control Block Work Area

EBX000 - EBX009 are used by this program

CE1UR2 is the beginning of this program's save area for caller's registers 2 - 5

---

## Select A Thread Utility Routine (SELEC)

This macro interfaces user application programs that support network qualified names to the select a thread utility programs. It is intended that this interface become the single interface to select a thread as user's migrate to take advantage of new networking functions such as network qualified names. See *TPF General Macros* for more information about the SELEC macro.

---

## Select A Thread Utility Program (CSF0)

This program performs the following services for a local host node SLU when it communicates with a remote primary logical unit:

- Constructs the output RCPL based on the caller's choice of either: 1) a specific SLU thread, or 2) the SLU with the least number of output messages on queue.
- Returns the node names of the session partners given the input RCPL associated with a message sent between the two.

Users issue an ENTRC macro to CSF0 when the request is to build an RCPL; an ENTRC to CSF1 when the names of the session partners are requested. The input requirements when requesting an output RCPL to be built are as follows:

Registers:

R1: Address of a 20 byte parameter list in the following format:

Byte 0-7 Node name of the destination. If the name

is less than 8 characters, it must be padded with blanks (X'40') on the right.

8-15 Node name of the origin. This is the 4 character TPF application name (as defined in the MSGRTA macro of SIP) with 4 blank characters appended to it.

16 One byte thread specification.

= 0 if the output RCPL is to be built using the SLU thread with the least number of messages on its queue.

= a binary number (1-255) of the specific thread for which the output RCPL is to be built. For example, the first SLU generated after the TPF application is designated as thread number 1, the second SLU is thread number 2, etc.

17-19 = 3 spare bytes which must be set to zero.

R7: Address of an available core block which can be used by this program.

When requesting node names of the session partners the input is as follows:

Registers:

R1: Address of a 20 byte parameter list in the following format:

Byte 0-15 An input RCPL (RC0PL) associated with a message received on the host node SLU-PLU session.

16-19 Spare bytes. Not used.

R7: Address of an available core block which can be used by this program.

Expected output from CSF0/CSF1 includes:

1. When requesting an output RCPL to be build:

Registers:

R0: Same as on input

R1: Same as on input except the data area contains an output RCPL which has the following format (refer to RC0PL):

RCPLDES3 (3 bytes) - This is the 3-byte resource identifier of the selected SLU.

RCPLDESP (1 byte) - Set to the CPU ID of this TPF host.

RCPLORGA (4 bytes) - The four character name of the TPF application.

RCPLCTL0 (1 byte) - Equals X'42'.

RCPLCTL1 (1 byte) - Equals X'00'.

RCPLCTL2 (1 byte) - Equals X'00'.

RCPLCTL3 (1 byte) - Equals X'00'.

**Note:** The application must set the appropriate indicators in RCPLCTL0 (for RCPL0FMT) and RCPLCTL2

before issuing the ROUTC macro.

R2-R7: Same as on input

#### Entry Control Block Work Area

EBCM01 Return Code  
= X'00' The request was performed successfully  
= X'04' Invalid node name specified  
= X'08' Invalid PLU/SLU pair  
- SLU not in session with PLU  
- not a PLU/SLU combination  
= X'0C' RIDCC Error  
= X'10' SLU not bound/not in-session  
= X'24' Invalid Request  
- not cross-domain environment  
- not LU  
= X'20' SLU not available  
= X'24' Discrepancy between input qualifier and input  
SLU  
= X'28' Qualifier invalid  
- does not point to valid SLU

## 2. When requesting the node name of the session pair

#### Registers:

R0 - Same as on input

R1 - Same as on input except the data area now contains the following:

Byte	0-7	Node name of the destination (left justified and padded with blanks).
	8-15	Node name of the origin.
	16	Equals the thread number of the SLU receiving the message.

R2 -R7 - Same as on input.

---

## Appendix D. Using the Path Information Unit (PIU) Trace Facility

The path information unit (PIU) trace facility provides a detailed trace of the data transferred between the TPF system and remote resources. Each time the TPF system sends or receives data, some, all, or none of that data is stored in an entry in the PIU trace table depending on the resources that you are tracing. Later, you can display the entries in the PIU trace table online, or write the PIU trace table to a real-time tape and create a PIU print (PIUPRT) report to view or print offline.

The data stored in a PIU trace table entry can be a PIU in FID2 or FID4 format, a network layer packet (NLP), or an 8-byte channel-to-channel (CTC) header (which is included in all read and write channel programs during normal CTC data transfer operations). You can trace the data for all of the resources in the network or for only specific resources in the network. You can also trace only particular types of data, such as data for high-performance routing (HPR) traffic or data for network control (NC) commands and virtual route (VR) pacing requests and responses.

---

### About the PIU Trace Table

The PIU trace table resides in core storage in the TPF system. The size of the PIU trace table is defined in keypoint 2 (CTK2) using the TRACSZ parameter in the SNAKEY macro. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro.

The number of entries in the PIU trace table varies depending on the size of the PIU trace table and how many bytes of the request/response unit (RU) are being stored in the PIU trace table.

The PIU trace table operates in wraparound mode; that is, once all of the entries in the PIU trace table are used, the TPF system begins overwriting the oldest entries with the new data.

To store data in the PIU trace table, you must first start the PIU trace facility and specify which data you want to trace. When you no longer want to trace data, you can stop the PIU trace facility. At any time, you can display the status of the PIU trace facility and the type of data you are tracing.

---

### Starting the PIU Trace Facility and Specifying Which Data to Trace

Use the ZNTRP command to start the PIU trace facility and specify which data to trace. The TPF system stores the data being traced in entries in the PIU trace table. The data stored in a PIU trace table entry can be a PIU in FID2 or FID4 format, an NLP, or an 8-byte CTC header (which is included in all read and write channel programs during normal CTC data transfer operations).

You can use the PIU trace facility to trace the following:

- All the data transferred between the TPF system and remote resources
- Only data for a specific resource, such as a logical unit (LU), adjacent link station (ALS), network control program (NCP), cross-domain resource manager (CDRM), channel-to channel (CTC) link, or system services control point (SSCP)
- Only HPR traffic, which includes ROUTE\_SETUP commands and all NLPs on all rapid transport protocol (RTP) connections

- Only data flowing over a specific RTP connection
- Only data flowing for HPR state changes, which includes HPR ROUTE\_SETUP commands, RTP connections starting, RTP connections stopping, and path switches
- Only network control (NC) commands, virtual route (VR) pacing requests, and VR pacing responses flowing for NCPs or CTC links.

You can start more than one trace at the same time; for example, you can trace the data for LU X and LU Y. Depending on how you define your traces, some traces may overlap other traces. For example, if you start tracing the data for LU X and then later start tracing the RTP connection being used by LU X, the trace defined for the RTP connection will overlap the trace defined for LU X. In addition, if you have traces defined for specific resources and then later start tracing all resources, the trace defined for all resources will overlap the traces defined for specific resources.

At any time, you can display information about the status of the PIU trace facility and the data being traced. See “Displaying Information about the PIU Trace Facility” on page 294 for more information.

To start the PIU trace facility, do the following:

1. Enter the ZNTRP command with the START parameter specified to start the PIU trace facility and specify which data you want to trace.
2. Enter the ZNTRP command with the RUSZ or CRUSZ parameter specified to specify how many bytes of the request/response unit (RU) to store in the PIU trace table. See “Defining How Much of the RU to Store in the PIU Trace Table” on page 293 for more information.
3. Enter **ZNTRP START TAPE** to start writing the PIU trace table to a real-time tape. See “Writing the PIU Trace Table to a Real-Time Tape” on page 293 for more information.

---

## Stopping the PIU Trace Facility

Use the ZNTRP command to stop the PIU trace facility. You can stop the PIU trace facility completely or you can stop only particular traces that you defined. See “Starting the PIU Trace Facility and Specifying Which Data to Trace” on page 291 for more information about defining traces.

To stop the PIU trace facility completely, enter **ZNTRP STOP ALLR**. All of the traces that you defined will end and data will no longer be stored in the PIU trace table.

To stop tracing all resources in the TPF system, but to continue the traces that you defined for specific resources, enter **ZNTRP STOP ALL**. For example, assume you were tracing data for LU X and then started tracing all resources in the TPF system. After you enter ZNTRP STOP ALL, the PIU trace facility will continue tracing data for only LU X.

To stop tracing HPR traffic or HPR state changes, enter **ZNTRP STOP HPR**.

To stop a trace that was defined for a specific resource, enter the ZNTRP command with the STOP parameter and the NETID, ID, or RTP parameters specified. If the trace that you stopped overlapped with another trace, the PIU trace facility will continue that other trace. For example, assume you were tracing LU X and then



started tracing the NCP being used by LU X. If you stop the trace for the NCP, the PIU trace facility will continue tracing LU X.

---

## Defining How Much of the RU to Store in the PIU Trace Table

The PIU trace facility automatically stores (or traces) 21 bytes of the RU in the PIU trace table. Use the ZNTRP command to increase or decrease how much of the RU is traced. You can define how much of the RU to trace for user data messages. You can also define how much of the RU to trace for SNA commands, data that flows on CDRM-CDRM sessions, and data that flows on CP-CP sessions.

To define how much of the RU to trace for user data messages, enter the ZNTRP command with the RUSZ parameter specified.

To define how much of the RU to trace for SNA commands, data that flows on CDRM-CDRM sessions, and data that flows on CP-CP sessions, enter the ZNTRP command with the CRUSZ parameter specified.

See *TPF Operations* for more information about the ZNTRP command.

---

## Writing the PIU Trace Table to a Real-Time Tape

Before you can create a PIUPRT report to view or print offline, you must write the PIU trace table to a real-time tape. See “Using the Offline PIU Print (PIUPRT) Utility to Create a PIUPRT Report” on page 301 for more information about creating a PIUPRT report.

To write the PIU trace table to a real-time tape, do the following:

1. Ensure you have a real-time tape mounted for the TPF system. See *TPF Operations* for more information about mounting a tape.
2. Enter the ZNTRP command with the START parameter specified to start the PIU trace facility and specify which data you want to trace. See “Starting the PIU Trace Facility and Specifying Which Data to Trace” on page 291 for more information.
3. Enter **ZNTRP START TAPE** to start writing the PIU trace table to the real-time tape.

The PIU trace facility will automatically write each 4-KB block of the PIU trace table to the real-time tape when that 4-KB block becomes full. If the length of the queue to the real-time tape becomes too large, the PIU trace facility stops writing the PIU trace table to tape until the queue becomes smaller. The maximum tape queue length is defined in keypoint 2 (CTK2) using the PIUTAPEQ parameter in the SNAKEY macro. See *TPF ACF/SNA Network Generation* for more information about the SNAKEY macro.

When you are ready to create a PIUPRT report, stop writing the PIU trace table to the real-time tape. To stop writing the PIU trace table to the real-time tape, do the following:

1. Enter **ZNTRP STOP TAPE**. The PIU trace facility will write the current 4-KB block of the PIU trace table to the real-time tape regardless of whether the current 4-KB block is full.
2. Remove the real-time tape so that it can be processed by the PIUPRT utility. See *TPF Operations* for more information about removing a real-time tape.

See *TPF Operations* for more information about the ZNTRP command.

---

## Displaying Information about the PIU Trace Facility

Use the ZNTRP command to display information about the PIU trace facility.

Enter **ZNTRP DISPLAY OPTIONS** to display the status of the PIU trace facility. The information displayed indicates the following:

- Whether the PIU trace table is being written to a real-time tape.
- How much of the RU is being traced for user data messages.
- How much of the RU is being traced for SNA commands, data that flows on CDRM-CDRM sessions, and data that flows on CP-CP sessions.

Enter **ZNTRP DISPLAY** to display information about the data being traced. The information displayed indicates whether you are tracing the data for all the resources in the network or for only specific resources. The information displayed also indicates whether you are tracing only a particular type of data.

See *TPF Operations* for more information about the ZNTRP command.

## Examples

The status of the PIU trace facility is displayed in the following example.

```
User:  ZNTRP DISPLAY OPTIONS

System: NTRP0045I 23.23.21 PIU TRACE OPTIONS INFORMATION
        PIU TRACE IS ACTIVE TO REAL TIME TAPE
        200 BYTES OF RU ARE BEING TRACED FOR USER DATA MESSAGES
        100 BYTES OF RU ARE BEING TRACED FOR SNA COMMANDS
        END OF DISPLAY
```

Data being transferred between the TPF system and all resources in the network is being traced in the following example.

```
User:  ZNTRP DISPLAY

System: NTRP0042I 16.28.22 PIU TRACE IS ACTIVE FOR ALL RESOURCES
```

Only network control (NC) commands and VR pacing requests and responses being transferred between the TPF system and the N42E720 resource are being traced in the following example.

```
User:  ZNTRP DISPLAY

System: NTRP0048I 12.31.25 PIU TRACE ACTIVE FOR THE FOLLOWING SELECTED RESOURCES
        NETID      NAME      RTP INDEX  DEVICE TYPE  VRONLY
        -----
                N42E720      -----  ALS/NCP/CTC   YES
        END OF DISPLAY
```

Only data transferred on the specified RTP connection is being traced in the following example.

```

User:  ZNTRP DISPLAY

System: NTRP0048I 12.31.25 PIU TRACE ACTIVE FOR THE FOLLOWING SELECTED RESOURCES
NETID      NAME      RTP INDEX  DEVICE TYPE  VRONLY
-----
                                000001    RTP CONNECTION
END OF DISPLAY

```

Only data transferred over sessions with the specified LUs is being traced in the following example.

```

User:  ZNTRP DISPLAY

System: NTRP0048I 12.31.25 PIU TRACE ACTIVE FOR THE FOLLOWING SELECTED RESOURCES
NETID      NAME      RTP INDEX  DEVICE TYPE  VRONLY
-----
                                G623      LU
                                RES0      LU
END OF DISPLAY

```

## Displaying the PIU Trace Table Online

Use the ZNPIU command to display the PIU trace table online. You can display all of the PIU trace table or only part of the PIU trace table. You can also display the HPR control messages in the PIU trace table and specify the format in which you want to display the PIU trace table.

Enter the ZNPIU command with the ALL parameter specified to display all of the PIU trace table entries. When you display all of the PIU trace table entries, the oldest entry (that is, the entry with the oldest time stamp) is displayed first, followed by the next oldest entry, and so on. Therefore, the last entry displayed is the newest entry in the PIU trace table.

You can also enter the ZNPIU command and specify the number of PIU trace table entries that you want to display. When you do this, the newest entries in the PIU trace table are displayed. For example, if you specify that you want to display only 11 entries from the PIU trace table, the 11 newest entries are displayed.

Keep in mind that the PIU trace facility can continue to store new data in the PIU trace table while you are displaying entries from the PIU trace table. Therefore, the newest entries continue to change as more data is stored in the PIU trace table. For example, if you display 11 entries from the PIU trace table, wait a few seconds while data continues to flow and then display 11 entries from the PIU trace table again, the 11 entries displayed the second time are different from the 11 entries displayed the first time.

Enter the ZNPIU command with the CONTROL parameter specified to include HPR control messages in the information that is displayed. An *HPR control message* is an NLP that contains a network layer header (NHDR) and transport header (THDR), but no data. If you do not specify the CONTROL parameter, the HPR control messages are not displayed.

When you enter the ZNPIU command to display the PIU trace table, the last line of the information displayed tells you how many entries are in the PIU trace table. This number includes HPR control messages. Remember that HPR control messages are included in the information displayed only when you specify the CONTROL parameter. Therefore, if the PIU trace table contains only HPR control messages

and you enter the ZNPIU command without the CONTROL parameter, no entries are displayed. However, the number of entries in the PIU trace table is displayed.

There are two formats in which you can display the PIU trace table. You can create a compacted display of the PIU trace table or a formatted display of the PIU trace table.

## Creating a Compacted Display of the PIU Trace Table

To create a compacted display of the PIU trace table, enter the ZNPIU command with the COMPACT parameter specified; also specify the CONTROL parameter if you want to include HPR control messages in the information that is displayed. See *TPF Operations* for more information about the ZNPIU command.

In the compacted display of the PIU trace table, each PIU trace table entry is displayed on a separate line and only part of the RU being traced is included. Different information is displayed depending on whether the PIU trace table entry is for an FID4 PIU, FID2 PIU, NLP, or 8-byte CTC header.

- For an FID4 PIU, the following information is displayed:

<b>RW</b>	Read/write operation code
<b>IN</b>	Channel command word (CCW) area index
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>DNA</b>	Destination network address
<b>ONA</b>	Origin network address
<b>VRSQ</b>	Virtual route sequence number
<b>SEQ</b>	Sequence number of the session
<b>CNT</b>	Combined length of the request/response header (RH) and the RU
<b>RH</b>	Request/response header
<b>RU</b>	Request unit.

- For an FID2 PIU, the following information is displayed:

<b>RW</b>	Read/write operation code
<b>IN</b>	Channel command word (CCW) area index
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>LNKHDR</b>	Link header
<b>TH</b>	Bytes 0 and 1 of the FID2 transmission header (TH)
<b>SID</b>	Session identifier
<b>SEQ</b>	Sequence number of the session
<b>RH</b>	Request/response header
<b>RU</b>	Request unit.

- For an NLP, the following information is displayed:

<b>RW</b>	Read/write operation code
-----------	---------------------------

<b>IN</b>	Channel command word (CCW) area index
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>RTP</b>	Rapid transport protocol control block (RTPCB) index
<b>PCID</b>	Procedure correlation identifier
<b>SEQ</b>	Sequence number of the session
<b>RH</b>	Request/response header
<b>RU</b>	Request unit.
• For an 8-byte CTC header, the following information is displayed:	
<b>RW</b>	Read/write operation code
<b>IN</b>	Channel command word (CCW) area index
<b>RU</b>	8-byte CTC header.

## Examples

Figure 95 shows an example of a compacted display of the PIU trace table. The header in the information shown contains three lines. The first line describes the information displayed for an FID4 PIU. The second line describes the information displayed for an FID2 PIU. The third line describes the information displayed for an NLP.

The example shows four entries in the PIU trace table. The entries are displayed in the following order:

- FID4 PIU
- FID2 PIU
- NLP
- 8-byte CTC header.

```
User:  ZNPIU COMPACT 4

System: NPIU0004I 17.36.50 PIU TRACE TABLE
RW IN DRID ORID DNA ONA VRSQ SEQ CNT RH RU
RW IN DRID ORID LNKHDR TH SID SEQ RH RU
RW IN DRID ORID RTP PCID SEQ RH RU
32 02 000002 00000F 0B0000 1F0000 0000 0000 0018 2B0000 0F00000010000000001
52 01 000070 000136 00100000 2F00 0101 0000 830100 000001
31 01 000204 00023E 000001 E383BE95283C9EC8 0001 EB8000 A1
05 0B 003F0001055E0544
35 PIUS IN TRACE TABLE
```

Figure 95. Compacted Display of the PIU Trace Table

See *IBM Systems Network Architecture Network Product Formats* for more information about PIU and NLP formats.

## Creating a Formatted Display of the PIU Trace Table

To create a formatted display of the PIU trace table, enter the ZNPIU command with the FORMAT or LONG parameter specified; also specify the CONTROL parameter if you want to display the HPR control messages.

For the formatted display of the PIU trace table, you can also specify the HEADERS parameter to display the NHDR and THDR for the NLPs in the PIU trace table. If you do not specify the HEADERS parameter, the NHDR and THDR are not displayed. Be sure to specify the HEADERS parameter when you specify the CONTROL parameter (because HPR control messages contain only an NHDR and a THDR).

You can also specify the NODATA parameter if you do not want to display the RU for PIUs and NLPs. This can be useful when you want to view control data at the RTP connection level. If you do not specify the NODATA parameter, the RU is displayed.

See *TPF Operations* for more information about the ZNPIU command.

In the formatted display of the PIU trace table, each PIU trace table entry is formatted and the entire RU being traced is displayed (if you did not specify the NODATA parameter). In addition, the RH indicators and THDR optional segments (for NLPs) are translated and displayed for you.

Different information is displayed depending on whether the PIU trace table entry is for an FID4 PIU, FID2 PIU, NLP, or 8-byte CTC header.

- For an FID4 PIU, the following information is displayed:

<b>RWI</b>	Read/write operation code
<b>CCW</b>	Channel command word (CCW) area index
<b>DNAME</b>	Destination name
<b>ONAME</b>	Origin name
<b>PCID</b>	Procedure correlation identifier
<b>TIME</b>	Time stamp
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>DNA</b>	Destination network address
<b>ONA</b>	Origin network address
<b>ERV</b>	Explicit route, virtual route, and transmission priority
<b>VRSEQ</b>	Virtual route sequence number
<b>SEQ</b>	Sequence number of the session
<b>CNT</b>	Combined length of the request/response header (RH) and the RU
<b>RH</b>	Request/response header
<b>RU</b>	Request unit
<b>RH INDICS</b>	Translated RH indicators. See "RH Indicators" on page 312 for more information.

- For an FID2 PIU, the following information is displayed:

<b>RWI</b>	Read/write operation code
<b>CCW</b>	Channel command word (CCW) area index
<b>DNAME</b>	Destination name

<b>ONAME</b>	Origin name
<b>PCID</b>	Procedure correlation identifier
<b>TIME</b>	Time stamp
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>LNKHDR</b>	Link header
<b>TH</b>	Bytes 0 and 1 of the FID2 transmission header (TH)
<b>SID</b>	Session identifier
<b>SEQ</b>	Sequence number of the session
<b>RH</b>	Request/response header
<b>RU</b>	Request unit.
<b>RH INDICS</b>	Translated RH indicators. See “RH Indicators” on page 312 for more information.
• For an NLP, the following information is displayed:	
<b>RWI</b>	Read/write operation code
<b>CCW</b>	Channel command word (CCW) area index
<b>DNAME</b>	Destination name
<b>ONAME</b>	Origin name
<b>PCID</b>	Procedure correlation identifier
<b>TIME</b>	Time stamp
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>LNKHDR</b>	Link header
<b>TCID1</b>	Transport connection identifier (TCID) that the TPF system assigned to the RTP connection
<b>TCID2</b>	TCID that the remote RTP endpoint assigned to the RTP connection
<b>SEQ</b>	Sequence number of the session
<b>RTP</b>	RTPCB index
<b>SA1</b>	Session address (SA) that the TPF system assigned to the LU-LU session
<b>SA2</b>	SA that the remote RTP endpoint assigned to the LU-LU session
<b>BSN</b>	Byte sequence number (BSN) of the RTP connection
<b>NHDR</b>	Network layer header (NHDR)
<b>THDR</b>	Transport header
<b>SEGMENTS</b>	Indicates that the following optional segments in the THDR are present:
<b>ARB</b>	Adaptive Rate-Based segment (X'22')
<b>CF</b>	Connection Fault segment (X'12')

<b>CIE</b>	Connection Identifier Exchange segment (X'10')
<b>COB</b>	Client Out of Band segment (X'0F')
<b>CS</b>	Connection Setup segment (X'0D')
<b>SI</b>	Switching Information segment (X'14')
<b>STATUS</b>	Status segment (X'0E')

**FID5** FID5 transmission header (TH)

**RH** Request/response header

**RU** Request unit

**RH INDICS** Translated RH indicators. See "RH Indicators" on page 312 for more information.

- For an 8-byte CTC header, the following information is displayed:

**RWI** Read/write operation code

**CCW** Channel command word (CCW) area index

**CTC HDR** 8-byte CTC header.

### Examples

Figure 96 on page 301 shows an example of a formatted display of the PIU trace table. The entries are displayed in the following order:

- FID4 PIU
- FID2 PIU
- NLP
- 8-byte CTC header.



```

User:  ZNPIU FORMAT 4 HEADERS

System: NPIU0005I 09.52.59 PIU TRACE TABLE
RWI-06 CCW-0B DNAME- TPFA ONAME- TPFB
PCID-0000000000000000 TIME-54.24 DRID-000001 ORID-000002
DNA-0A0000 ONA-0B0001 ERVR-0001 VRSQ-0005 SEQ-0001 CNT-0020 RH-EB8000
RU-ACTCDRM RH INDICS- RSP SC FI PRSP
  0  0  14021111 40404040 40404040 05000000 .....
 16 10  000B3F06 0600274E B40020FE 00 .....
RWI-51 CCW-01 DNAME- VTAMNET.VTAM2 ONAME- APPC
PCID-C43E70C65E829A43 TIME-53.56 DRID-800002 ORID-000102
LNKHDR-00FE0000 TH-2C00 SID-0101 SEQ-0023 RH-0B9081
RU- RH INDICS- REQ FMD FI OIC ER BB CEB
  0  0  0E0502FF 0003D000 000422F0 F0F30027 .....003..
 16 10 12C48000 00000018 60E383BE 953FE66D .D.....-T....W.
 32 20 F40DE5E3 C1D4D5C5 E34BE5E3 C1D4F206 4.VTAMNE T.VTAM2.
 48 30 81000201 31004712 CA0580C1 0000103C .....A....
 64 40 00F4E5E3 C1D4D5C5 E34BC1D7 D7C3103D .4VTAMNE T.APPC..
 80 50 00F3E5E3 C1D4D5C5 E34BC7D4 E3F10326 .3VTAMNE T.GMT1..
 96 60 DA093E07 8080FFFF FFFF1282 00F3E5E3 .....3VT
112 70 C1D4D5C5 E34BD3C3 D3F8C4C4 007512C5 AMNET.LC L8DD...E
128 80 00040180 000008E2 F3F2F7F0 40404000 .....S 3270....
144 90 000C2C01 087BC3D6 D5D5C5C3 E3144612 .....CO NNECT...
160 A0 80150DE5 E3C1D4D5 C5E34BE5 E3C1D4F2 ...VTAMN ET.VTAM2
176 B0 21164700 00000098 2D000000 00000000 .....
192 C0 00017100 00000014 46128016 0DE5E3C1 .....VTA
208 D0 D4D5C5E3 4BE5E3C1 D4F22116 47000000 MNET.VTA M2.....
224 E0 00882D00 00000000 00000001 71000000 .....
240 F0 00 .....
RWI-32 CCW-01 DNAME- GMT1 ONAME- VTAMNET.LCL8DD
PCID-E383BE953FE66DF4 TIME-54.00 DRID-000204 ORID-00023E
LNKHDR-009D0000 TCID1-2D105A9AC4000008 TCID2-020DFB2D00000317 SEQ-023E
RTP-000008 SA1-30C480EEBD5EB606 SA2-0000000000000000 BSN-00000000
NHDR
  0  0  C208DAFF 00 B....
THDR SEGMENTS- CIE ARB
  0  0  2D105A9A C4000008 3804000B 00000068 ....D...
 16 10  00000000 03228000 000A8F20 00000000 .....
 32 20  03100000 820DFB2D 00000317 .....
FID5-5D00023E30C480EEBD5EB606 RH-EB8000
RU-BIND RH INDICS- RSP SC FI PRSP
  0  0  31010000 00008002 0000B700 80000000 .....
 16 10  00000000 00000000 00000000 00000060 .....
 32 20 16E383BE 953FE66D F40DE5E3 C1D4D5C5 .T....W. 4.VTAMNE
 48 30 E34BE5E3 C1D4F262 08800000 00060000 T.VTAM2. ....
 64 40 032B1601 01144612 80150DE5 E3C1D4D5 .....VTAMN
 80 50 C5E34BE5 E3C1D4F2 21 ET.VTAM2 .
RWI-06 CCW-0B CTC HDR-003A00010205024D
426 PIUS IN TRACE TABLE

```

Figure 96. Formatted Display of the PIU Trace Table

See *IBM Systems Network Architecture Network Product Formats* for more information about PIU and NLP formats.

## Using the Offline PIU Print (PIUPRT) Utility to Create a PIUPRT Report

Use the offline PIUPRT utility to create a PIUPRT report, which you can view or print offline. Unlike using the ZNPIU command to display the PIU trace table online, the PIUPRT utility offers you more flexibility to change the information contained in the PIUPRT report. For example, you can print in the PIUPRT report only the data transferred between the TPF system and a specific remote resource, or only the data that flowed over a specific RTP connection. See “Defining the PIUPRT Report” on page 303 for more information.

Another difference between creating a PIUPRT report and using the ZNPIU command to display the PIU trace table online is that you create a PIUPRT report from the PIU trace table on a real-time tape rather than in core storage. If you enter the ZNPIU command to display the PIU trace table online while you are tracing active resources, the oldest entries in the PIU trace table may be written over with new data. Therefore, after this happens, you cannot display those entries online. However, because you create a PIUPRT report from the PIU trace table on a real-time tape, you never have this problem.

The PIUPRT utility runs on an MVS system. Before you can use the PIUPRT utility to create a PIUPRT report, you must do the following:

1. Compile the PIUPRT utility
2. Submit the object code to the object library
3. Link the object code to the link library.

To create a PIUPRT report, do the following:

1. Follow the steps in “Starting the PIU Trace Facility and Specifying Which Data to Trace” on page 291 to start the PIU trace facility and specify which data you want to trace.
2. When you are ready to create a PIUPRT report, perform a tape switch for the real-time tape. See *TPF Operations* for more information about performing a tape switch.
3. Create the job control language (JCL) needed to run the PIUPRT utility. See “Sample JCL for the PIUPRT Utility” for an example.
4. Define the PIUPRT report by updating the PARM= statement in the PIUPRT JCL. This allows you to specify the format of the PIUPRT report and the data you want to include in it. See “Defining the PIUPRT Report” on page 303 for more information.
5. Submit the PIUPRT JCL to the MVS system to run the PIUPRT utility and create the PIUPRT report. See “PIUPRT Utility Return Codes” on page 312 for information.
6. View or print the PIUPRT report.

## Sample JCL for the PIUPRT Utility

Figure 97 is an example of the JCL that you can use to run the PIUPRT utility. Change the tape number, shown as XXXXXX, to the tape number for the real-time tape that contains the PIU trace table. Change the link library name, shown as *ACP.DEVP.TEST.LK*, to the name of your link library.

```
//PIU EXEC PGM=PIUPRT,PARM='FORMAT ALL'
//STEPLIB DD DISP=SHR,DSN=ACP.DEVP.TEST.LK
//PRINT DD SYSOUT=A,DCB=(LRECL=133,BLKSIZE=3990,RECFM=FBA)
//RTL DD DSN=RTL,DCB=(LRECL=4095,BLKSIZE=32760,RECFM=U),
// DISP=OLD,LABEL=(2,BLP),UNIT=TAPE,VOL=SER=XXXXXX
//SYSUDUMP DD SYSOUT=A
/*
/* RECFM=VB FOR TAPES CREATED IN BLOCKED FORMAT.
```

Figure 97. JCL for the PIUPRT Utility

## Defining the PIUPRT Report

Use the PARM= statement in the PIUPRT JCL to define the PIUPRT report. You can specify the data that you want to print in the PIUPRT report as well as how you want to format the PIUPRT report.

Unlike using the ZNPIU command to display a specific number of entries from the PIU trace table, you can actually define the type of data that you want to print in a PIUPRT report. For example, you can print the entire PIU trace table on the real-time tape or you can print only the data that flowed over a particular RTP connection. You can also print only the data for a particular LU-LU session or print only the data that has a time stamp in a specified range.

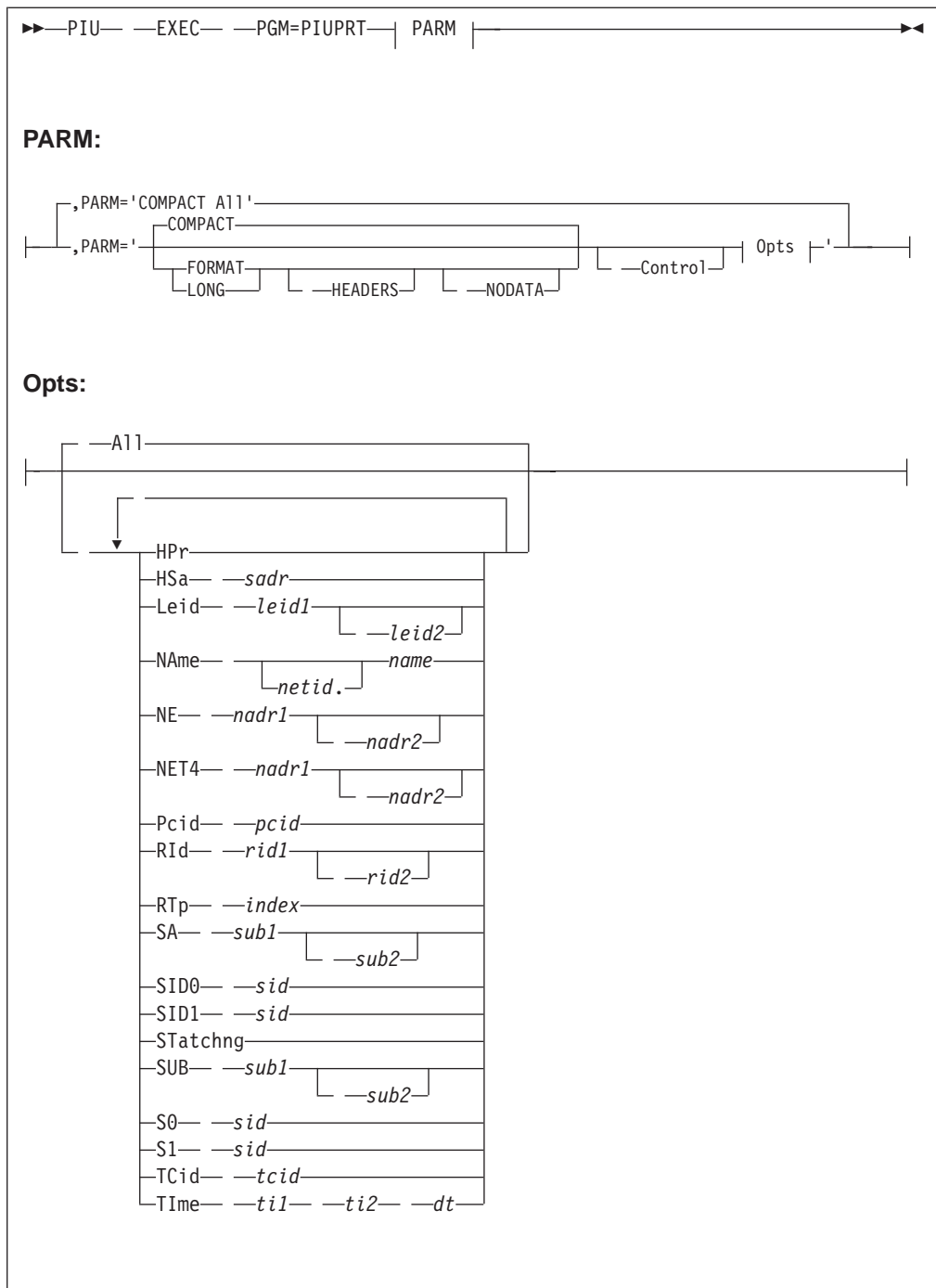
Specify the CONTROL parameter for the PARM= statement to include HPR control messages in the PIUPRT report. An *HPR control message* is an NLP that contains an NHDR and THDR, but no data. If you do not specify the CONTROL parameter, the HPR control messages are not printed in the PIUPRT report.

Depending on the data contained in the PIU trace table on the real-time tape and the values that you specify for the PARM= statement in the PIUPRT JCL, a PIUPRT report can contain PIUs in FID2 or FID4 format, NLPs, and 8-byte CTC headers (which are included in all read and write channel programs during normal CTC data transfer operations). There are also two formats in which you can create the PIUPRT report. Use the PARM= statement to specify whether you want to create a compacted PIUPRT report or a formatted PIUPRT report.

See “Sample Compacted PIUPRT Report” on page 307 and “Sample Formatted PIUPRT Report” on page 309 for an example of a compacted PIUPRT report and a formatted PIUPRT report.

### **PARM= Statement for the PIUPRT JCL**

Many parameters are available for the PARM= statement in the PIUPRT JCL that allow you to change the contents of the PIUPRT report to your specific needs. The following information shows the syntax for the PARM= statement and describes the parameters.



### All

includes in the PIUPRT report all of the path information units (PIUs) and network layer packets (NLPs), except for HPR control messages, in the PIU trace table.

### COMPACT

creates a compacted PIUPRT report in which each entry is printed on a single line. See "Sample Compacted PIUPRT Report" on page 307 for an example of a compacted PIUPRT report.

**Control**

includes the HPR control messages in the PIUPRT report. If you do not specify this parameter, the HPR control messages are not included in the PIUPRT report.

**FORMAT**

creates a formatted PIUPRT report. See “Sample Formatted PIUPRT Report” on page 309 for an example of a formatted PIUPRT report.

**HEADERS**

prints the network layer header (NHDR) and transport header (THDR) for each HPR network layer packet (NLP) included in the PIUPRT report. If you do not specify this parameter, the NHDR and THDR are not printed for the NLPs included in the PIUPRT report.

**HP**

includes only HPR traffic (NLPs and ROUTE\_SETUP commands) in the PIUPRT report.

**HSa** *sadr*

includes in the PIUPRT report only the NLPs for the specified HPR LU-LU session, where *sadr* is the 16-digit hexadecimal session address of the HPR LU-LU session.

**Leid** *leid1 leid2*

includes in the PIUPRT report only the PIUs for the specified origin or destination network extension facility (NEF) logical endpoint identifier (LEID) or range of LEIDs, where *leid1* and *leid2* are 6-digit hexadecimal LEIDs.

**LONG**

creates a formatted PIUPRT report. See “Sample Formatted PIUPRT Report” on page 309 for an example of a formatted PIUPRT report.

**Note:** This parameter is the same as the FORMAT parameter.

**Name** *netid.name*

includes in the PIUPRT report only the PIUs and NLPs for the specified resource, where *netid* is the 1- to 8-character network identifier and *name* is the 1- to 8-character resource name. The network ID and name of a resource must both begin with a letter (A–Z), @, #, or \$. The remaining characters can be letters (A–Z), numbers (0–9), @, #, or \$.

Use the asterisk (\*) as a wildcard character to specify a group of network identifiers or resource names that begin or end with a common string of characters. For example, to include in the PIUPRT report all of the resources that have a name beginning with T46, specify **T46\***.

**NE** *nadr1 nadr2*

includes in the PIUPRT report only the PIUs for the specified origin or destination network address or range of network addresses, where *nadr1* and *nadr2* are 6-digit hexadecimal network addresses.

**NET4** *nadr1 nadr2*

includes in the PIUPRT report only the PIUs for the specified origin or destination network address or range of network addresses, where *nadr1* and *nadr2* are 6-digit hexadecimal network addresses.

**NODATA**

omits the RU from the PIU trace table entries contained in the PIUPRT report. If you do not specify this parameter, the RU for each PIU trace table entry contained in the PIUPRT report is printed.

**Pcid** *pcid*

includes in the PIUPRT report only the PIUs and NLPs for the specified procedure correlation identifier (PCID), where *pcid* is the 16-digit hexadecimal PCID.

**Rid** *rid1 rid2*

includes in the PIUPRT report only the PIUs and NLPs for the specified origin or destination resource identifier (RID) or range of RIDs, where *rid1* and *rid2* are 6-digit hexadecimal RIDs.

**RTP** *index*

includes in the PIUPRT report only the NLPs for the specified RTP connection, where *index* is the 6-digit hexadecimal RTPCB index of the RTP connection.

**SA** *sub1 sub2*

includes in the PIUPRT report only the PIUs for the specified origin or destination subarea or range of subareas, where *sub1* and *sub2* are 4-digit hexadecimal subareas.

**SID0** *sid*

includes in the PIUPRT report only the PIUs for a specific PU 2.1 LU-LU session, where *sid* is the 4-digit hexadecimal TPF-assigned session identifier (ID).

**SID1** *sid*

includes in the PIUPRT report only the PIUs for a specific PU 2.1 LU-LU session, where *sid* is the 4-digit hexadecimal NCP-assigned session ID.

**STatchng**

includes in the PIUPRT report only ROUTE\_SETUP commands and NLPs for RTP connections that are starting, ending, or performing a path switch.

**SUB** *sub1 sub2*

includes in the PIUPRT report only the PIUs for the specified origin or destination subarea or range of subareas, where *sub1* and *sub2* are 4-digit hexadecimal subareas.

**S0** *sid*

includes in the PIUPRT report only the PIUs for a specific PU 2.1 LU-LU session, where *sid* is the 4-digit hexadecimal TPF-assigned session ID.

**S1** *sid*

includes in the PIUPRT report only the PIUs for a specific PU 2.1 LU-LU session, where *sid* is the 4-digit hexadecimal NCP-assigned session ID.

**TCid** *tcid*

includes in the PIUPRT report only NLPs for the specified transport connection identifier (TCID), where *tcid* is the 16-digit hexadecimal TCID.

**Time** *ti1 ti2 dt*

includes in the PIUPRT report only the PIUs and NLPs in the specified time-stamp range, where *ti1* and *ti2* are the beginning and ending times in the format *hh.mm.ss*, and *dt* is the date in the format *ddmmm*; for example 11:45:00 12:00:00 15DEC.

**Sample PARM= Statements for the PIUPRT JCL**

A compacted PIUPRT report of all the data in the PIU trace table, including HPR control messages, is created in the following example.

```
//PIU EXEC PGM=PIUPRT,PARM='CONTROL'
```

A compacted PIUPRT report is created in the following example. The report includes all data transferred between the TPF system and remote resources that have a name beginning with T46.

```
//PIU EXEC PGM=PIUPRT,PARM='NA T46*'
```

A formatted PIUPRT report of all HPR traffic is created in the following example. HPR control messages and the NHDR and THDR for each NLP are included in the PIUPRT report. However, no RU data is included in the PIUPRT report.

```
//PIU EXEC PGM=PIUPRT,PARM='FORMAT HEADERS NODATA CONTROL HPR'
```

A compacted PIUPRT report is created in the following example. The report includes all of the PIUs and NLPs in the PIU trace table that have an origin or destination RID in the range 000001–000100 and a time stamp in the range 9:30–9:40 on August 14.

```
//PIU EXEC PGM=PIUPRT,PARM='RID 000001 000100 TIME 09:30:00 09:40:00 14AUG'
```

### Sample Compacted PIUPRT Report

To create a compacted PIUPRT report, specify the COMPACT parameter in the PARM= statement of the PIUPRT JCL; also specify the CONTROL parameter if you want to print the HPR control messages in the PIUPRT report. See “Defining the PIUPRT Report” on page 303 for more information.

In a compacted PIUPRT report, each PIU, NLP, or CTC header is printed on a single line and only part of the RU that was traced is included. Different information is printed in the compacted PIUPRT report depending on whether you are printing FID2 PIUs, FID4 PIUs, NLPs, or CTC headers.

- For an FID4 PIU, the following information is printed:

<b>RW</b>	Read/write operation code
<b>IN</b>	Channel command word (CCW) area index
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>DNA</b>	Destination network address
<b>ONA</b>	Origin network address
<b>VRSQ</b>	Virtual route sequence number
<b>SEQ</b>	Sequence number of the session
<b>CNT</b>	Combined length of the request/response header (RH) and the RU
<b>RH</b>	Request/response header
<b>RU</b>	Request unit.

- For an FID2 PIU, the following information is printed:

<b>RW</b>	Read/write operation code
<b>IN</b>	Channel command word (CCW) area index
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>LNKHDR</b>	Link header
<b>TH</b>	Bytes 0 and 1 of the FID2 transmission header (TH)

- |            |                                |
|------------|--------------------------------|
| <b>SID</b> | Session identifier             |
| <b>SEQ</b> | Sequence number of the session |
| <b>RH</b>  | Request/response header        |
| <b>RU</b>  | Request unit.                  |
- For an NLP, the following information is printed:

<b>RW</b>	Read/write operation code
<b>IN</b>	Channel command word (CCW) area index
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>RTP</b>	RTPCB index
<b>PCID</b>	Procedure correlation identifier
<b>SEQ</b>	Sequence number of the session
<b>RH</b>	Request/response header
<b>RU</b>	Request unit.
  - For an 8-byte CTC header, the following information is printed:

<b>RW</b>	Read/write operation code
<b>IN</b>	Channel command word (CCW) area index
<b>RU</b>	8-byte CTC header.

Figure 98 shows an example of a compacted PIUPRT report. The header in the PIUPRT report contains three lines. The first line describes the information printed for an FID4 PIU. The second line describes the information printed for an FID2 PIU. The third line displayed describes the information printed for an NLP.

The example prints four entries in the PIU trace table in the following order:

- FID4 PIU
- FID2 PIU
- NLP
- 8-byte CTC header.

```

*****
TRANSACTION PROCESSING FACILITY SNA PIU TRACE OUTPUT
*****
A DISPLAY OF PIUS BASED ON THE FOLLOWING USER PARAMETERS WILL BE PERFORMED:
COMPACT ALL

TRANSACTION PROCESSING FACILITY PIU TRACE REPORT
RW IN DRID ORID DNA ONA VRSQ SEQ CNT RH RU
RW IN DRID ORID LNKHDR TH SID SEQ RH RU
RW IN DRID ORID RTP PCID SEQ RH RU
*****
THESE PIUS WERE WRITTEN TO TAPE ON 18AUG AT 12.17.32 *****
32 01 000002 000010 0B0000 230000 0000 0000 0018 2B0000 0F00000100000000230000000B010100000023FF00
51 02 000174 000138 007B0000 2D00 0101 0174 6B8000 31010202004000020000A70080000000000000000000000000FE5E3C1D4D5C5E34BC7
31 02 00023E 00091A 000001 E383BE951E86713C 0001 039000 F5C31140401D4011C1C21DF011C1501D4011C2D21DF011C2601D4011C3E21DF011C3F01D40
05 0B 001D000198991015

```

Figure 98. Compacted PIUPRT Report

See *IBM Systems Network Architecture Network Product Formats* for more information about PIU and NLP formats.



## Sample Formatted PIUPRT Report

To create a formatted PIUPRT report, specify the **FORMAT** or **LONG** parameter in the **PARM=** statement of the PIUPRT JCL; also specify the **CONTROL** parameter if you want to include HPR control messages in the PIUPRT report.

For a formatted PIUPRT report, you can also specify the **HEADERS** parameter to print the **NHDR** and **THDR** for NLPs in the PIU trace table. If you do not specify the **HEADERS** parameter, the **NHDR** and **THDR** are not printed. Be sure to specify the **HEADERS** parameter when you specify the **CONTROL** parameter.

You can also specify the **NODATA** parameter if you do not want to print the **RU** for PIUs and NLPs in the PIUPRT report. This can be useful when you want to view control data at the RTP connection level. If you do not specify the **NODATA** parameter, the **RU** is printed.

In a formatted PIUPRT report, each PIU, NLP, or CTC header is formatted and the entire **RU** that was traced is printed (if you did not specify the **NODATA** parameter). In addition, the **RH** indicators and the **THDR** optional segments (for NLPs) are translated and printed in the PIUPRT report.

Different information is printed in the formatted PIUPRT report depending on whether you are including **FID2** PIUs, **FID4** PIUs, NLPs, or CTC headers.

- For an **FID4** PIU, the following information is printed:

<b>RWI</b>	Read/write operation code
<b>CCW</b>	Channel command word (CCW) area index
<b>DNAME</b>	Destination name
<b>ONAME</b>	Origin name
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>PCID</b>	Procedure correlation identifier
<b>TIME</b>	Time stamp
<b>DNA</b>	Destination network address
<b>ONA</b>	Origin network address
<b>ERV</b>	Explicit route, virtual route, and transmission priority
<b>VRSQ</b>	Virtual route sequence number
<b>SEQ</b>	Sequence number of the session
<b>CNT</b>	Combined length of the request/response header (RH) and the RU
<b>RH</b>	Request/response header
<b>RU</b>	Request unit
<b>RH INDICS</b>	Translated RH indicators. See "RH Indicators" on page 312 for more information.

- For an **FID2** PIU, the following information is printed:

<b>RWI</b>	Read/write operation code
<b>CCW</b>	Channel command word (CCW) area index

<b>DNAME</b>	Destination name
<b>ONAME</b>	Origin name
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>PCID</b>	Procedure correlation identifier
<b>TIME</b>	Time stamp
<b>LNKHDR</b>	Link header
<b>TH</b>	Bytes 0 and 1 of the FID2 transmission header (TH)
<b>SID</b>	Session identifier
<b>SEQ</b>	Sequence number of the session
<b>RH</b>	Request/response header
<b>RU</b>	Request unit.
<b>RH INDICS</b>	Translated RH indicators. See “RH Indicators” on page 312 for more information.

- For an NLP, the following information is printed:

<b>RWI</b>	Read/write operation code
<b>CCW</b>	Channel command word (CCW) area index
<b>DNAME</b>	Destination name
<b>ONAME</b>	Origin name
<b>DRID</b>	Destination resource identifier (RID)
<b>ORID</b>	Origin RID
<b>PCID</b>	Procedure correlation identifier
<b>TIME</b>	Time stamp
<b>LNKHDR</b>	Link header
<b>TCID1</b>	Transport connection identifier (TCID) that the TPF system assigned to the RTP connection
<b>TCID2</b>	TCID that the remote RTP endpoint assigned to the RTP connection
<b>SEQ</b>	Sequence number of the session
<b>RTP</b>	RTPCB index
<b>SA1</b>	Session address (SA) that the TPF system assigned to the LU-LU session
<b>SA2</b>	SA that the remote RTP endpoint assigned to the LU-LU session
<b>BSN</b>	Byte sequence number (BSN) of the RTP connection
<b>NHDR</b>	Network layer header (NHDR)
<b>THDR</b>	Transport header
<b>SEGMENTS</b>	Indicates that the following optional segments in the THDR are present:
<b>ARB</b>	Adaptive Rate-Based segment (X'22')

<b>CF</b>	Connection Fault segment (X'12')
<b>CIE</b>	Connection Identifier Exchange segment (X'10')
<b>COB</b>	Client Out of Band segment (X'0F')
<b>CS</b>	Connection Setup segment (X'0D')
<b>SI</b>	Switching Information segment (X'14')
<b>STATUS</b>	Status segment (X'0E')
<b>FID5</b>	FID5 transmission header (TH)
<b>RH</b>	Request/response header
<b>RU</b>	Request unit
<b>RH INDICS</b>	Translated RH indicators. See "RH Indicators" on page 312 for more information.
• For an 8-byte CTC header, the following information is printed:	
<b>RWI</b>	Read/write operation code
<b>CCW</b>	Channel command word (CCW) area index
<b>CTC LINK</b>	Name of the CTC link
<b>CTC HDR</b>	8-byte CTC header.

Figure 99 on page 312 shows an example of a formatted PIUPRT report. The entries are printed in the following order:

- FID4 PIU
- FID2 PIU
- NLP
- 8-byte CTC header.

```

*****
TRANSACTION PROCESSING FACILITY SNA PIU TRACE OUTPUT
*****
A DISPLAY OF PIUS BASED ON THE FOLLOWING USER PARAMETERS WILL BE PERFORMED:
FORMAT HEADERS ALL

*****
TRANSACTION PROCESSING FACILITY PIU TRACE REPORT
*****
THESE PIUS WERE WRITTEN TO TAPE ON 18AUG AT 12.17.08 *****
RWI=52 CCW=01 DNAME= TPFB          ONAME= N34H710          DRID= 000002 ORID= 000010 PCID=          TIME= 08.12
DNA= 0B0000          ONA= 230000          ERVR= 0000          VRSQ= 0000          SEQ= 0001          CNT= 0034          RH= 2B0000
RU = ER-ACT-RP          RH INDICS: REQ NC FI OIC NR
0 ( 0) 0C000001 00010100 00002300 00000000 0B800000 00000000 00000000 00A9BF17          .....Z..
32 ( 20) F51D3A23 54000000 00002300 00000001 00          5.....

-----
RWI=31 CCW=02 DNAME= ELMNGR          ONAME= VTAMNET.CLUBB002 DRID= 0000EE ORID= 00005F PCID= E383BE95D67FFA3 TIME= 15.33
LNKHDR= 00790000          TH = 0101          SID= 0101          SEQ= 0465          RH= EB8000
RU = BIND          RH INDICS: RSP SC FI PRSP
0 ( 0) 31010703 30200002 0081D6D6 81000000 00000000 00000000 0000000E E5E3C1D4          .....a00a...VTAM
32 ( 20) D5C5E34B C5D3D4D5 C7D90000 10E5E3C1 D4D5C5E3 4BC3D3E4 C2C2F0F0 F26016E3          NET.ELMN GR...VTA MNET.CLU BB002--T
64 ( 40) 83BE956D 67FFA30D E5E3C1D4 D5C5E34B E5E3C1D4 F22C0A01 08404040 40404040          c.n...t. VTAMNET. VTAM2...
96 ( 60) 402D0908 C9D5E3C5 D9C1C3E3          ...INTE RACT

-----
RWI=31 CCW=01 DNAME= VTAMNET.JFT10010 ONAME= JFT1          DRID= 00092B ORID= 00091C PCID= E383BE95415060AE TIME= 59.13
LNKHDR= 00DD0000          TCID1= 1F9C92CAC2000005          TCID2= 0D5CAAC700000326          SEQ= 092B
RTP= 000005          SA1= 30C250A1C3567004          SA2= 0000000000000000          BSN= 000005FA
NHDR
0 ( 0) C2088000 0220D000 00000000 0000FF00          B.....
THDR
0 ( 0) 0D5CAAC7 00000326 3C040008 000000A9 000005FA 03228000 0010A211 00000000          *.G....Z.....S....
FID5= 5D00092BB0C250A1C3567004          RH= 6B8000
RU = BIND          RH INDICS: REQ SC FI OIC DR
0 ( 0) 31010303 B1903082 01018585 81010200 00000000 18500000 7E00000C E5E3C1D4          .....b...eea...&...=...VTAM
32 ( 20) D5C5E34B D1C6E3F1 00050004 244A0810 E5E3C1D4 D5C5E34B D1C6E3F1 F0F0F1F0          NET.JFT1 .....VTAMNET. JFT10010
64 ( 40) 6016E383 BE954150 60AE0DE5 E3C1D4D5 C5E34BE5 E3C1D4F2 2B160101 14661280          -.Tc.n.& -.VTAMN ET.VTAM2.....
96 ( 60) 150DE5E3          ..VT

-----
RWI=06 CCW=0B CTC LINK = V2CTCL1          CTC HDR = 03D200012B836A60

```

Figure 99. Formatted PIUPRT Report

See *IBM Systems Network Architecture Network Product Formats* for more information about PIU and NLP formats.

## PIUPRT Utility Return Codes

When you submit the PIUPRT JCL to run the PIUPRT utility, you will receive one of the return codes in the following table:

Table 25. Return Codes for the PIUPRT Utility

Return Code	Description
0	PIUPRT report was created with no errors.
1	PIUPRT report was not created. The parameters specified for the PARM= statement in the PIUPRT JCL were not correct. See "Defining the PIUPRT Report" on page 303 for more information.
2	PIUPRT report was not created. An error occurred while reading the input file.

## RH Indicators

The following information describes the RH indicators that are translated in the formatted display of the PIU trace table and the formatted PIUPRT report.

- RSP Negative response
- BB Begin brackets
- CD Change direction
- CEB Conditional end bracket
- CS Code selection indicator

<b>DFC</b>	Data flow control PIU
<b>DR</b>	Definite response requested
<b>EB</b>	End brackets
<b>ED</b>	Enciphered data
<b>ER</b>	Exception response requested
<b>FI</b>	Format indicator
<b>FIC</b>	First in chain
<b>FMD</b>	Function management data PIU
<b>LIC</b>	Last in chain
<b>MIC</b>	Middle in chain
<b>NC</b>	Network control PIU
<b>NR</b>	No response requested
<b>OIC</b>	Only in chain
<b>PAC</b>	Pacing request or pacing response
<b>PID</b>	Padded data
<b>PRSP</b>	Positive response
<b>QR</b>	Queued response
<b>REQ</b>	Request
<b>RLW</b>	Request larger window size
<b>RSP</b>	Response
<b>SC</b>	Session control PIU
<b>SDI</b>	Sense data indicator.

---

## Including the PIU Trace Table in System Error Dumps

To include the PIU trace table in system error dumps, use the ZIDOT command to specify the SNA keyword (ISNA) in the dump override table. See *TPF Operations* for more information about the ZIDOT command.



## Appendix E. VTAM Mode Table Entries

Figure 100 and Figure 101 contain valid VTAM mode table entries for LU devices supported by TPF. The tables are intended to aid in the understanding of VTAM mode table entries for LUs that TPF recognizes for each supported device. This is what the VTAM mode table should look like if LUTYPE is coded LUTYPE=ANY.

TPF-Descript.	02 FM- PROF	03 TS- PROF	04 PRI- PROT	05 SEC- PROT	06-7 COM- PROT	08 SSN- DPAC	09 SRC- VPAC	0A-B RUSIZES	0C PSN- DPAC
FMMR	2	3	80	80	0000	3F	0	8787	0
NEF	2	3	F0	90	0800	0	0	8787	0
3614	3	4	30	30	0080	1	1	8585	0
3600 Multi-thread/ non-recoverable	3	4	D0	90	4000	0	0	8785	0
3600 Single thread/ non-recoverable/ brackets	3	4	D1	91	7080	0	0	8785	0
3600 Single thread/ recoverable/ brackets	3	4	F1	91	7080	0	0	8785	0
SLU_P (rec)	4	4	B1	B0	7080	3F	3F	8785	0

Figure 100. VTAM Mode Table Entries - Non-3270 LU0. Valid combinations of non-3270 LU0 mode table entry values. PSERVIC is hex 0 for all entries.

MODENT PARAMETERS and their respective valid values for TPF not reflected in the tables are:

### TYPE=1

designates non-negotiable BIND. TPF supplies TYPE=1 regardless of the values in the suggested BIND.

### ENCR=0

designates no encryption supported. Note that this does not rule out private protocol encryption as is supported with 3600 devices. TPF supplies ENCR=0 regardless of the values in the suggested BIND.

### COS=cosname

specifies class\_of\_service name and is valid in all mode table entries.

TPF will determine the device type and reflect this in the RVT at BIND time.

BIND Hex Offset->		02	03	04	05	06-7	08	09	0A-B	0C-D
LU		FM-	TS-	PRI-	SEC-	COM-	SSN-	SRC-		PSN-
Prof	TPF-Image	PROF	PROF	PROT	PROT	PROT	DPAC	VPAC	RUSIZES	DPAC
(TPF TRMEQ Label)		PSERVIC BIND Offset-----> 0E 0F--11 12 13 14-5 16-7 18 19								
LU0	3271-12	2	2	00	40	0000	0	0	8585	0
(TP77M2)		24x80	3277-2				00	000000	00 00 0000	0000 02 00
(TP84M2)		24x80	3284/86-2				00	000000	00 00 1850	0000 02 00
LU1	3274/6-x1C	3	3	B1	90	3080	0	0	8585	0
(TPLU1M1)		3287/9	w/SCS	2k			01	000000	F9 00 0000	0000 00 00
(TPLU1M2)		3287/9	w/SCS	4k			01	000000	F9 00 0000	0000 00 00
LU2	3274/6-x1C	3	3	B1	90	3080	0	0	8585	0
(TP77L2)		24x80	3277-2				02	000000	00 00 0000	0000 02 00
(TP78M2)		24x80	3278-2				02	000000	00 00 1850	0000 7E 00
(TP78M2)		24x80	3278-2,3,4				02	000000	00 00 1850	1850 7F 00
(TP78M3)		24x80	3278-3				02	000000	00 00 1850	2050 7F 00
(TP78M4)		24x80	3278-4				02	000000	00 00 1850	2850 7F 00
LU3	3274/6-x1C	3	3	B1	90	3080	0	0	8585	0
(TP87M2)		24x80	3287/89-2				03	000000	00 00 1850	0000 7E 00
(TP87M2)		24x80	3287/89-2,3,4				03	000000	00 00 1850	1850 7F 00
(TP87M2)		24x80	3287/89-2,3,4				03	000000	00 00 1850	2050 7F 00
(TP87M3)		24x80	3287/89-3				03	000000	00 00 1850	2050 7F 00
(TP87M4)		24x80	3287/89-4				03	000000	00 00 1850	2850 7F 00

#### Notes:

1. The TPF terminal equate (TRMEQ) labels define the valid terminal types for resources which may communicate with TPF applications. For resources defined to TPF as TYPE=ANY, TPF will set the RVT device type 2 field (RV1DVTP2) from the VTAM PSERVIC values received in the SESINIT command from the logon manager, based on the PSERVIC values defined above. For non-3270 devices, PSERVIC value must be zero.
2. LU0 printer presentation field X'13' - X'14' may need to be zeroed when the BIND is built. The field is non-zero to alert TPF to mark RV1DVTP2 as a printer.

Figure 101. VTAM Mode Table Entries - 3270s. Valid combinations of 3270 mode table entry values.



---

## Appendix F. TPF Message Processing Flow User Extensions

This section details how you can extend TPF terminal support. The key feature of this extended support are user-written programs that transform the protocol used by the terminal into the standard TPF Application Program Interface. This extended support may be performed as part of your normal application process or as a separate application front ending your normal application process. Similarly, although it may also be used for other SNA protocols, this discussion focuses on using the PSV exits to implement a Communication and Transmission Control Program (CTCP) for the NPSI Generalized Access to X.25 Terminal Extension (GATE) environment. See "Process Selection Vector (PSV)" on page 253 for detailed information on PSVs.

---

### TPF Inbound Message Flow Extensions

Figure 102 on page 318 illustrates the extended TPF inbound message flow.

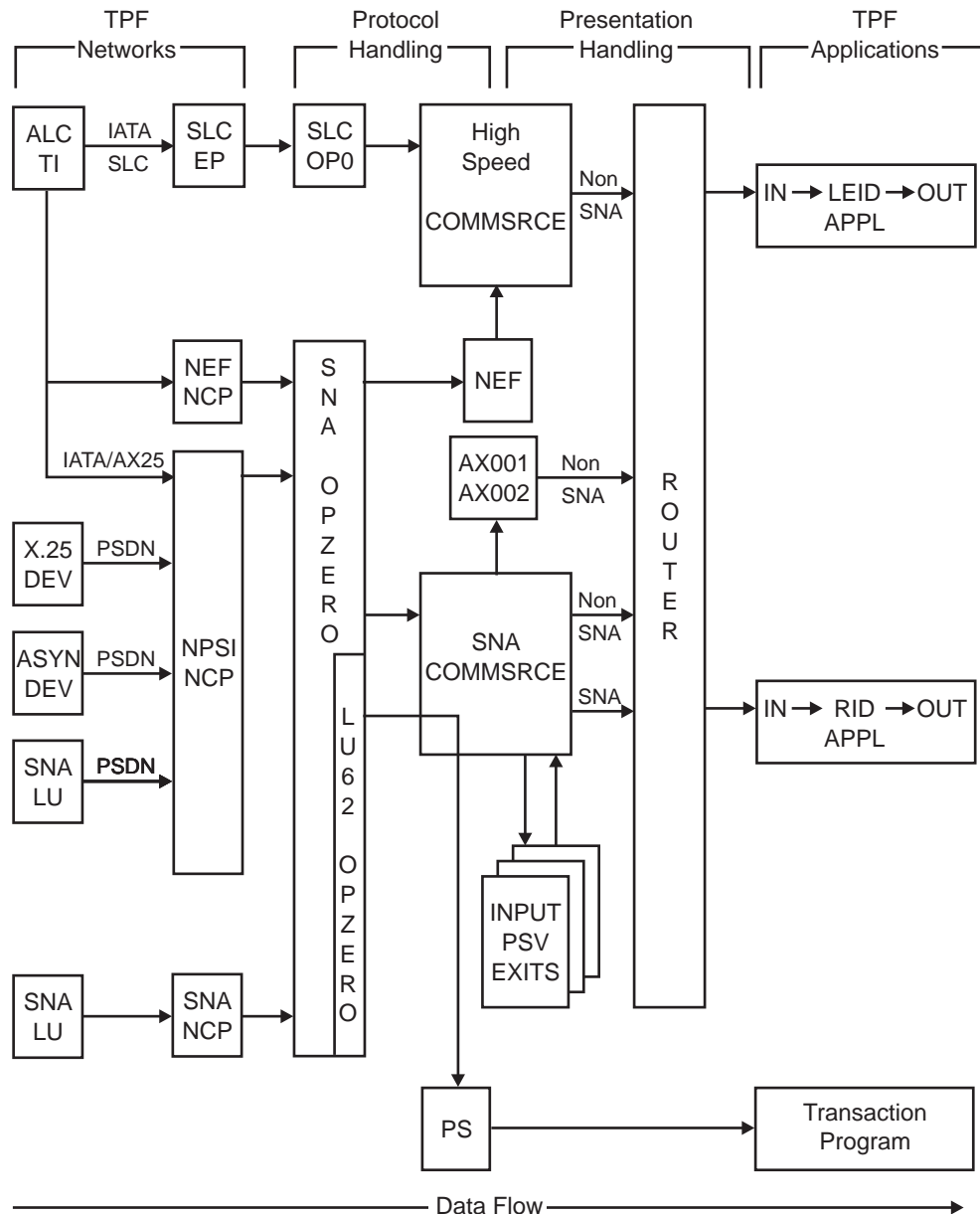


Figure 102. Extended TPF Inbound Message Flow

The PSV name is passed to the exit as a way of identifying the functions and processing required. The PSV name may be used to infer the characteristics of the input message, based upon its source, because each PSV name is unique. A PSV name may share a process with other PSVs or may be associated with a unique process.

When control is returned to COMMSRCE from the exit routine, COMMSRCE then:

- Sets the ECB flag to indicate either an SNA or non-SNA message.
- Sets the ECB field EBROUT to the RID or LEID returned in the RCPL origin field. This field is used by the System Error routines to return a message to the terminal, if an error occurs in delivering or processing the message.
- Starts the appropriate trace and data collection routines to capture information for debugging and performance analysis.

- Starts the appropriate routine to deliver the message to the destination set in the RCPL. The destination can be:
  - an application,
  - the Log Processor,
  - the Unsolicited Message Package,
  - an LEID Addressed End Point, or
  - a RID Addressed Logical Unit.

## TPF Outbound Message Flow Extensions

Figure 103 illustrates the extended TPF outbound message flow.

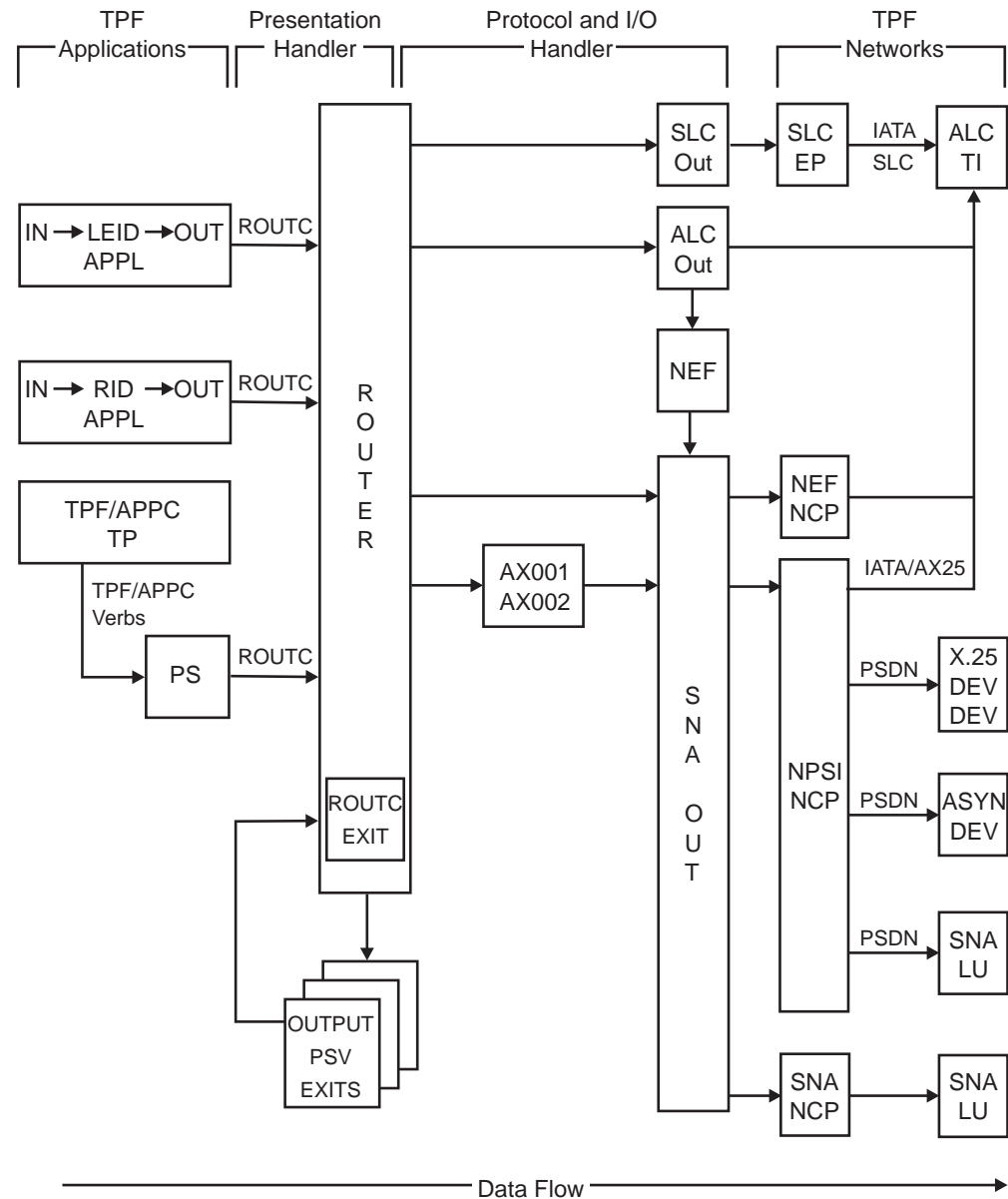


Figure 103. Extended TPF Outbound Message Flow

## ROUTC Exit

As shown in the Figure 103, immediately preceding the PSV exit in the ROUTC code, is the user ROUTC Exit. For an illustration of the user ROUTC Exit, see

Figure 94 on page 255. See *TPF System Installation Support Reference* for additional information about user exits. This exit provides you with the opportunity to dynamically indicate the PSV routine to be selected for a message. This exit is required for messages that have been presented to ROUTC using the LEID interface.

ROUTC macro code starts the PSV routine indicated on return from the user ROUTC Exit, or the PSV routine previously associated with the RID of the destination LU, if the message was presented to ROUTC using a RID interface.

The exit routine is expected to issue the ROUTC macro to send the message to its actual destination. When the exit routine completes its processing it should issue an EXITC macro to return control to the system.

Optionally, the exit routine may need to queue the message on DASD before transmission. TPF provides a generalized queuing package to allow the user to queue messages. See "Queue Manager" on page 269 for more information.

## Appendix G. Sample TPF CTCP Implementation Using PSVs

A TPF Communications and Transmission Control Program (CTCP) is a user-written control program that manages the data flow between devices accessed through a Packet Switched Data Network (PSDN) and existing or new TPF applications. A TPF CTCP is a collection of one or more ECB-controlled programs implemented as a standard TPF application or as a Process Selection Vector (PSV). The main difference of implementing a CTCP as a TPF application, versus a TPF PSV, is related to which side of the TPF ROUTC Application Program Interface (API) the CTCP processing occurs.

A CTCP is required when either a Generalized Access to X.25 Transport Extension (GATE) or Dedicated Access to X.25 Transport Extension (DATE) is specified as an NCP Packet Switching Interface (NPSI) option for handling an X.25 link connected to a PSDN. The information presented in this appendix is based upon the requirements of an interface between a TPF/CTCP and the NPSI/GATE option.

The following sections cover a suggested X.25 Control Block structure to be defined by you for a GATE CTCP implementation using PSV routines to handle X.25 commands and X.25 data messages. These sections are based on the flows shown in Figure 102 on page 318 and Figure 103 on page 319.

### Introduction to CTCP Functions

A TPF/CTCP is required to handle the NPSI commands and data messages that flow between NPSI and TPF on the NPSI to CTCP LU-LU sessions in the GATE environment, as shown in Figure 104.

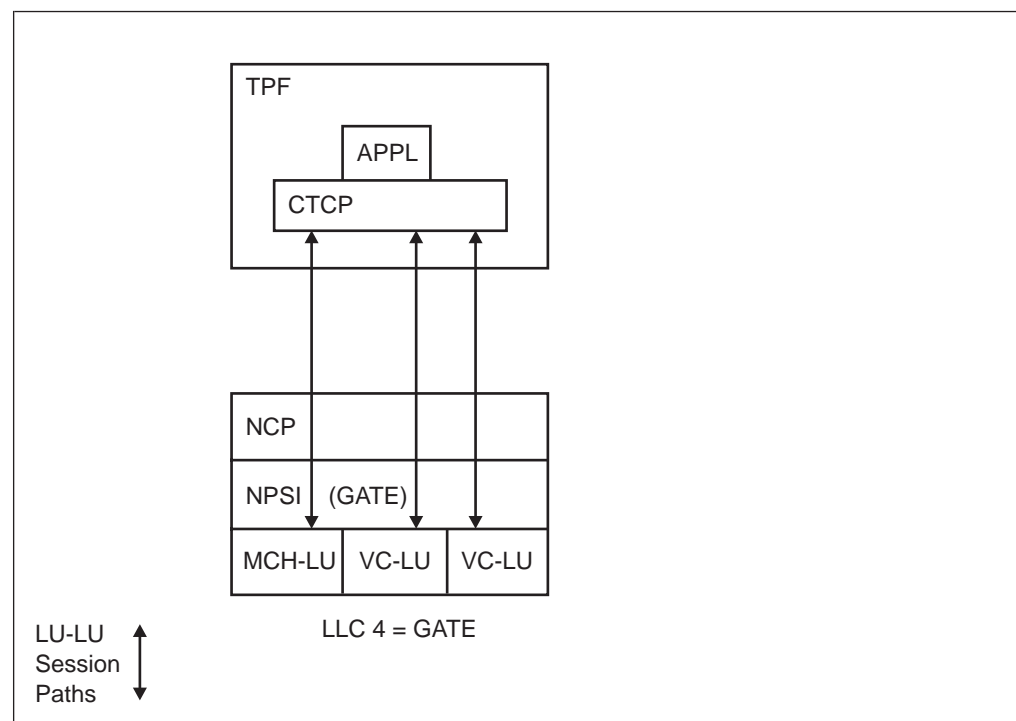


Figure 104. TPF/NPSI Sessions Involving a CTCP

The NPSI commands, which flow between NPSI and the CTCP, control access to the services of the X.25 Packet Switched Data Network (PSDN). The data flows between NPSI, the CTCP, and the application(s) are used to exchange data messages transported through the PSDN.

A principal function of a CTCP is to handle NPSI X.25 network commands for switched virtual circuits, such as:

- Requesting a connection via a CALL\_REQUEST command
- Accepting a connection via a CALL\_CONFIRM command
- Requesting termination of a connection via a CLEAR command.

## Generalized Access to X.25 Transport Extension (GATE)

With the GATE option, the CTCP is the common session partner with NPSI for both data messages and commands. Therefore, a GATE CTCP can be used to provide the equivalent of Open System Interconnect (OSI) layers 4, 5, 6 and a portion of 7.

OSI layer 4 is the Transport layer and includes:

- Establishment phase - switched call requests
- Data transfer phase - blocking, segmentation, multiplexing and demultiplexing
- Termination phase - switched call clearing

OSI layer 5 is the Session layer and includes:

- OSI Session establishment and termination
- Normal and expedited exchanges
- Exception reporting

OSI layer 6 is the Presentation layer and provides:

- Data Transformation
- Data Formatting
  - ASCII
  - EBCDIC
  - Binary
- Syntax Selection

OSI layer 7 is the Application layer and provides:

- Identification of partners
- Authorization
- Privacy
- Dialog Discipline
- End to End Recovery
- Data Transfer

The material that follows details the GATE CTCP environment by describing how you may extend the TPF communication support using PSV routines to allow communication between non-SNA terminal end points and existing or new TPF applications.

---

## User CTCP Control Blocks

This section suggests a control block structure that could be used by the CTCP you implement in TPF to define resources accessed via an X.25 network.

Exit points exist which permit you to create, initialize, and maintain these user control blocks. These exit points include:

- A CTIN main storage allocation exit to reserve space for the tables.
- An exit at TPF Restart to initialize the tables after a software or hardware IPL.
- An exit at Cycle Up and Cycle Down to force checkpointing of the tables.
- An exit in the Critical Record Filing routine to force checkpoint on a catastrophic error.

See the *TPF System Installation Support Reference* for additional information about these exit points.

In SNA, the LU represents a port into the network. NPSI uses the LU to represent both virtual circuits and to provide a port for controlling an X.25 link. The virtual circuit LU is called a VC\_LU. The port for controlling the X.25 link is called an MCH\_LU. An MCH\_LU may relate to one or many VC\_LUs. Both VC\_LUs and MCH\_LUs are defined in the RVT and have a:

- RID - Resource Identifier
- Name - the name of the LU

The suggested user control blocks fall into two categories:

1. The user control blocks defining the network accessed by the X.25 interface protocol and the links and virtual circuits that provide access to the network.
2. The user control blocks defining the terminal and terminal controller resources that can be reached through the X.25 Network.

The CTCP control blocks pertinent to network access are:

- XNCB - X.25 Network Control Block
- XLCB - X.25 Link Control Block
- VCCB - Virtual Circuit Control Block

The control block defining terminals is the End Point Control Block (EPCB).

In addition to these control blocks, all of which are assumed to be main storage resident, the XNCL (X.25 Network Contact List), which could be DASD resident, contains the CPCBs (Contact Point Control Blocks) which define the terminal controllers in the switched networks. Figure 105 illustrates the relationship between elements of the X.25 Network and the suggested user control blocks.

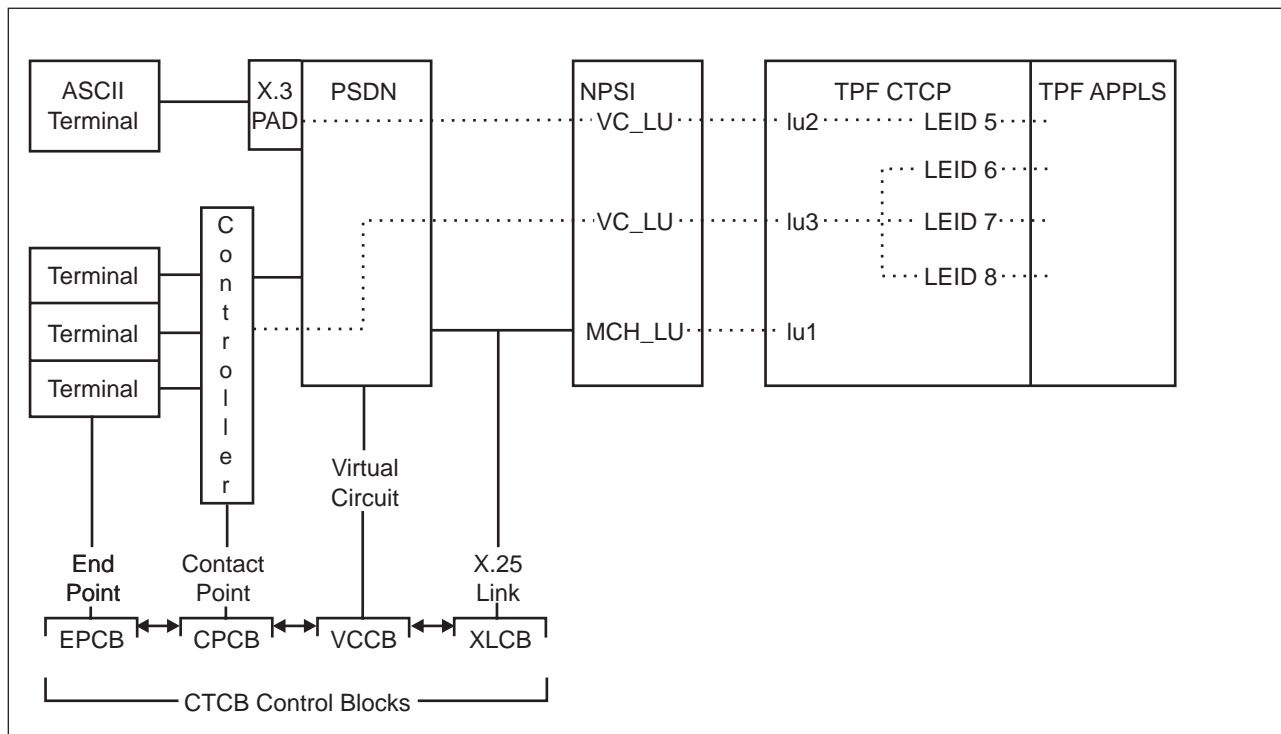


Figure 105. TPF X.25 Network Elements and Corresponding User Control Blocks

## X.25 Network Control Block

The X.25 Network Control Block (XNCB) provides one place to store information about the X.25 network (for example, TRANSPAC or TELENET), and to anchor and index the Network Contact List (XNCL) and X.25 Link Control Blocks (XLCBs) for the network. Information contained in the XLCB might be:

- Network Identification
- Attributes (packet size, window size)
- Command sequence rules
- File Address of the XNCL records for this network
- Pointer to first XLCB for this network

## X.25 Link Control Block (XLCB)

The X.25 Link Control Block corresponds to a NPSI MCH\_LU and contains information about an X.25 link:

- MCH\_LU name,
- X.25 Network ID,
- link options and status, and
- a pointer to the first Virtual Circuit Control Block (VCCB) for the link.

## Virtual Circuit Control Block (VCCB)

The Virtual Circuit Control Block corresponds to a NPSI VC\_LU and contains information about a virtual circuit:

- VC\_LU name,
- status,
- information about the currently attached remote node (the contact information from the XNCL), and



- a pointer to the next VCCB for the X.25 link.

The VCCB is a logical extension of the TPF RVT entry that defines the corresponding VC\_LU. A VCCB entry is either active or idle. When a VCCB is idle, it is available for call in or call out. When the virtual circuit is in use, it contains information about the remote node.

## User Terminal Control Blocks

The remaining control blocks define terminals and terminal controllers that can access TPF applications. The key information needed by these control blocks is a description of the device and its network identification.

The purpose of these terminal tables is two-fold:

1. The terminal tables contain the information to convert the terminal addressing used by the network to the addressing technique used by TPF and its applications (either LEID or RID addressing).
2. The terminal tables define the terminal characteristics to enable conversion from the character set and presentation space definition used by the application to that passed between TPF and the terminal.

Terminals and terminal controllers in an X.25 network are identified by a call user ID. The call user ID-to-device definition relationship requires a way to index to a device definition given a call user ID, and a way to discover the call user ID given a device. The control blocks used are the:

- End-Point Control Block (EPCB) - An EPCB entry describes a terminal. When a switched virtual circuit between TPF and the terminal is established, the address of the VCCB is placed in the EPCB. For a terminal accessed by a permanent virtual circuit, the address of the VCCB is established when the EPCB is initialized.
- Contact Point Control Block (CPCB) - A CPCB defines a terminal controller and its call out and call in identifiers (call user IDs). The CPCB associates a virtual circuit with its user. When a call out is requested, the CPCB provides the destination's network identifier. When a CALL\_REQUEST is received, the caller is identified by locating the CPCB with a matching call user ID. For both call out and call in, pertinent CPCB information is copied into the VCCB. From this information, the CTCP can determine the type of controller and how to code and decode the text in a packet. Collectively, all the CPCBs for a network comprise a Network Contact List (XNCL) for that network.

## End Point Control Block (EPCB)

The EPCB defines the remote resource, usually a terminal. The EPCB contains the following information:

- terminal address (LEID) used by TPF,
- terminal characteristics,
- terminal status,
- a pointer to the currently active VCCB for this terminal,
- a pointer to the Contact Point Control Block that defines the terminal controller, and
- a pointer to the next terminal using the same controller.

## Contact Point Control Block (CPCB)

The CPCB defines a remote terminal controller, its call identifier, and its characteristics. The CPCB contains the following information:

- The network the contact point resides in.
- The identifier or call user ID used to request a virtual circuit to the controller.
- The identifier or call user ID provided in a call request on call in.
- Controller characteristics.
- Controller status.
- A pointer to the first terminal (EPCB) attached to the controller. (This pointer is usually the LEID of the terminal.)

## Control Block Residence

Figure 106 on page 327 illustrates the structure and residency of the suggested CTCP Control Blocks while calls are active.

The control blocks that are referenced for every packet residing in main storage include:

- EPCB - End Points,
- VCCB - Virtual Circuits,
- XLCB - X.25 Links, and
- XNCB - X.25 Networks.

The CPCBs may reside on DASD because they are used only when a CALL\_REQUEST command is received or a call out is required. The contact information from the CPCB, <ci>, needed for processing messages is copied to the VCCB when the virtual circuit has been established.

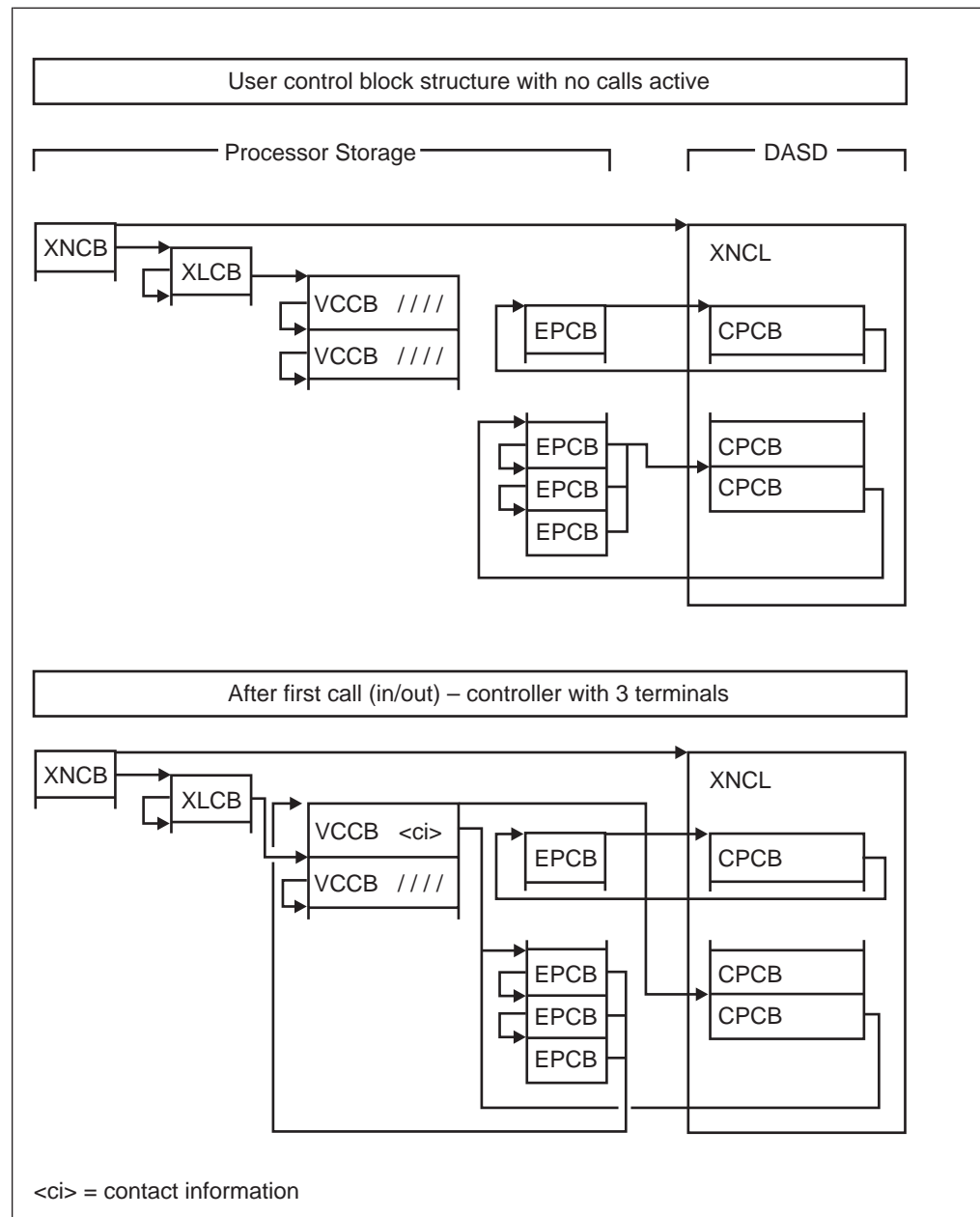


Figure 106. TPF CTCP Control Block Structure

## TPF SNA Session Awareness

When resources are defined as requiring Session Awareness (see “Session Status Awareness Services” on page 215), you are expected to replace the supplied CSXD program with your own code. You can code this exit to reflect the status changes of virtual circuit LUs (VC\_LUs) and multichannel link LUs (MCH\_LUs) in your corresponding VCCB and XLCB control blocks for use by your CTCP.

Whether to start CSXD when a session starts or ends is determined at network definition time. LUs that use session awareness processing must specify AWARE=YES on the OSTG RSC statement or in the Dynamic LU user exit for that LU.

## Session Start

When the Session Awareness Exit (CSXD) is started for an LU using the NPSI/GATE option with the Fast Connect (FC) feature and a session starts, the CTCP should be prepared to receive CALL\_REQUESTs and CLEAR\_REQUESTs. Your exit code should update your corresponding control block (XLCB / VCCB) to indicate that this resource is now active. Optionally, the CTCP can now initiate and terminate calls.

## Session End

When the Session Awareness Exit (CSXD) is started for an LU using the NPSI/GATE option with the Fast Connect (FC) feature and a session ends, the CTCP no longer receives requests for calls. Your exit code should update your corresponding control block (XLCB / VCCB) to indicate that this resource is now inactive and should disconnect the control block structure associated with the virtual circuit.

---

## TPF X.25 Command Message Flows

In this section, the handling of the NPSI protocols involved in the establishment, acceptance, and discontinuance of switched virtual circuits using NPSI/GATE with the Fast Connect (FC) feature is discussed in the context of a CTCP written by you.

The TPF X.25 support provides a means of establishing switched (dial-up) connections between TPF applications and resources accessed via a PSDN. TPF's SNA support, augmented by the SNA Session Awareness routines provided by you (see "TPF SNA Session Awareness" on page 327 in this section, and "Session Status Awareness Services" on page 215), handles the SNA interfaces, both to the VTAM SSCP and the NPSI LUs, necessary to setup and takedown LU-LU sessions required for the connections.

While the SNA connection is transparent to the TPF application, you are required to supply code to manage the underlying NPSI interface protocols required to setup and takedown switched calls.

The following examples are covered:

- A user on a PSDN initiates a call into TPF. This results in a CALL\_REQUEST command being sent from NPSI/GATE to the CTCP. When this call is accepted, the CTCP sends a CALL\_CONFIRM command to NPSI/GATE.
- A user on a PSDN requests termination of a call with TPF. This results in a CLEAR command being sent from NPSI/GATE to the CTCP. Acceptance of the CLEAR command is not required. It is assumed by NPSI/GATE.
- The CTCP initiates a call by sending a CALL\_REQUEST command to NPSI/GATE. NPSI/GATE indicates call setup completion by sending a CALL\_CONFIRM command to the CTCP.
- The CTCP requests termination of a call by sending a CLEAR command to NPSI/GATE. NPSI/GATE disconnects the call and sends a CLEAR\_CONFIRM command back to the CTCP.

Before your CTCP can respond to NPSI/GATE in these examples, you must take such actions as reflecting in your control blocks the changing status of the Virtual Circuits underlying the call.

In the following sections, the CTCP is assumed to be defined via a MSGRTA statement as a TPF application named CTCP. TPF creates an associated RVT

definition of a TPF Primary LU resource named CTCP. The CTCP functions are handled by a PSV routine (one or more ECB controlled programs) named GATE1. GATE1 is specified as the PSV on the OSTG RSC statement for each NPSI LU (both MCH\_LU and VC\_LU) defined to TPF that are in session with the CTCP PLU. It is further assumed that the sessions between the CTCP and the NPSI LU resources (both MCH\_LU and necessary VC\_LU resources) have been established before any NPSI command flows.

**Note:** For the discussed X.25 command flows, the command data received is never presented to a TPF application, but all processing for the command occurs within the GATE1 PSV.

## PSV Processing of NPSI/GATE Commands

When the CTCP to NPSI/GATE LU-LU session has been established for a VC\_LU, all data transmitted across the session is intercepted and given to the PSV routine associated with the VC\_LU; in this case the PSV routine is named GATE1. Whether processing inbound or outbound traffic, the GATE1 PSV routine must perform the CTCP functions associated with the message type (NPSI/GATE command versus data). Examples of the CTCP functions performed by the GATE1 PSV routine are:

- Validation of connection end points.
- Address translation (SNA addresses to TPF internal LEID and application name and vice-versa).
- Control Block Management:
  - Connect Control Block to reflect remote end point connection and path (associate with VC\_LU). (Store VC\_LU RID in EPCB.)
  - Update Control Block to reflect remote end point status.
  - Disconnect Control Block to reflect remote end point connection termination (free end point VC\_LU association). (Clear VC\_LU RID in EPCB.)
- Interface with Queue Manager to:
  - Start transmission of messages queued for remote end point.
  - Stop transmission of messages queued for remote end point.
  - Purge messages queued for remote end point.

(See “Queue Manager” on page 269 for additional information.)

Figure 108 on page 331 shows a generic flow of X.25 commands received from the network processed by the GATE1 PSV on behalf of the CTCP. Figure 111 on page 334 shows a generic flow of X.25 commands initiated by a TPF application/CTCP handled by the GATE1 PSV on behalf of the CTCP destined for the network.

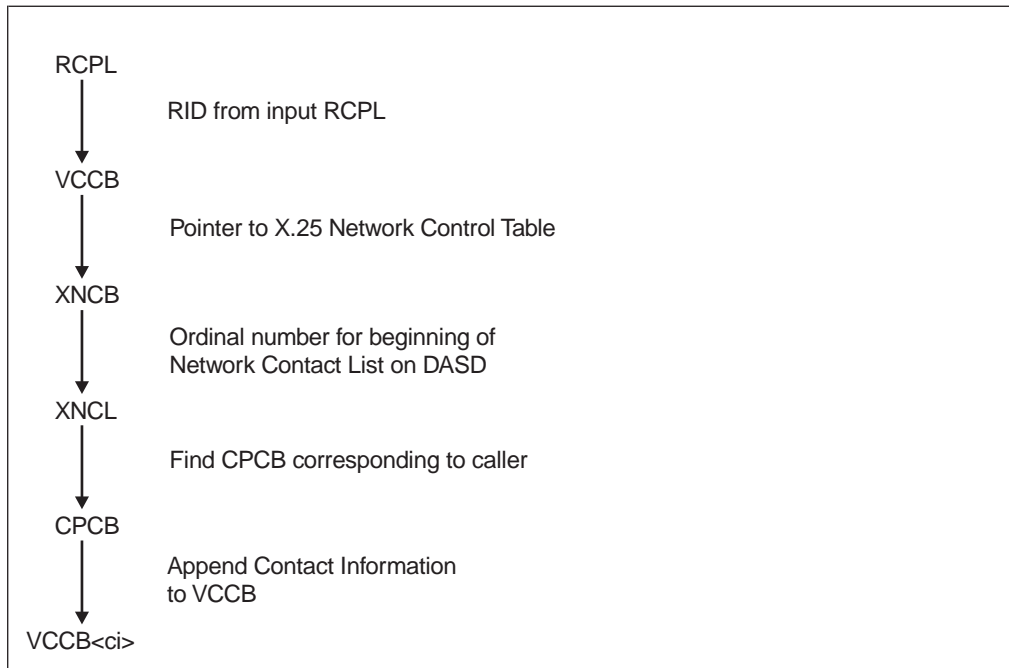


Figure 107. Control Block Relationship for Inbound CALL\_REQUEST Command

### CALL\_REQUEST Command received from NPSI/GATE

See Figure 107 and Figure 108 on page 331 for this scenario.

As shown in the first 3 flow items of Figure 108 on page 331, upon receipt of a PIU from the NPSI/GATE VC\_LU, TPF presents this to the GATE1 PSV using the standard TPF API (RCPL and message block). The first character in each message identifies it as a NPSI/GATE command or a data message. Upon finding a command indication, GATE1 further examines the text of the command to determine that it has just received a CALL\_REQUEST. This is the dial-in situation and the CTCF (GATE1 PSV) must validate the caller and connect the required control block structure in TPF to allow data messages to flow to the application.

Using the RID in the RCPL associated with the CALL\_REQUEST, the PSV routine GATE1 indexes to the associated Virtual Circuit Control Block (VCCB). From the VCCB, the X.25 network ID is obtained and the X.25 Network Control Block (XNCB) for the network is accessed. The ordinal number of the first fixed file record for the X.25 Network Contact List (XNCL) is obtained for the list of valid callers to be searched to insure that this caller is authorized. When a match is found and the caller has been validated, the contact information <ci> for this caller is obtained from the CPCB on file. This information identifies the terminal controller and is appended to the Virtual Circuit Control Block (VCCB), thus making the connection between the NPSI/GATE virtual circuit and the terminals available through the contact point for this call.

Finally, GATE1 (refer to Figure 108 on page 331 flow items number 4 and 5):

- Builds a CALL\_CONFIRM command message.
- Modifies the RCPL (swaps origin and destination address fields and sets appropriate output RCPL indicators).

- Sets the BYPASS SEND INTERCEPT (CE1CPA X'40') indicator to bypass ROUTC EXIT activation for the ensuing ROUTC. See *TPF System Installation Support Reference* for additional information about user exits.
- Issues a ROUTC to forward the CALL\_CONFIRM command to NPSI.
- Terminates the ECB (EXITC).

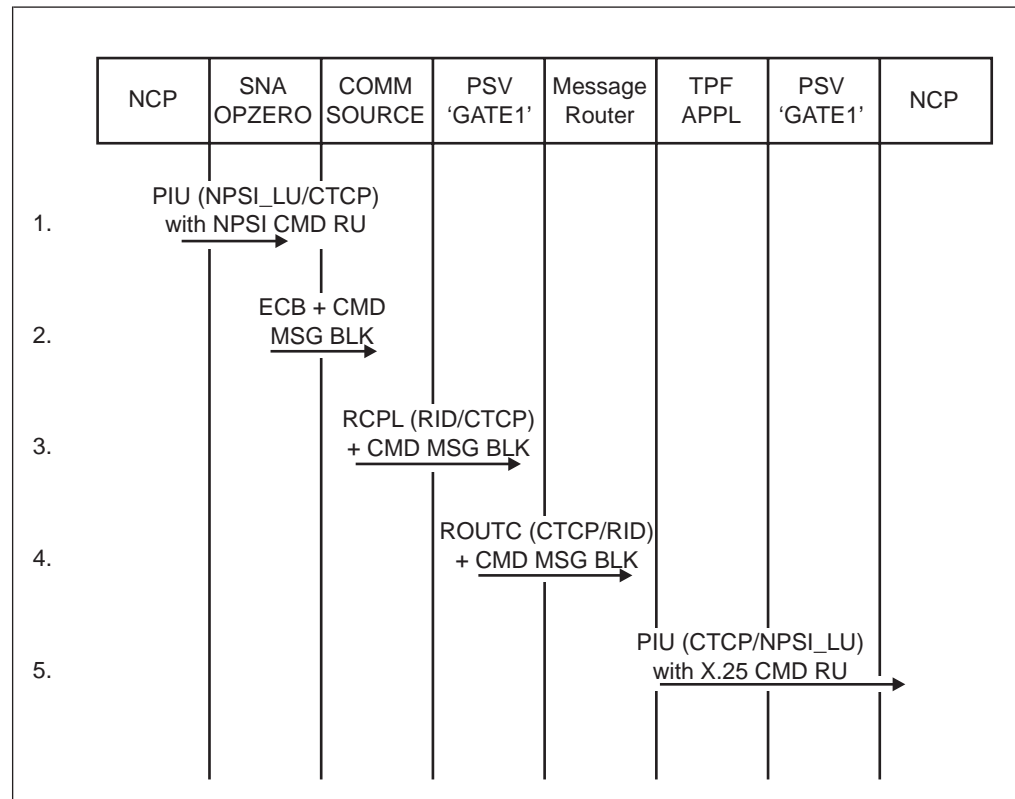


Figure 108. Generic TPF Inbound X.25 CTCP Command/Reply Flow

### CLEAR Command Received from NPSI/GATE

The remote contact point (terminal controller) can send an X.25 clear request when the connection to the host is to be broken. NPSI/GATE sends a CLEAR command to the CTCP when it has completed its handling of the clear request from the X.25 interface. Again assuming GATE/FC, the NPSI/GATE CLEAR command arrives over the same virtual circuit as normal data messages.

See Figure 108 and Figure 109 on page 332 for this scenario.

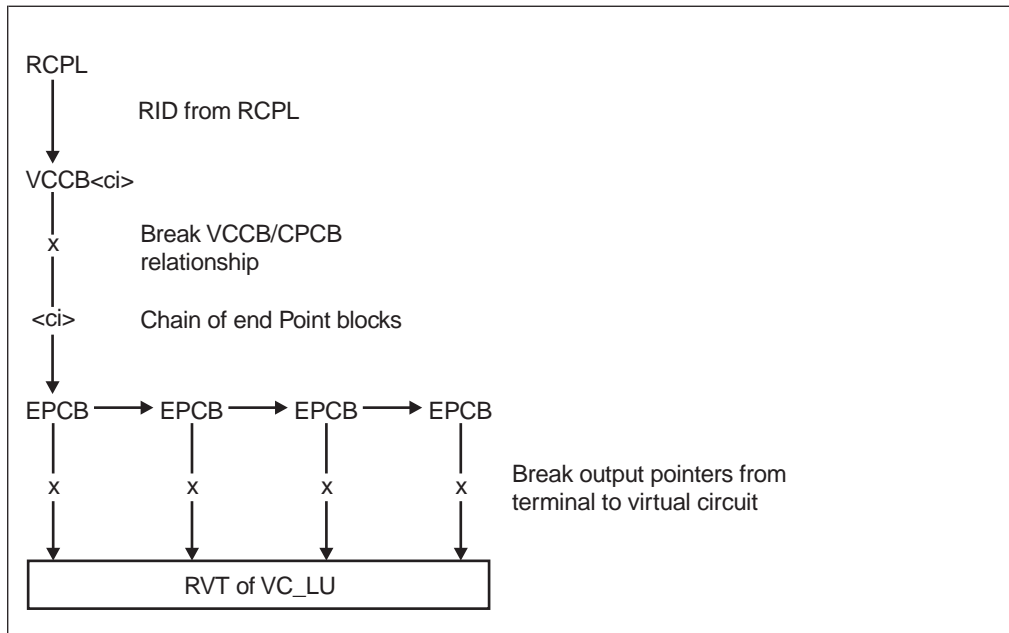


Figure 109. Control Block Relationship for Inbound CLEAR Command

As shown in first 3 flow items of Figure 108 on page 331, upon receipt of a PIU from the VC\_LU destined for the CTCP, TPF presents this to the GATE1 PSV using the standard TPF API (RCPL and message block). The inbound GATE1 PSV routine recognizes that this is a NPSI/GATE CLEAR command and the process shown in Figure 109 begins. The RID of the NPSI/GATE VC\_LU, passed by TPF in the RCPL, is used to find the Virtual Circuit Control Block (VCCB). The contact information <ci> appended to the Virtual Circuit Control Block (VCCB<ci>) is saved before resetting (clearing) the <ci> to break the relationship with the VCCB that was established during call in processing. The VCCB is then marked as available to the chain of virtual circuits in the X.25 Link Control Block (XLCB) for this X.25 link.

Next, each end point on this contact point must be handled. For each End Point Control Block (EPCB) in the <ci> chain, the EPCB field containing the VC\_LU RID is cleared, thus breaking the application/end point connection across the X.25 network.

GATE1 PSV then terminates the CLEAR command ECB (EXITC).

### TPF Initiated Call Out Using CALL\_REQUEST Command to NPSI/GATE

Your CTCP can initiate a call out process when there is a need to communicate with a disconnected remote terminal controller. This might occur when there is:

- time initiated ticket printing,
- urgent unsolicited output messages,
- output queues triggered by queue count, or
- computer operator requests for call out.

These events trigger sending a CALL\_REQUEST command to NPSI/GATE. Figure 110 on page 333 and Figure 111 on page 334 are references for this scenario.



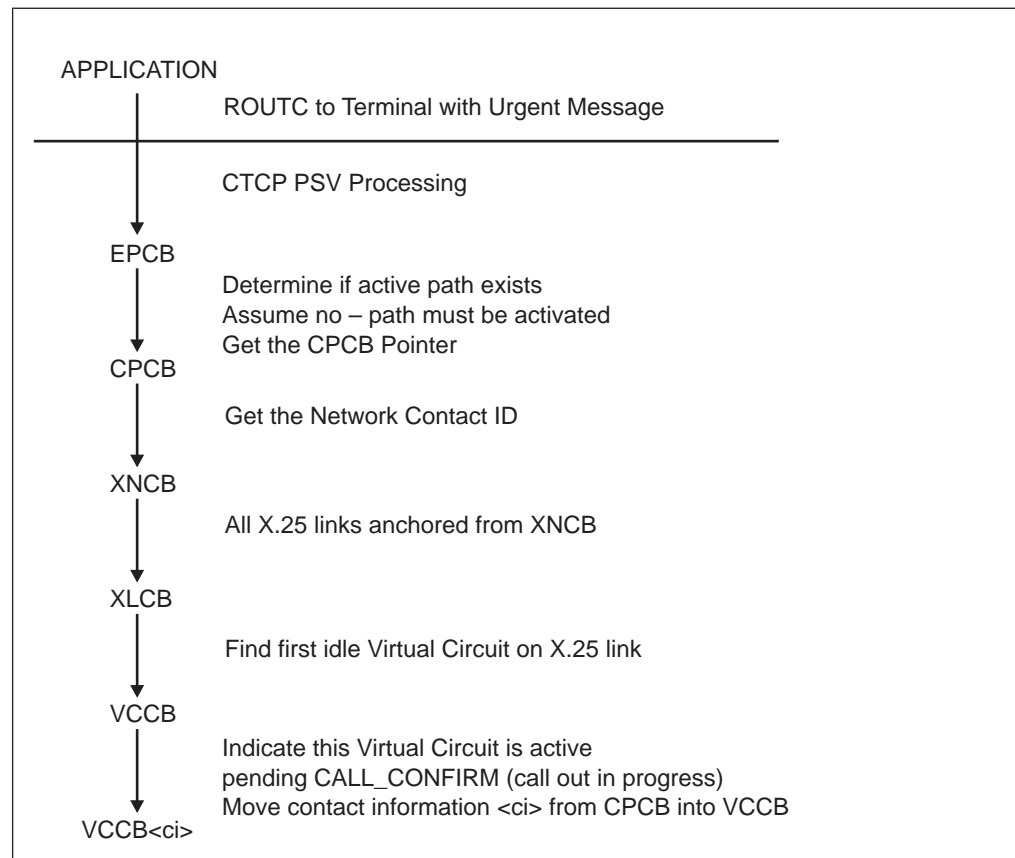


Figure 110. Control Block Relationship for CALL\_OUT Request

An application sends a message to a terminal via a ROUTC. Your ROUTC EXIT processing code determines that the LEID specified in the RCPL requires GATE1 PSV processing. See the *TPF System Installation Support Reference* for additional information about user exits. Your code returns to the TPF ROUTC processing, indicating that the message must be given to GATE1 for processing. TPF ROUTC removes the outbound message from the application's ECB and attaches it to a newly created ECB to be presented to the GATE1 PSV.

For this activation, your CTCP, implemented in the 'GATE1' PSV routine, uses the terminal's LEID specified in the RCPL to access the EPCB and determines that no call is currently active with the terminal. You activate the Queue Manager facility which places the message on the terminal's message queue and start your CTCP call out processing. (See "Queue Manager" on page 269 for detailed information.)

Your CTCP call out processing uses the pointer in the EPCB to access the CPCB on DASD. From the CPCB, the call user ID of the controller (contact point) for this terminal is obtained and a NPSI/GATE CALL\_REQUEST command can be constructed.

After the call user ID is obtained, the CTCP finds an idle X.25 virtual circuit using the X.25 Link Control Block chain for the network identified in the CPCB.

The contact information <ci> for this terminal controller is copied from the CPCB into the VCCB for the selected, available virtual circuit, and the VCCB is flagged as having a call out pending. All tables are set appropriately and the necessary

information is extracted to build a NPSI/GATE CALL\_REQUEST command. An outbound RCPL specifies the CTCP as origin and the RID of the VCCB is selected, as the destination is then constructed.

Finally, the GATE routine sets the BYPASS SEND INTERCEPT (CE1CPA X'40') indicator to bypass ROUTC EXIT activation for the ensuing ROUTC. GATE issues the ROUTC macro with the modified RCPL and message block to forward the CALL\_REQUEST to the network.

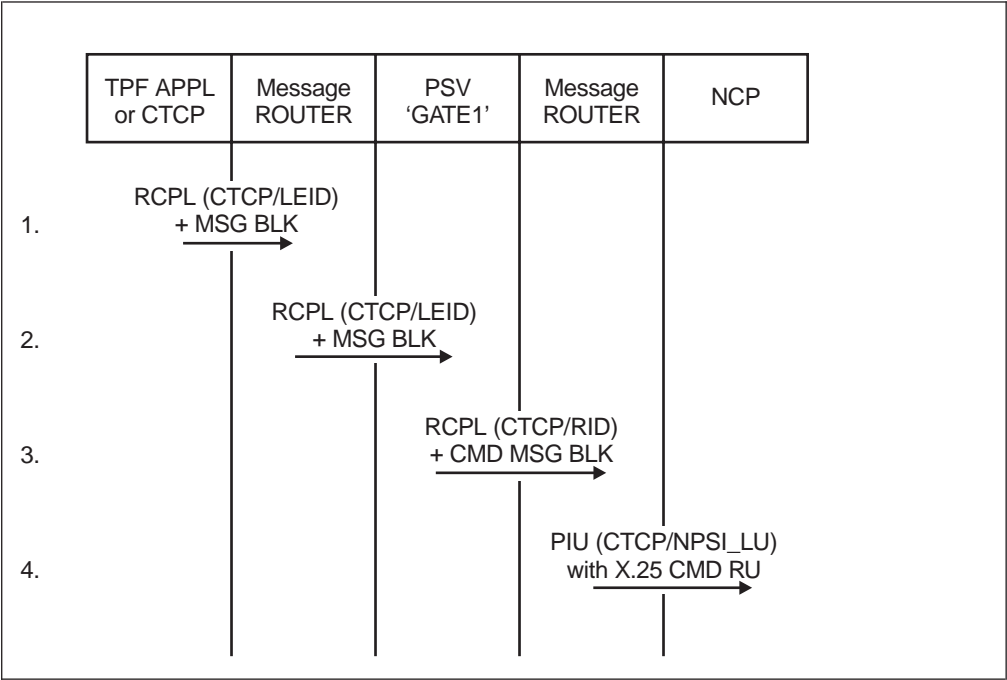


Figure 111. Generic TPF Outbound X.25 CTCP Command Processing Flow

**CALL\_CONFIRM Command Received from NPSI/GATE**

An inbound CALL\_CONFIRM command is the normal response to an outbound CALL\_REQUEST command. The CALL\_CONFIRM shows that the remote controller has accepted the call and is now ready to receive data.

With GATE/Fast Connect, the CALL\_CONFIRM comes in on the same VC\_LU-to-CTCP session on which the CALL\_REQUEST was sent. Figure 112 and Figure 108 on page 331 are references for this scenario.

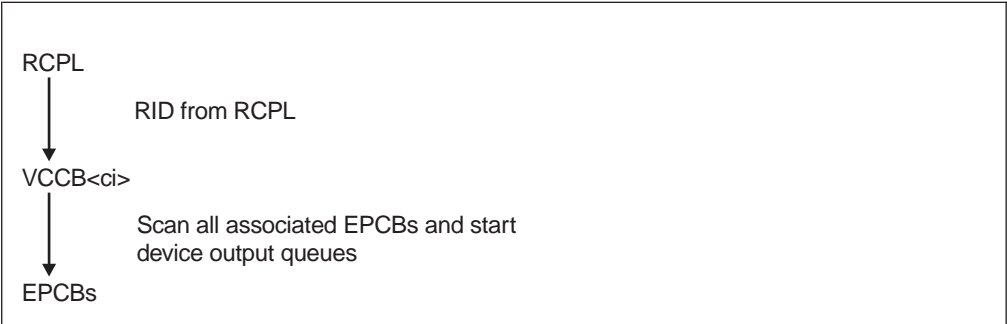


Figure 112. Control Block Relationship for Inbound CALL\_CONFIRM Command

As shown in the first 3 flow items of Figure 108 on page 331, upon receipt of a PIU from the VC\_LU, TPF presents this message to the GATE1 PSV using the standard TPF API (RCPL and message block).

Upon determining that this is a CALL\_CONFIRM command from NPSI/GATE, the following major actions are performed:

- Change the VCCB status from call pending to call active.
- Activate the CTCP X.25 control blocks so that data can be sent and received on this virtual circuit.
- For each terminal reachable on this call, initiate the sending of output if there is data queued.

When the CALL\_CONFIRM command from NPSI/GATE is passed to GATE1, the VCCB is located by using the RID provided by TPF in the RCPL. The virtual circuit (VCCB) is marked active and each terminal or end point activated. This process involves storing the RID for the virtual circuit in each EPCB associated with the contact point. As each end point EPCB is handled, its output pending indicator is interrogated and if on, GATE1 interacts with the Queue Manager facility to initiate sending queued output to each terminal via ROUTC. For more information, see “Queue Manager” on page 269.

Upon completion of initiating the output for each terminal, the GATE1 terminates the ECB (EXITC).

### **TPF Initiated CLEAR Command Sent to NPSI/GATE**

The CTCP should end a call that it initiated when the virtual circuit is no longer needed. This might occur under the following circumstances:

- When output queues for terminals on this contact point have remained empty long enough.
- Computer operator requests termination of call.

These events trigger sending a CLEAR command to NPSI/GATE.

The linkage between the EPCBs and the VCCB should be deactivated in order that no further attempt to send messages on this call will be made. The VCCB is marked to show the VC\_LU as CLEAR pending CLEAR CONFIRM. The VC\_LU should not be reused until the CLEAR\_CONFIRM is received from NPSI/GATE. Otherwise, the processing is the same as for receiving a CLEAR command from NPSI. See “CLEAR Command Received from NPSI/GATE” on page 331.

### **CLEAR\_CONFIRM Command Received from NPSI/GATE**

After a CLEAR command is sent from the CTCP to NPSI/GATE, NPSI must notify the CTCP when the switched virtual circuit connection has been cleared by returning a CLEAR\_CONFIRM command.

The action to be taken is to change the status of the VC\_LU from CLEAR pending CLEAR\_CONFIRM to CLEAR.

---

## **TPF X.25 Data Message Flows**

### **CTCP Input Data Message Processing**

The CTCP receives both commands and data from NPSI for virtual circuits defined for LLC4 (GATE) support. For this dialog, it is assumed that the NPSI/GATE support is further defined to have Fast Connect (FC) capability, resulting in both data

messages and NPSI command messages flowing on the same virtual circuit to the host. The first text character in each message indicates whether it is a NPSI command or a data message.

In the following sections, the CTCP is assumed to be defined via a MSGRTA statement as a TPF application named CTCP. TPF creates an associated RVT definition of a TPF Primary LU resource named CTCP. The CTCP functions are handled by a PSV routine (one or more ECB controlled programs) named GATE1. GATE1 is specified as the PSV on the OSTG RSC statement for each NPSI LU (both MCH\_LU and VC\_LU) defined to TPF which are in session with the CTCP PLU. It is further assumed that the sessions between the CTCP and the NPSI LU resources (both MCH\_LU and necessary VC\_LU resources) have been established prior to any NPSI command flows, and that although the PIU received is directed to the CTCP PLU, all data messages are ultimately destined for a TPF application named 'APPL'.

This section describes the input-related flow and control blocks for inbound data messages flowing through your CTCP Process. This material is divided into two parts: data flow for soft copy and hard copy devices.

**Soft Copy Device Data Flow**

See Figure 113 for this discussion of soft copy device data flow.

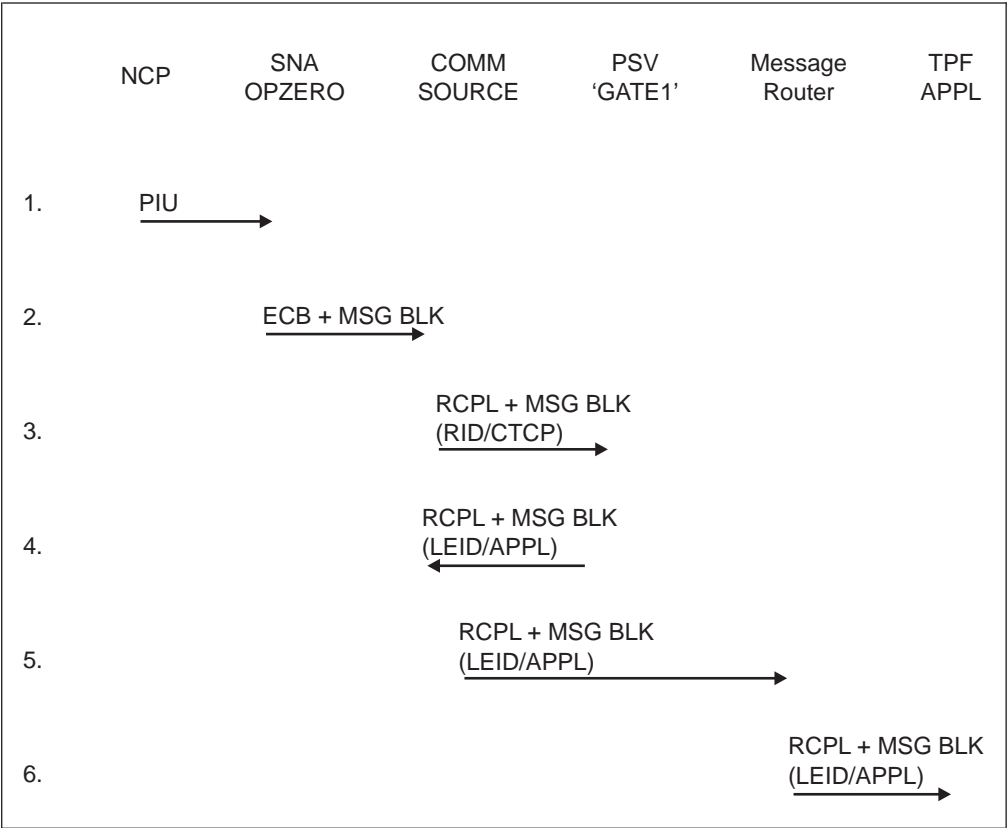


Figure 113. Soft Copy Input Data Flow

**Soft Copy Device Input Message Data Flow:**

1. An input PIU is read by the TPF system from the NCP.
2. The PIU is passed to SNA Opzero, where an ECB is created and the PIU is placed into the appropriate message block.

3. The Communications Source package (COMMSRCE) is then started. This is where the RCPL is created, including the RID of the origin VC\_LU and the name of the destination CTCP. COMMSRCE passes control to the GATE1 PSV routine.
4. The GATE1 PSV routine may perform various actions before returning to COMMSRCE, including the following:
  - Convert the RID of the input VC\_LU, specified in the RCPL, to an LEID. The LEID is determined from contact information <ci> found in the VCCB<ci> and the address information, if any, contained in the message text.
  - Obtain the associated EPCB for the LEID.
  - Modify the RCPL to reflect the LEID, rather than the VC\_LU RID, as the message origin.
  - Modify the RCPL to reflect APPL, rather than the CTCP, as the message destination. The application name is obtained from information stored in the EPCB during logon processing.
  - Translate the message into character coding required by the application.
  - Convert the input data from AMSG format to the message format required by the application.
  - Store the RID into the EPCB to facilitate finding the return address for the resulting output.

The GATE1 routine returns control to COMMSRCE in order to:

- Interface to TPF data collection.
  - Interface to TPF Program Test Vehicle, if appropriate.
5. COMMSRCE then passes control to the Message Router package with the RCPL, passed from the GATE1 routine, and the modified message block.
  6. The Message Router enters the TPF application (APPL) specified in the RCPL.

### Hard Copy Device Data Flow

See Figure 114 for this discussion of hard copy device data flow.

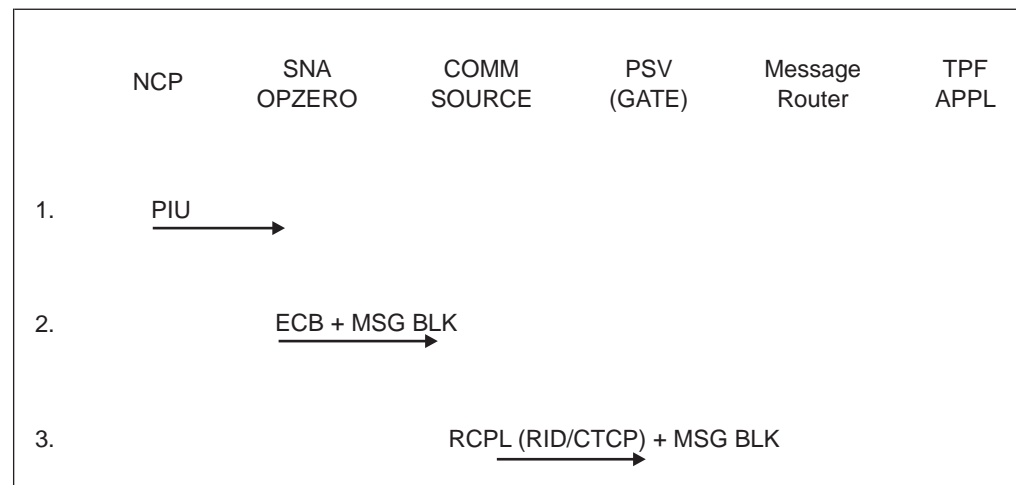


Figure 114. Hard Copy Input Data Flow

#### Hard Copy Device Input Message Data Flow:

1. An input PIU is read by the TPF system from the NCP.
2. The PIU is passed to SNA Opzero, where an ECB is created and the PIU is placed into the appropriate message block.

3. The Communications Source package (COMMSRCE) is then started. The RCPL is created, including the RID of the origin VC\_LU and the name of the destination CTCP. The Communications Source package passes control to the GATE1 PSV routine.
4. The GATE1 routine may perform various including the following:
  - Convert the RID of the input VC\_LU, specified in the RCPL, to an LEID. The LEID is determined from contact information <ci> found in the VCCB<ci> and the address information, if any, contained in the message text.
  - Obtain the associated EPCB for the LEID.
  - Modify the RCPL to reflect the LEID, rather than the VC\_LU RID, as the message origin.
  - Modify the RCPL to reflect APPL, rather than the CTCP, as the message destination. The application name is obtained from information stored in the EPCB during logon processing.
  - Store the RID into the EPCB to facilitate finding the return address for the resulting output.
  - Determine whether the message is a positive or negative acknowledgment (answer-back) and indicate the type of acknowledgement in the GDA area of the RCPL.
  - Swap the Origin and Destination addresses within the RCPL.

At this point, output processing, where the queueing package is started with the printer acknowledgement message, must be activated to perform the necessary queue and transmit manipulations as dictated by the acknowledgement type. The GATE1 PSV may activate the output processing in either of two ways:

1. Using the Input Message ECB - You can directly activate the GATE1 PSV output processing.
2. Using a new ECB - Issues ROUTC with the modified RCPL (swapped origin and destination fields) and same message block. Your ROUTC EXIT processing code determines that the LEID specified in the RCPL requires GATE1 PSV processing. See the *TPF System Installation Support Reference* for additional information about user exits. Your ROUTC EXIT code returns to the TPF ROUTC processing indicating that the message must be given to GATE1 for processing. TPF ROUTC removes the outbound message from the ECB associated with the input message and attaches it to a newly created ECB to be presented to the GATE1 PSV for output processing.

The continuation of the flow is shown in Figure 118 on page 342 and Figure 119 on page 343. This is covered later in "Hard Copy Device Data Flow" on page 339.

## CTCP Output Data Message Processing

This section describes the output related flow and control blocks for outbound data messages flowing through your CTCP process. This material is divided into two parts: data flow for soft copy and hard copy devices.

### Soft Copy Device Data Flow

See Figure 115 on page 339 for this discussion of soft copy device data flow.

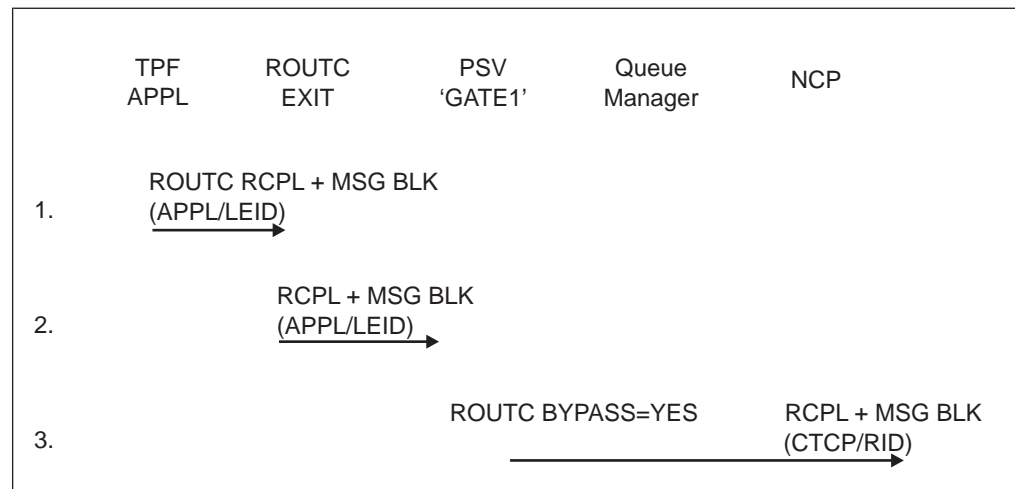


Figure 115. Soft Copy Output Data Flow

### Soft Copy Device Output Message Data Flow:

1. The TPF application (APPL) issues a ROUTC macro with an RCPL and a message block. The origin in the RCPL is the TPF application and the destination is an LEID representing the destination terminal.
2. Your ROUTC EXIT processing code determines that the LEID specified in the RCPL requires CTCP processing by the GATE1 PSV routine. See the *TPF System Installation Support Reference* for additional information about user exits. Your code returns to the TPF ROUTC processing indicating that the message must be given to GATE1 for processing. TPF ROUTC removes the outbound message from the application's ECB and attaches it to a newly created ECB to be presented to the GATE1 PSV.
3. On invocation, the GATE1 routine uses the LEID in the output RCPL to find the associated terminal entry in the EPCB. From this entry, the RID of the X.25 Virtual Circuit LU for this terminal is obtained. The RID of the VC\_LU replaces the LEID in the destination field of the RCPL.  
The End Point Control Block for this terminal is found by using the LEID as an index. This control block provides the information necessary to reformat the message (that is, physical terminal address, and character code translation indicator). Thus, the GATE1 routine prepares the output message for transport to the X.25 packet switching network.
4. Finally, the GATE1 routine sets the BYPASS SEND INTERCEPT (CE1CPA X'40') indicator to bypass ROUTC EXIT activation for the ensuing ROUTC. GATE1 issues the ROUTC macro with the modified RCPL and message block.

### Hard Copy Device Data Flow

The following section describes different cases of data flow for hard copy type devices.

**Hard Copy Device Output Message Data Flow (Enqueue / Dequeue / Transmit):** See Figure 116 on page 340 for this discussion of hard copy device data flow.



- 340 TPF V4R1 ACF/SNA Data Communications Reference



**Hard Copy Device Output Message Data Flow (ENQUEUE / EXIT):** See Figure 117 for this discussion of hard copy device data flow.

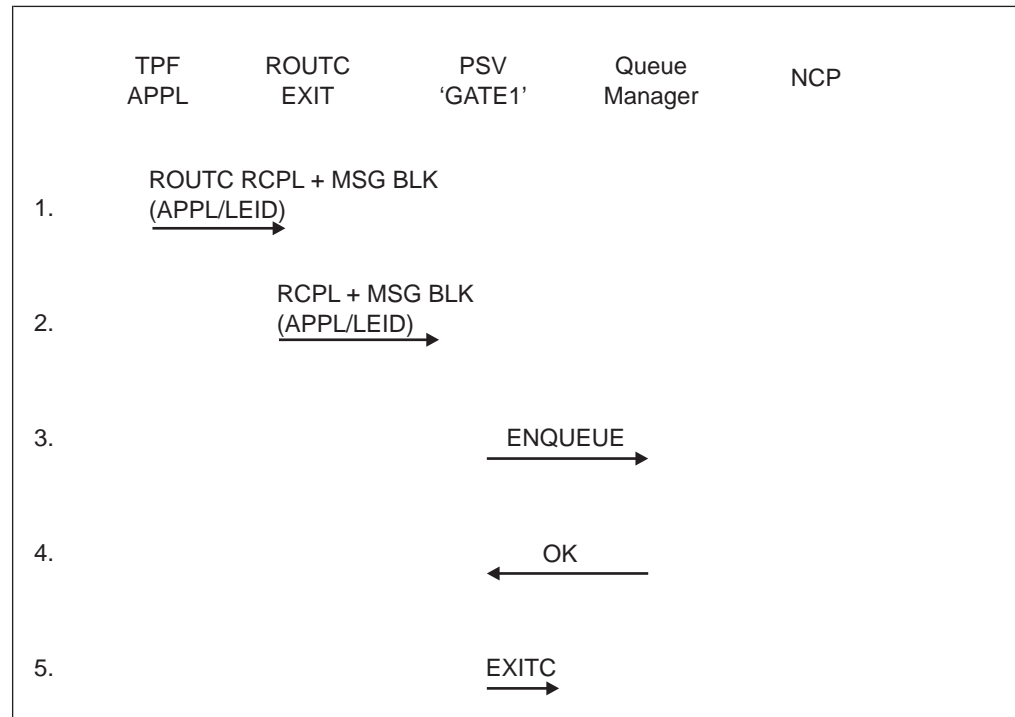


Figure 117. Hard Copy Enqueue and Exit

1. The TPF application issues a ROUTC macro with an RCPL and a message block. The origin in the RCPL is the TPF application and the destination is an LEID representing the destination printer.
2. Your ROUTC EXIT processing code determines that the LEID specified in the RCPL requires GATE1 PSV processing. See the *TPF System Installation Support Reference* for additional information about user exits. Your code returns to the TPF ROUTC processing indicating that the message must be given to GATE1 for processing. TPF ROUTC removes the outbound message from the application's ECB and attaches it to a newly created ECB to be presented to the GATE1 PSV.
3. On invocation, the GATE1 PSV activates the TPF Queue Manager package, to queue the message. For more information on the Queue Manager interface, see "Queue Manager" on page 269.
4. If the GATE1 PSV determines that transmission to the device is not possible or does not wish to transmit, the ECB is terminated. You can write time initiated routines that directly interface with the Queue Manager in order to dequeue and transmit the queued messages according to your specifications.

**Hard Copy Device Output Message Data Flow (WAKEUP):** See Figure 118 on page 342 for this discussion of hard copy device data flow.

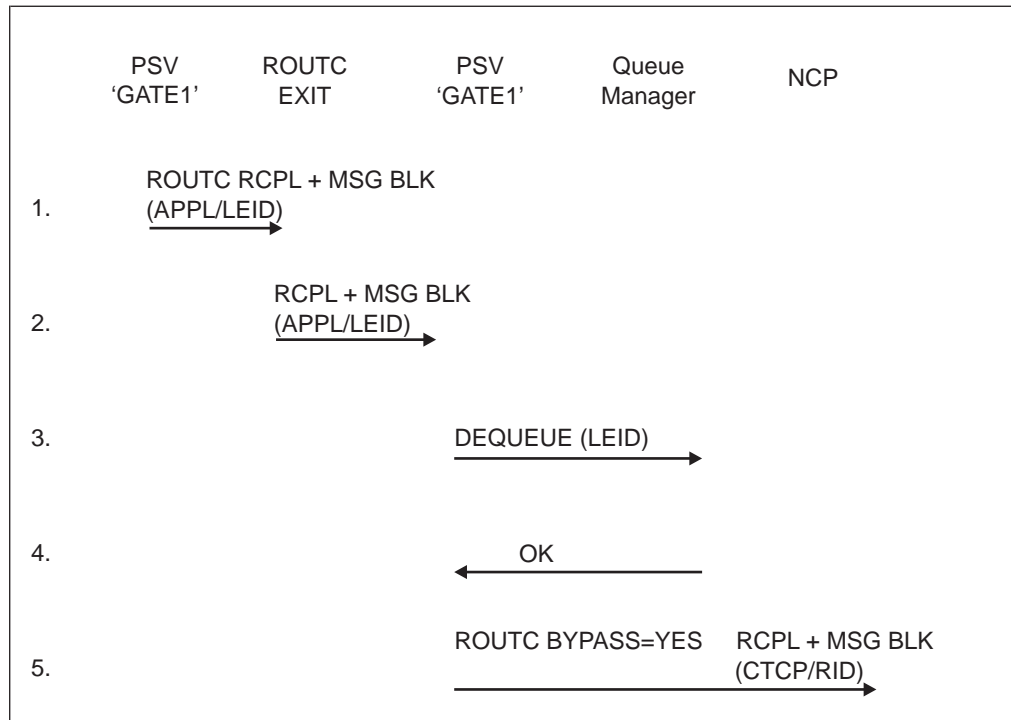


Figure 118. Hard Copy Queue Wake Up

In this case, the ROUTC macro was issued by the GATE1 PSV code activated from COMMSRCE when a positive acknowledgment (answer-back) to a previously sent message was received (see Figure 114 on page 337). The intent is to wake up the existing queue.

1. Using the modified RCPL and message, the GATE1 PSV code activated by COMMSRCE forwards the message via ROUTC.
2. Your ROUTC EXIT processing code determines that the LEID specified in the RCPL requires GATE1 PSV processing. Your code returns to the TPF ROUTC processing indicating that the message must be given to GATE1 for output (WAKEUP) processing. TPF ROUTC removes the outbound message from the ECB associated with the input message and attaches it to a newly created ECB to be presented to the GATE1 PSV for output processing.
3. On invocation, the GATE1 output processing determines that the existing message is a positive acknowledgment. A possible implementation to achieve this might be to have the GATE1 PSV code associated with the input message ECB indicate in the GDA area of the RCPL that this is a positive acknowledgement.
4. If the GATE1 PSV determines transmission to that device is possible, the Queue Manager is activated to dequeue a message block from the top of the QCB (see "Queue Manager" on page 269).
5. The GATE1 routine may first reformat the message and then replace the LEID of the destination in the RCPL with the RID of the VC\_LU and the origin field of the RCPL with the CTCP application name.
6. Finally, the GATE1 routine sets the BYPASS SEND INTERCEPT (CE1CPA X'40') indicator to bypass ROUTC EXIT activation for the ensuing ROUTC. GATE1 issues the ROUTC macro with the modified RCPL and message block.

**Hard Copy Device Output Message Data Flow (RETRANSMISSION):** See Figure 119 on page 343 for this discussion of hard copy device data flow.

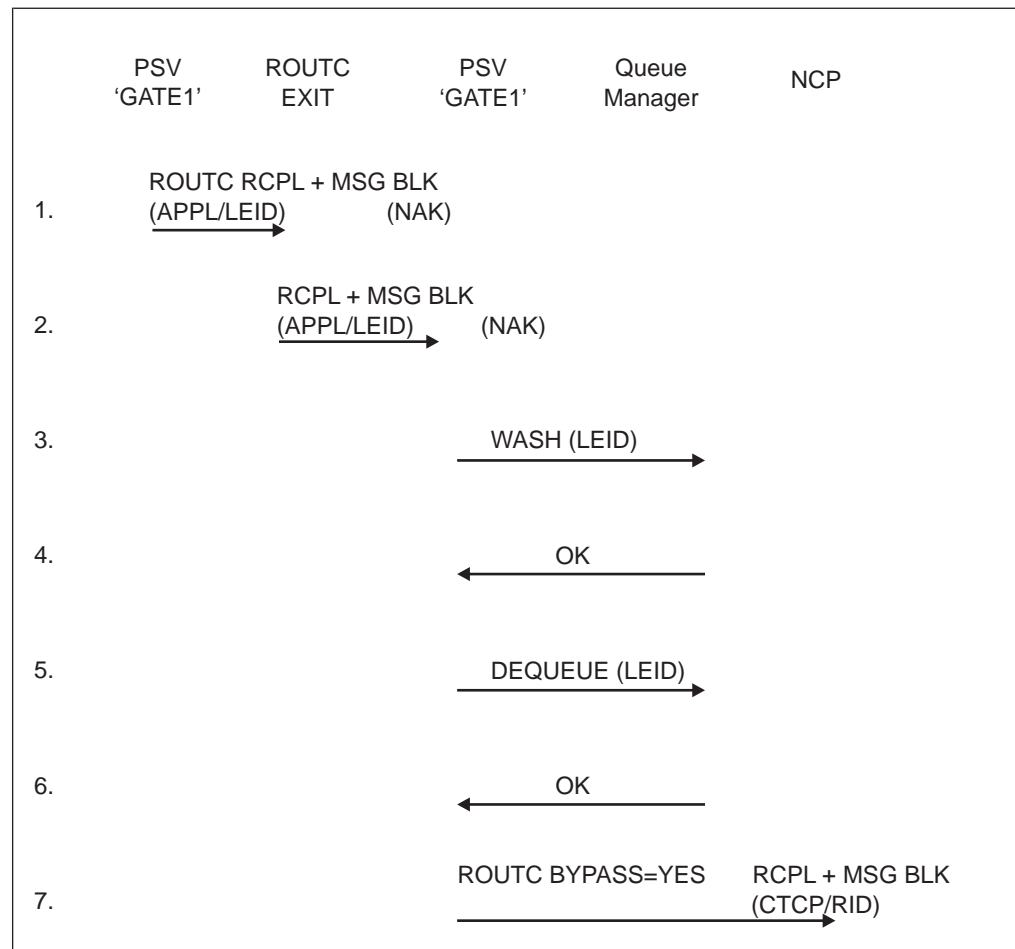


Figure 119. Hard Copy Repeat Last Message/Negative Acknowledgment

In this case, the ROUTC macro was issued by the GATE1 PSV code activated from COMMSRCE when a negative acknowledgment (answer-back) to a previously sent message was received (see Figure 114 on page 337). The intent is to wash the previously sent message and retransmit it.

1. Using the modified RCPL and message, the GATE1 PSV code activated by COMMSRCE forwards the message via ROUTC.
2. Your ROUTC EXIT processing code determines that the LEID specified in the RCPL requires GATE1 PSV processing. Your code returns to the TPF ROUTC processing indicating that the message must be given to GATE1 for output (WAKEUP) processing. TPF ROUTC removes the outbound message from the ECB associated with the input message and attaches it to a newly created ECB to be presented to the GATE1 PSV for output processing.
3. On invocation, the GATE1 output processing determines that the existing message is a negative acknowledgment. A possible implementation to achieve this might be to have the GATE1 PSV code associated with the input message ECB indicate in the GDA area of the RCPL that this is a negative acknowledgement. the GATE1 PSV output processing then requests from the Queue Manager to WASH the previously sent message (see “Queue Manager” on page 269).
4. If the GATE1 PSV determines transmission to that device is possible, the Queue Manager is activated to dequeue a message block from the top of the QCB.

5. The GATE1 routine may first reformat the message and then replace the LEID of the destination in the RCPL with the RID of the VC\_LU and the origin field of the RCPL with the CTCP application name.
6. Finally, the GATE1 routine sets the BYPASS SEND INTERCEPT (CE1CPA X'40') indicator to bypass ROUTC EXIT activation for the ensuing ROUTC. GATE1 issues the ROUTC macro with the modified RCPL and message block.

## Appendix H. SNA Command Flow

The following sections show examples of session activation and deactivation for different LU types.

### PU 5 and PU 2.1 LEN Session Activation

The following sections show examples of session activation when the TPF system is connected as a PU 5 node or PU 2.1 low-entry networking (LEN) node. See “APPN Session Activation” on page 374 for session activation flows when the TPF system is connected to the network as an APPN node.

### CDRM-CDRM Session Activation

The following sections show examples of session activation based on your configuration.

#### Across a CTC Connection (TPF to TPF, or TPF to VTAM)

To activate the CDRM-CDRM session from the TPF system, enter the following:

```
ZNETW ACT ID=rrrrrrrr
```

Where:

*rrrrrrrr*

The name of the remote CDRM.

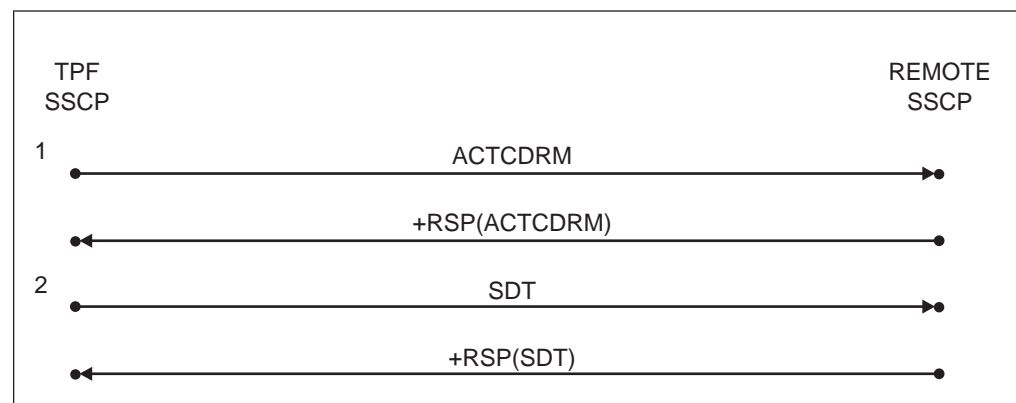


Figure 120. CDRM-CDRM Session Started by the TPF System

Here are the flows when activated by the remote SSCP:

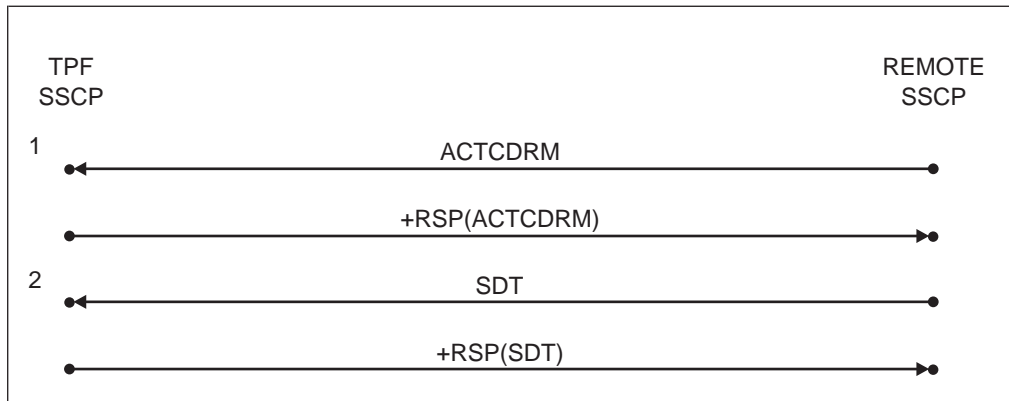


Figure 121. CDRM-CDRM Session Started by the Remote SSCP

### Across an NCP Connection (TPF to VTAM)

To activate the CDRM-CDRM session from VTAM, enter the following:

```
V NET,ACT,ID=rrrrrrrr
```

Where:

*rrrrrrrr*

The name of the TPF CDRM.

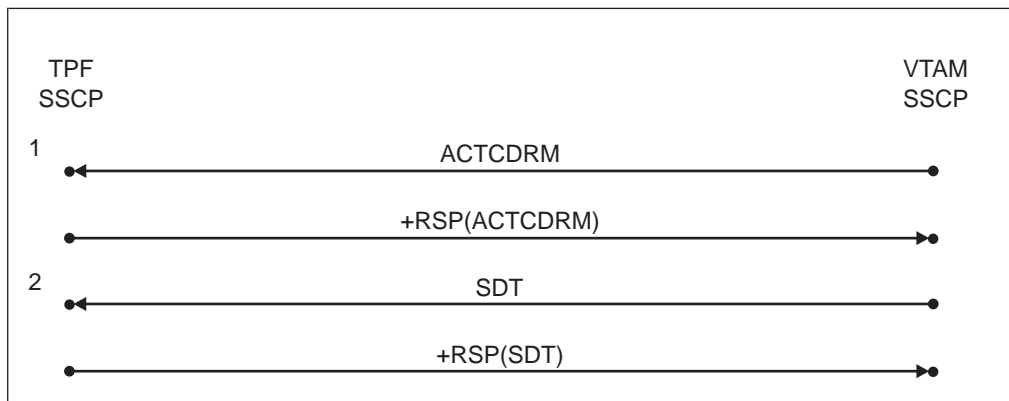


Figure 122. CDRM-CDRM Session Started by VTAM

The CDRM-CDRM session can be activated from the TPF side. The ACTCDRM request sent by TPF will be rejected by the gateway NCP, but then that NCP will forward the request to VTAM who will drive the session. Here are the flows for this case:

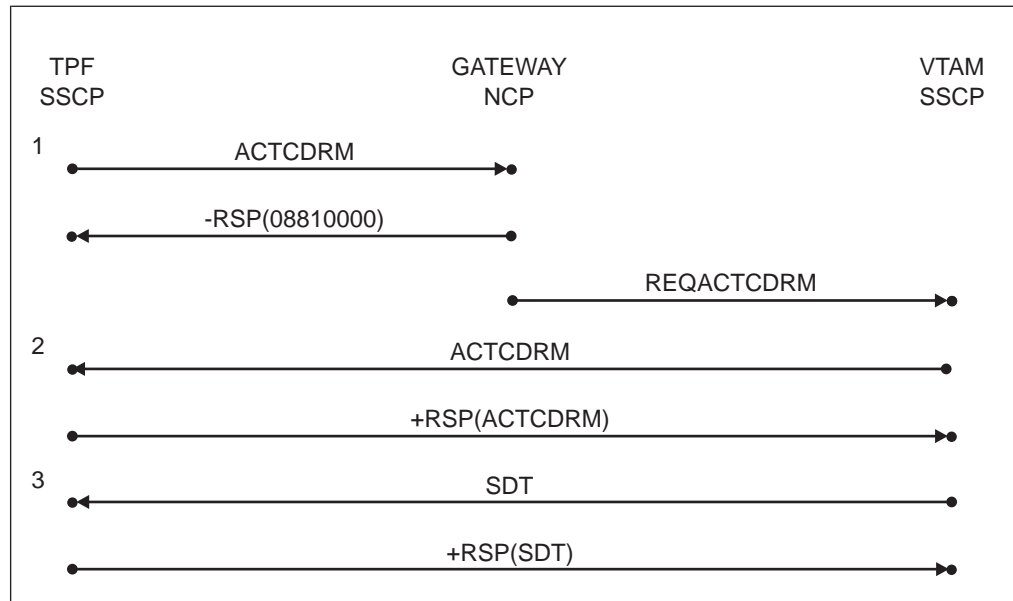


Figure 123. CDRM-CDRM Session Started from the TPF side

### Session Activation Between TPF APPL (PLU) and Remote APPL (SLU)

To activate the APPL-APPL session from the TPF system, enter the following:

```
ZNETW ACT ID=ssssssss LOGON=pppp CDRM=cccccccc
```

Where:

*ssssssss*

The name of the remote APPL

*pppp*

The name of the TPF APPL

*cccccccc*

The name of remote CDRM.

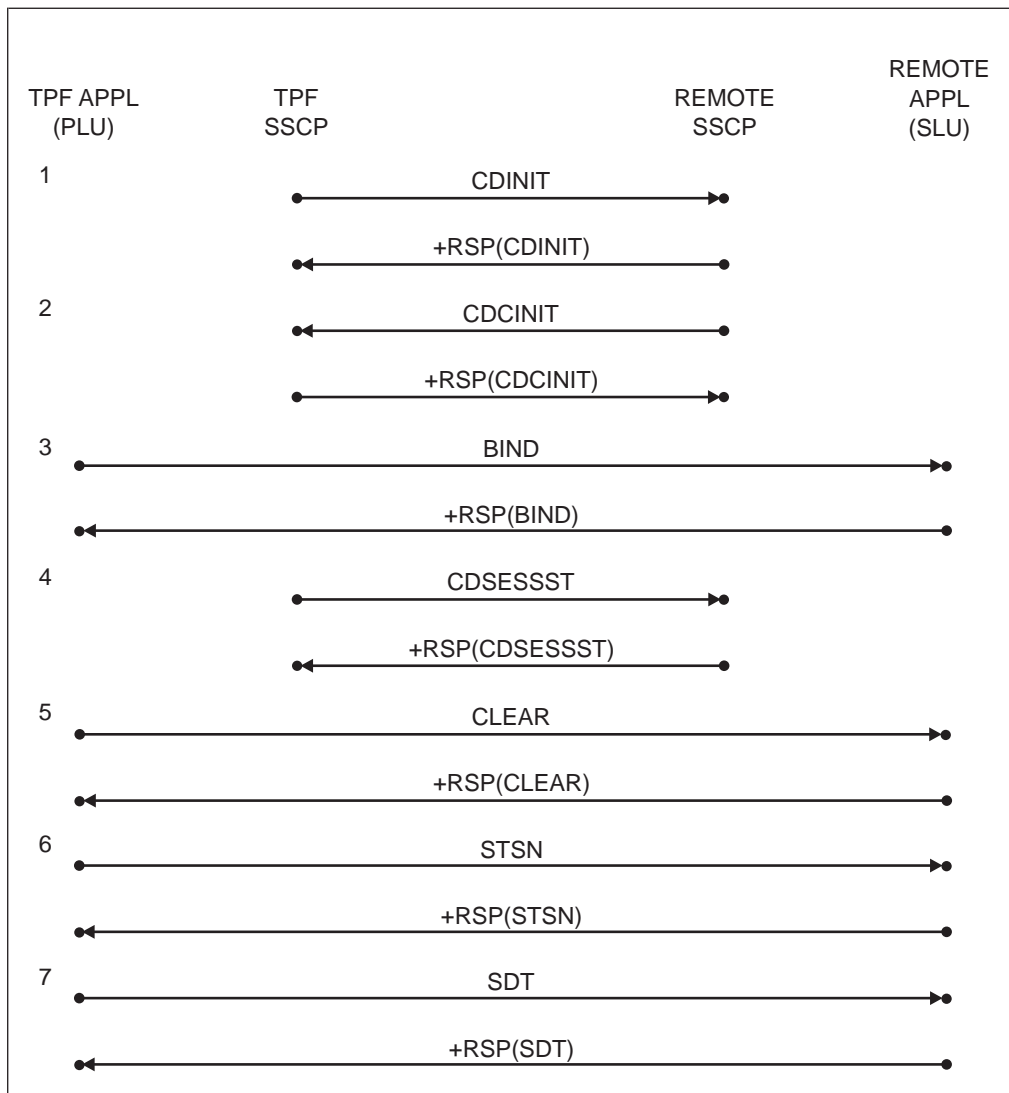


Figure 124. APPL-APPL Session Started by TPF (PU 5)

**Note:** If the normal flow sequence numbers in the RVT (RV2SISEQ and RV2SOSEQ) are both zero, then CLEAR and STSN are not sent; only SDT is sent on the LU-LU session following the BIND. This applies to all the flow diagrams in this section.

Here are the flows when the session is started from the remote side:



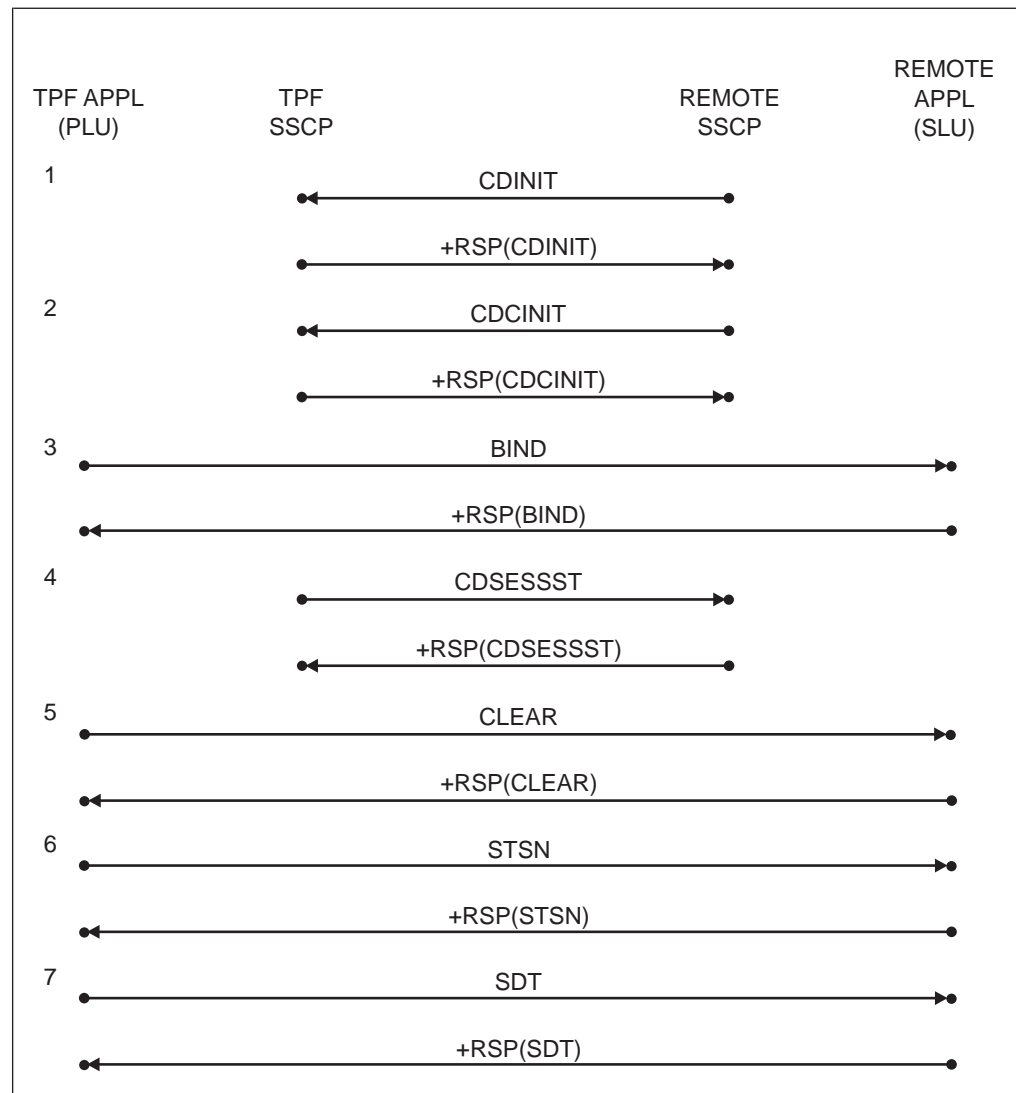


Figure 125. APPL-APPL Session Started by Remote LU (PU 5)

If the remote SSCP is VTAM, then it is possible that the CDINIT request sent (step #1) is a CDINIT Queue Only request. In this case a CDINIT Dequeue request must be sent before CDCINIT. Here are the flows:

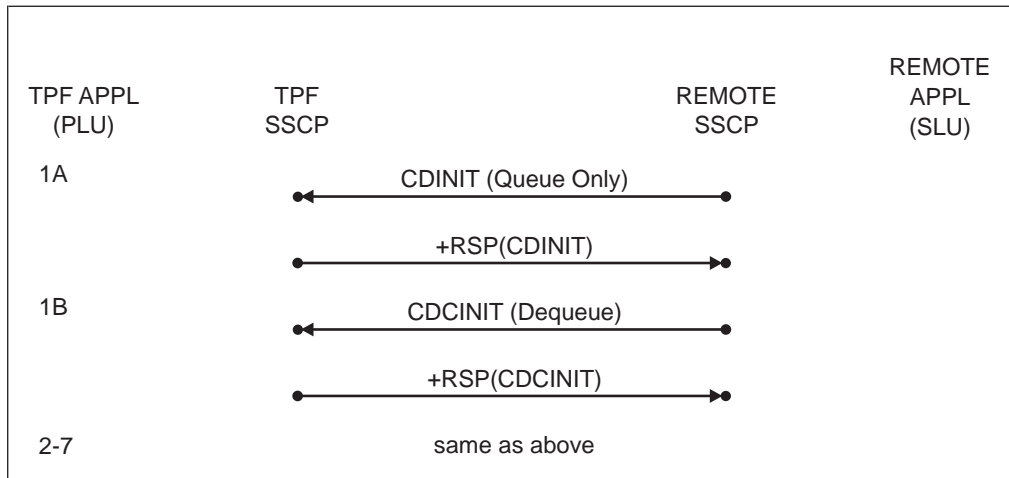


Figure 126. APPL-APPL Session Started by Remote LU (PU 5) and is Queued

For PU 2.1 LEN, the session can only be started from the remote side and requires the use of the Logon Manager in VTAM. Here are the flows:

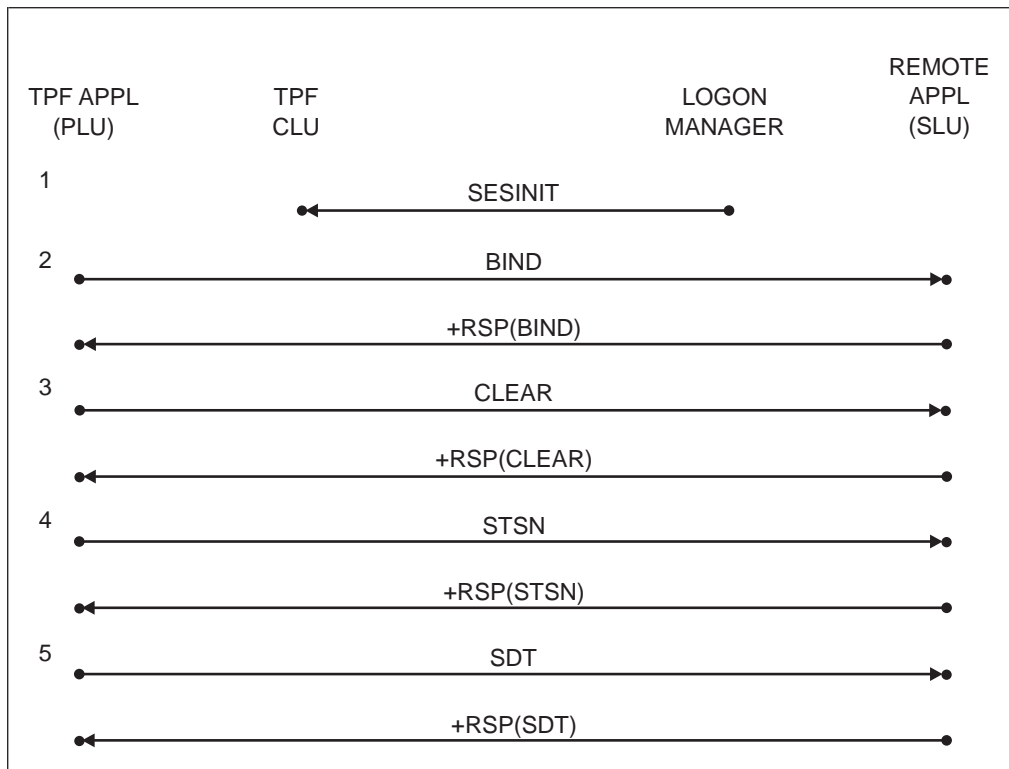


Figure 127. APPL-APPL Session Started by Remote LU (PU 2.1 LEN)

**Note:** If the normal flow sequence numbers in the RVT (RV2SISEQ and RV2SOSEQ) are both zero, then CLEAR and STSN are not sent; only SDT is sent on the LU-LU session following the BIND.

### Session Activation for Host-Node SLU Session

To activate the host-node SLU session from the TPF system, enter the following:

```
ZNETW ACT ID=ssssssss LOGON=pppp CDRM=cccccccc
```

Where:

ssssssss

The name of the remote APPL

pppp

The name of the TPF APPL

cccccccc

The name of the remote CDRM.

This message brings up a session between the remote APPL and each TPF SLU-thread of APPL *pppp* that is not already in session.

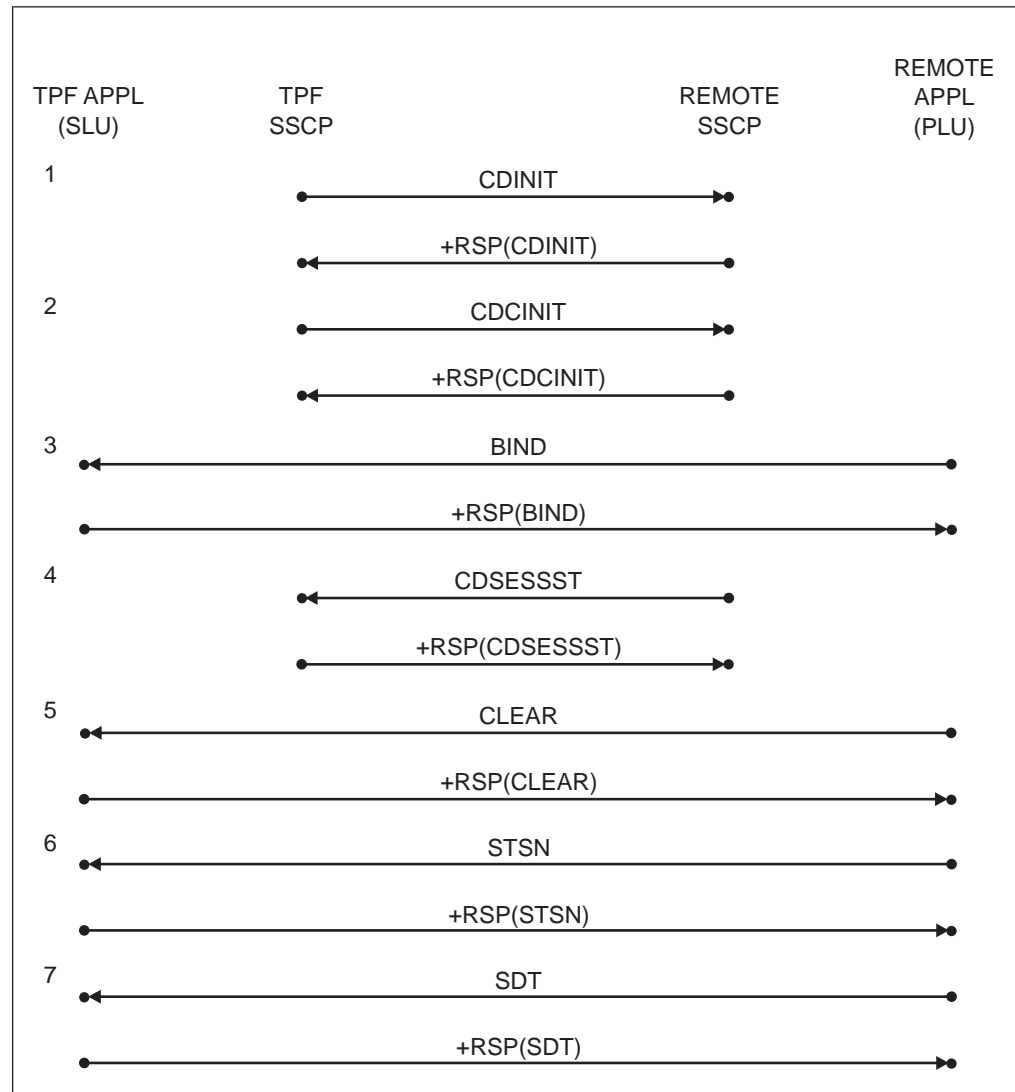


Figure 128. Host-Node SLU Session Started by TPF (PU 5)

**Note:** The remote PLU may not send CLEAR and STSN following the BIND. It is possible that only SDT is sent on the LU-LU session following the BIND.

For remote initiated PU 5 sessions, the only difference in the flows is that the remote side sends the CDINIT request (step #1 in the previous figure). Also, the remote side may send a CDINIT Queue Only request, in which case a CDINIT Dequeue request must also be sent by the remote side before the TPF system can send CDCINIT.

For PU2.1 LEN, the session can only be started from the remote side. Here are the flows:

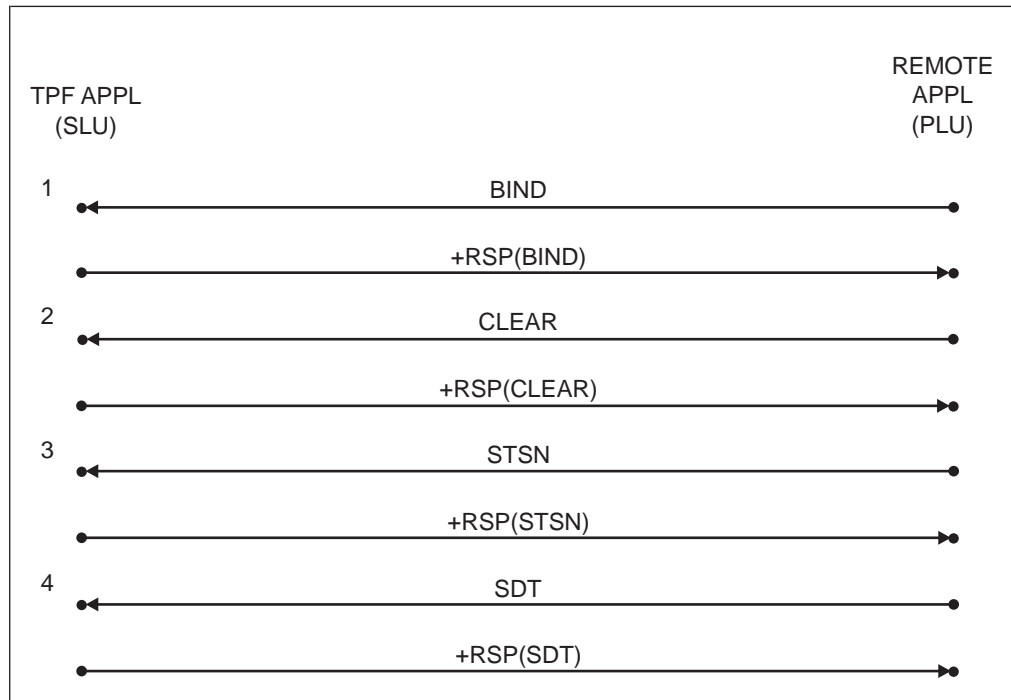


Figure 129. Host-Node SLU Session Started by Remote LU (PU 2.1 LEN)

### FMMR-FMMR Session Activation

To activate the FMMR-FMMR session, enter the following:

```
ZNETW ACT ID=rrrrrrrr LOGON=llllllll CDRM=cccccccc
```

Where:

*rrrrrrrr*

The name of the remote FMMR

*llllllll*

The name of the local FMMR

*cccccccc*

The name of the remote CDRM.

For FMMR-FMMR sessions, the PLU is the FMMR whose LU name is greater. Here are the flows when initiated from the side owning the FMMR PLU:

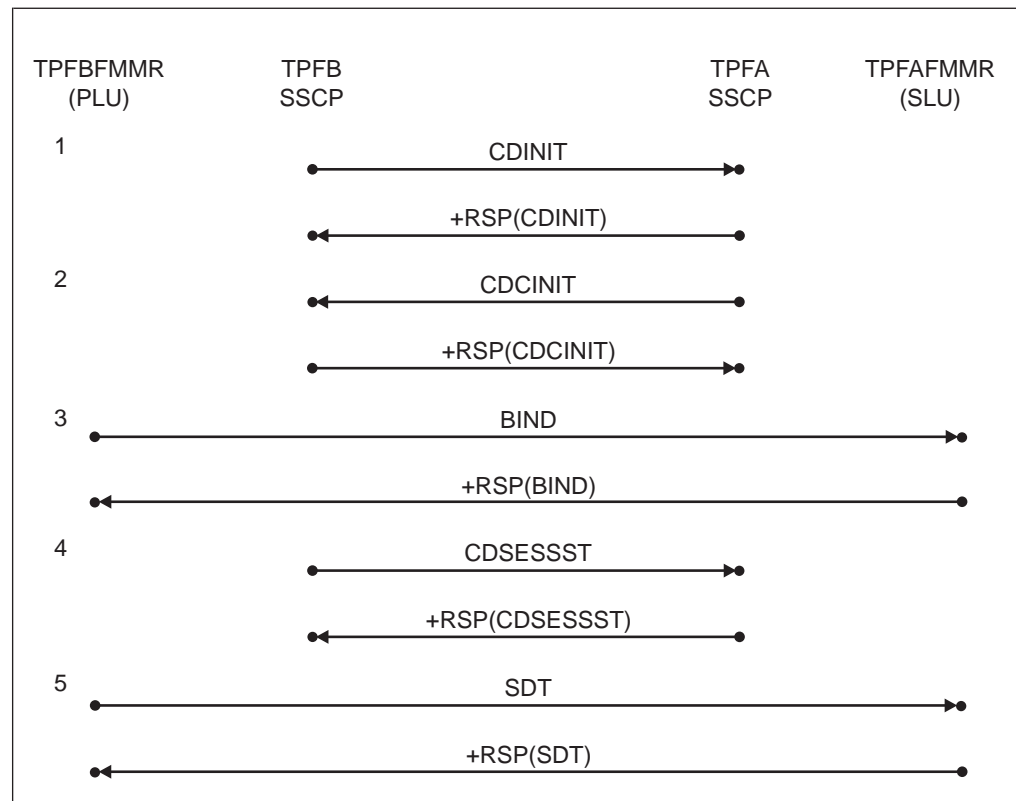


Figure 130. PU 5 FMMR-FMMR Session Initiation from the PLU

Here are the flows when initiated from the side owning the FMMR SLU:

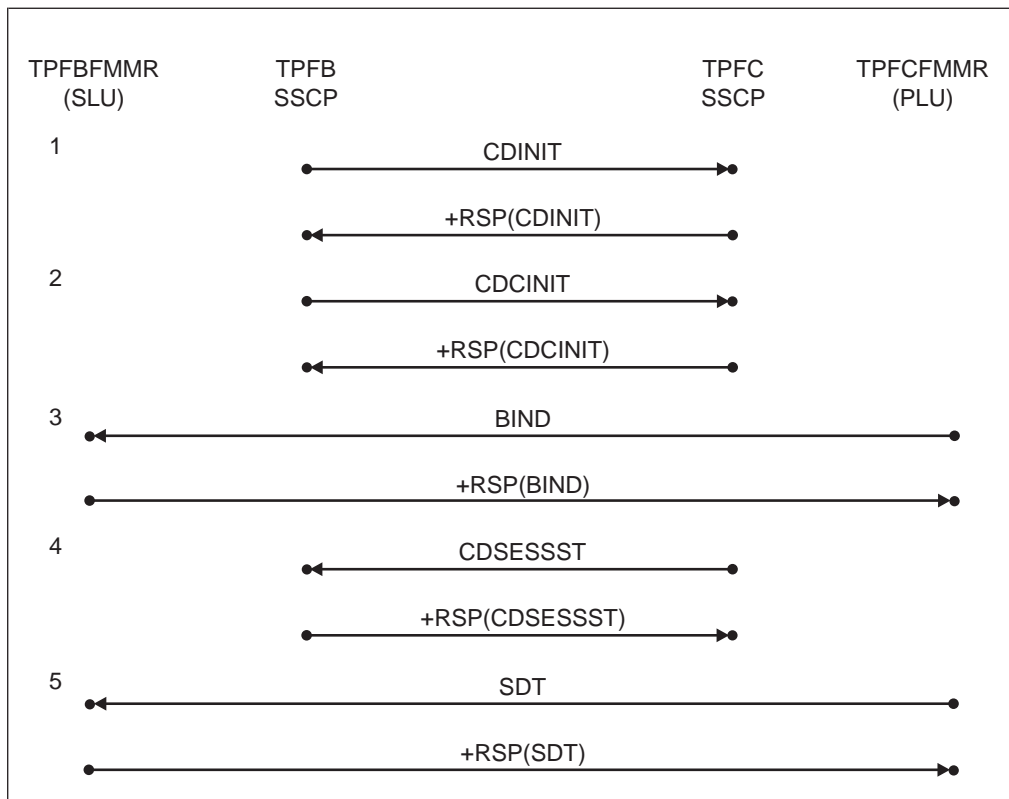


Figure 131. PU 5 FMMR-FMMR Session Initiation from the SLU

For PU 2.1 LEN, the session must be started from VTAM and use the Logon Manager. Here are the flows:

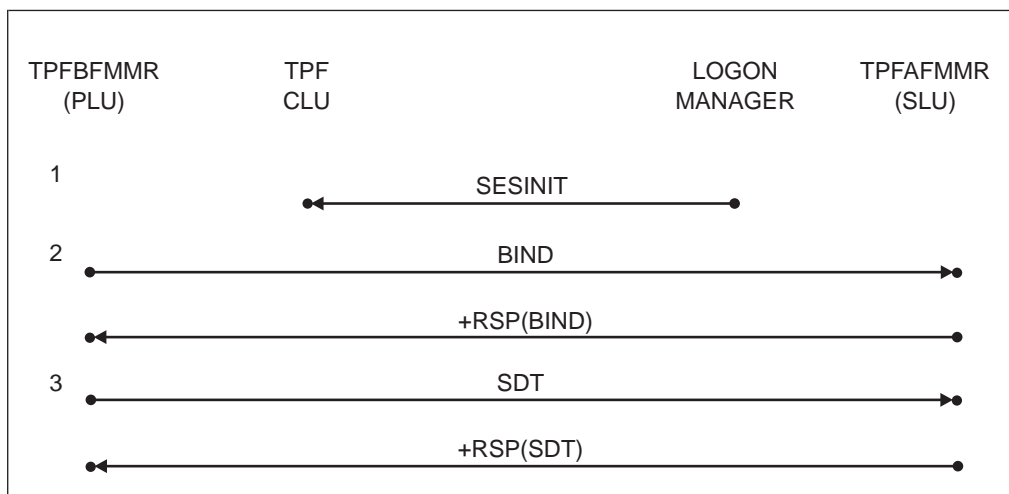


Figure 132. PU 2.1 LEN FMMR-FMMR Session Initiation

### Session Activation between TPF APPL and Remote 3270

Here are the flows for when the remote terminal logs on to the TPF APPL:

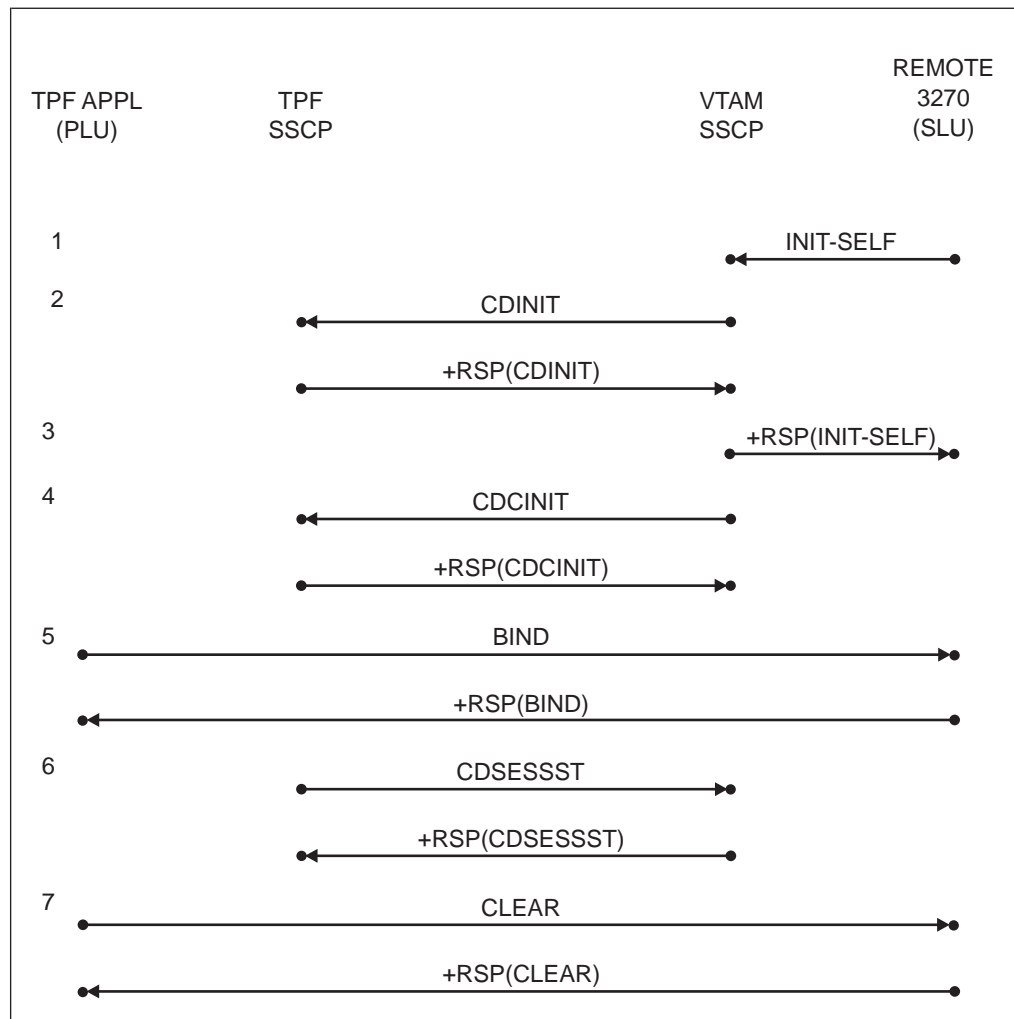


Figure 133. 3270 Session Started by Remote Terminal (PU 5)

Here are the PU2.1 LEN flows for when the remote terminal logs on to the TPF APPL:

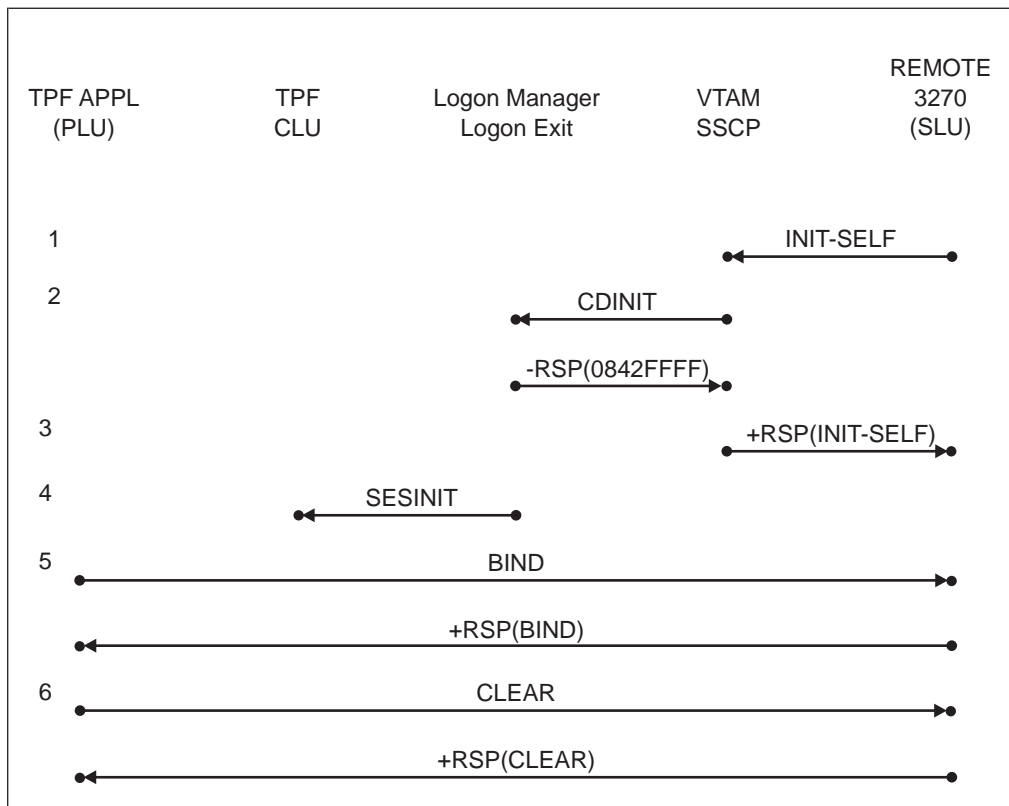


Figure 134. 3270 Session Started by Remote Terminal (PU 2.1 LEN)

### Session Activation for PU 5 LU 6.2 Sessions (Started by TPF PLU)

The following are the flows when the TPF system is the primary LU (PLU) and activates a new session (parallel or single) in a PU 5 environment. The session is activated by an ALLOCATE verb. In step #5, a FLUSH verb is issued causing the buffered ATTACH (FMH5) to be transmitted to the remote LU.



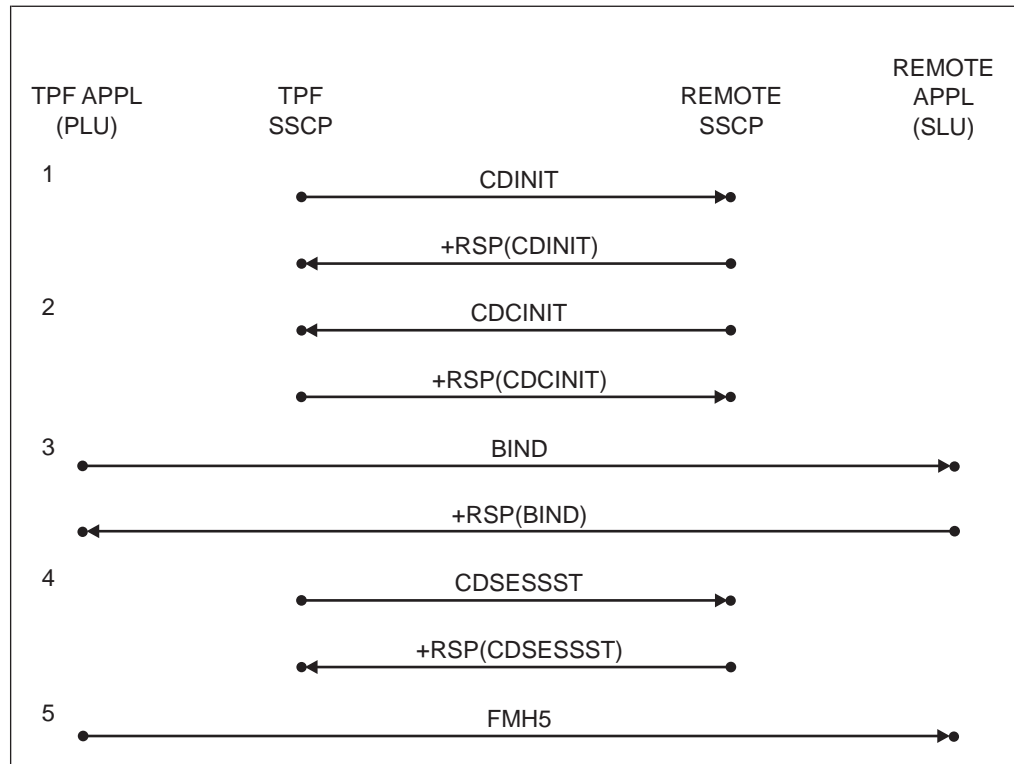


Figure 135. PU 5 LU 6.2 Session Started by TPF PLU

Here are the flows when the session is started from the remote side:

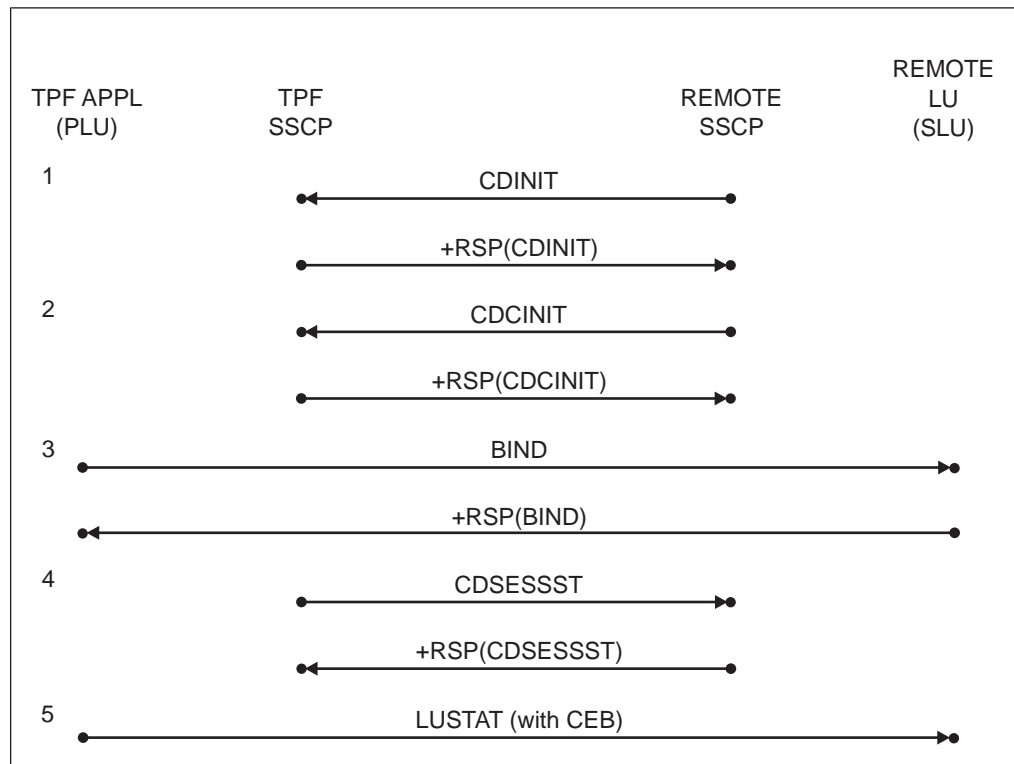


Figure 136. PU 5 LU 6.2 Session Started by Remote SLU

## Session Activation for PU 5 LU 6.2 Sessions (Started by TPF SLU)

The following are the flows when a TPF secondary LU (SLU) thread activates a new single session in a PU 5 environment. The session is activated by an ALLOCATE verb. In step #6, a FLUSH verb is issued causing the buffered ATTACH (FMH5) to be transmitted to the remote LU. This example assumes the TPF system is the contention winner.

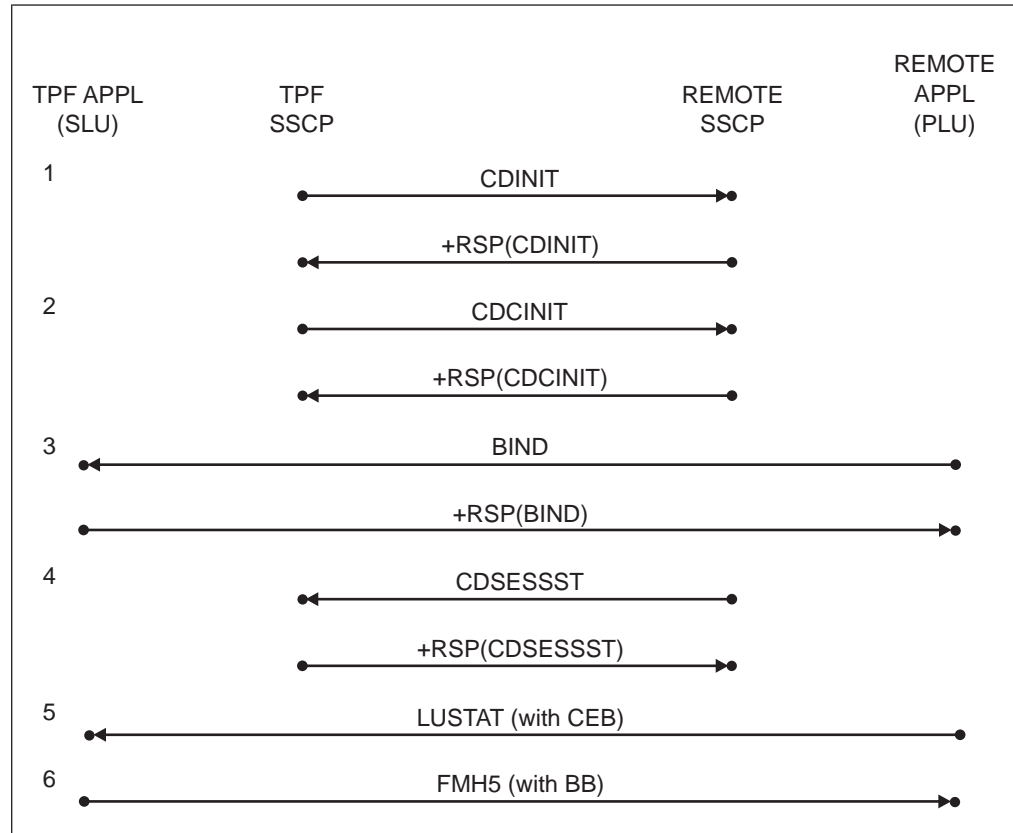


Figure 137. PU 5 LU 6.2 Session Started by TPF SLU

Here are the flows when the session is started from the remote side. If the remote LU started the session with an ALLOCATE request, the flow is step #5A; otherwise, the flow is step #5B.

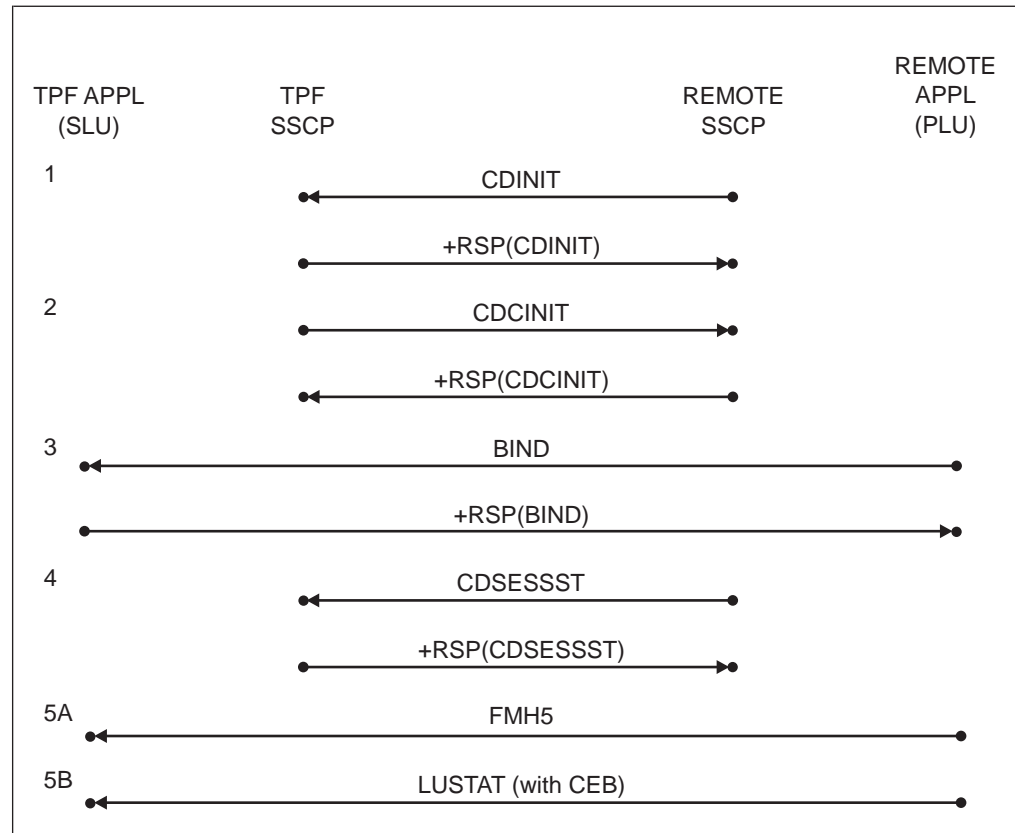


Figure 138. PU 5 LU 6.2 Session Started by Remote PLU

### Session Activation for PU 2.1 LEN LU 6.2 Sessions

The following are the flows when the TPF system activates a new session (parallel or single) that is not using an SLU thread and is in a PU 2.1 LEN environment. The session is activated by an ALLOCATE verb. In step #4, a FLUSH verb is issued causing the buffered ATTACH (FMH5) to be transmitted to the remote LU.

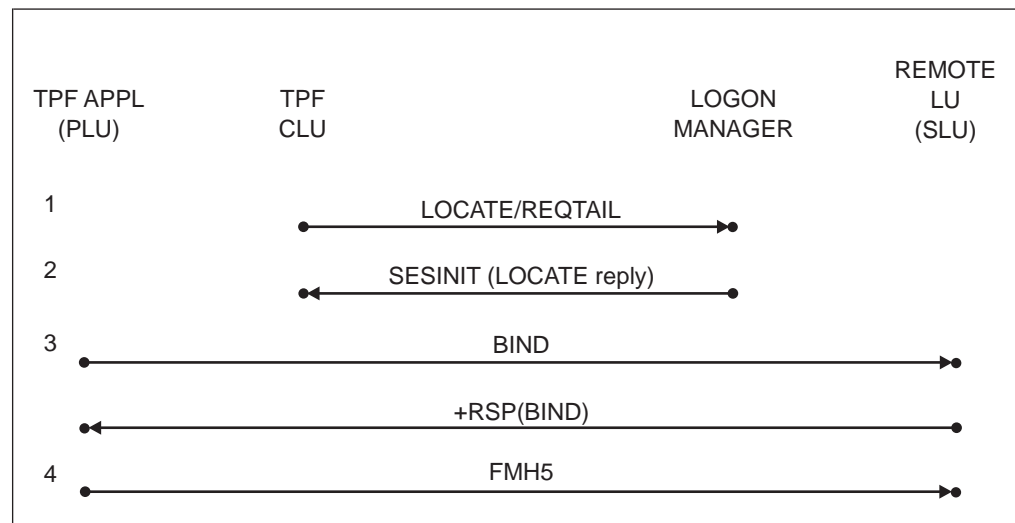


Figure 139. PU 2.1 LEN LU 6.2 Session Started by TPF

The following flows show a single session being started by a remote SLU (dependent LU).

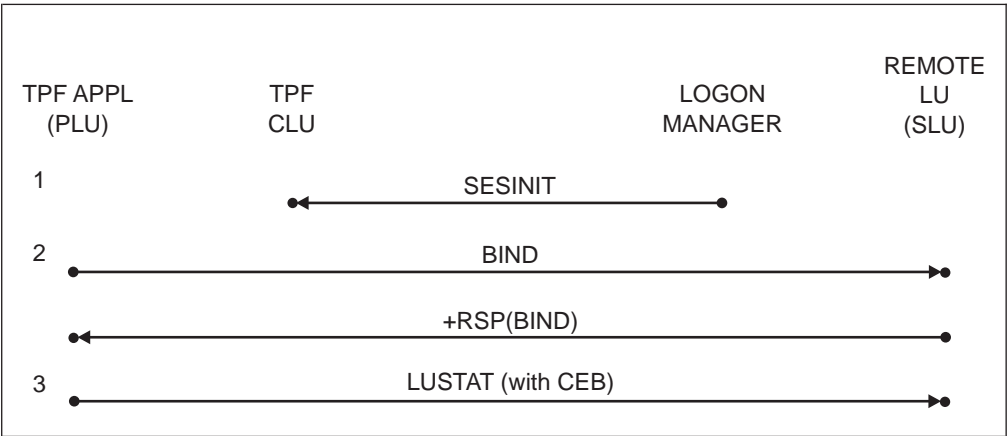


Figure 140. PU 2.1 LEN LU 6.2 Session Started by Remote SLU

**Note:** If the remote LU is an independent LU, then it sends a BIND directly to the TPF system.

## PU 5 and PU 2.1 Session Deactivation

The following sections show examples of session deactivations.

### CDRM-CDRM Session Deactivation

The following sections show different ways to deactivate a CDRM-CDRM session.

**Normal CDRM-CDRM Session Deactivation:** To deactivate the CDRM-CDRM session from the TPF system, enter the following:

```
ZNETW ACT ID=rrrrrrrr
```

Where:

*rrrrrrrr*

The name of the remote CDRM.

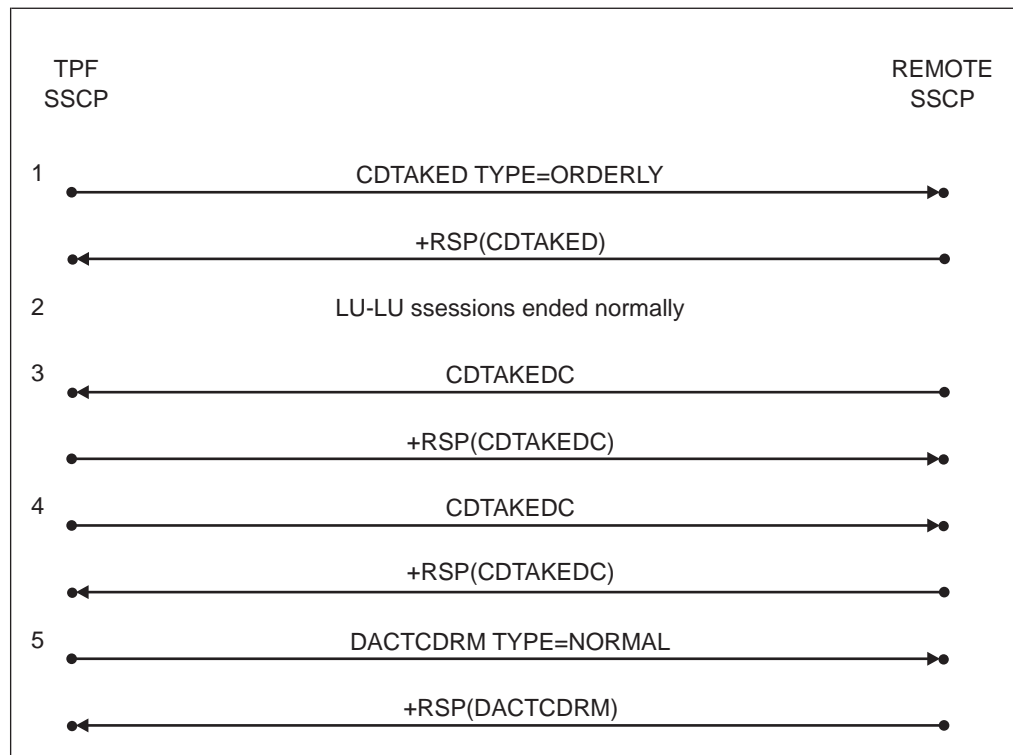


Figure 141. Normal CDRM-CDRM Session Deactivation

**Note:** Either SSCP can send CDTAKED. After all initiating and active LU-LU sessions have been ended, the SSCP receiving the CDTAKED request sends CDTAKEDC first. Next, the SSCP that sent CDTAKED sends a CDTAKEDC followed by a DACTCDRM.

**Immediate CDRM-CDRM Session Deactivation:** To deactivate the CDRM-CDRM session from the TPF system, enter the following:

```
ZNETW ACT ID=rrrrrrrr,I
```

Where:

*rrrrrrrr*

The name of the remote CDRM.

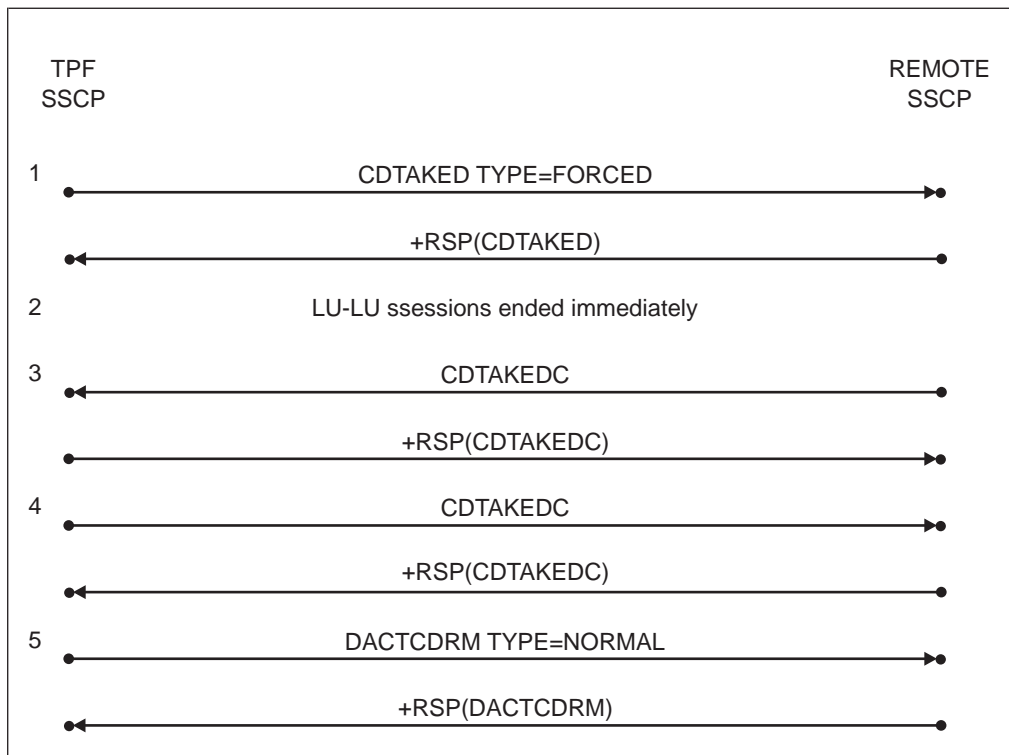


Figure 142. Immediate CDRM-CDRM Session Deactivation

**Note:** Either SSCP can send CDTAKED. After all initiating and active LU-LU sessions have been ended, the SSCP receiving the CDTAKED request sends CDTAKEDC first. Next the SSCP that sent CDTAKED sends a CDTAKEDC followed by a DACTCDRM.

To deactivate the CDRM-CDRM non-disruptively, enter the following:

```
ZNETW ACT ID=rrrrrrrr,I,SAVESESS
```

Where:

*rrrrrrrr*

The name of the remote CDRM.

In this case, the flows are identical except that active LU-LU sessions are not ended (step #2); only the LU-LU sessions that are in the process of being activated are ended.

**Forced CDRM-CDRM Session Deactivation:** To deactivate the CDRM-CDRM session from the TPF system, enter the following:

```
ZNETW ACT ID=rrrrrrrr,F
```

Where:

*rrrrrrrr*

The name of the remote CDRM.

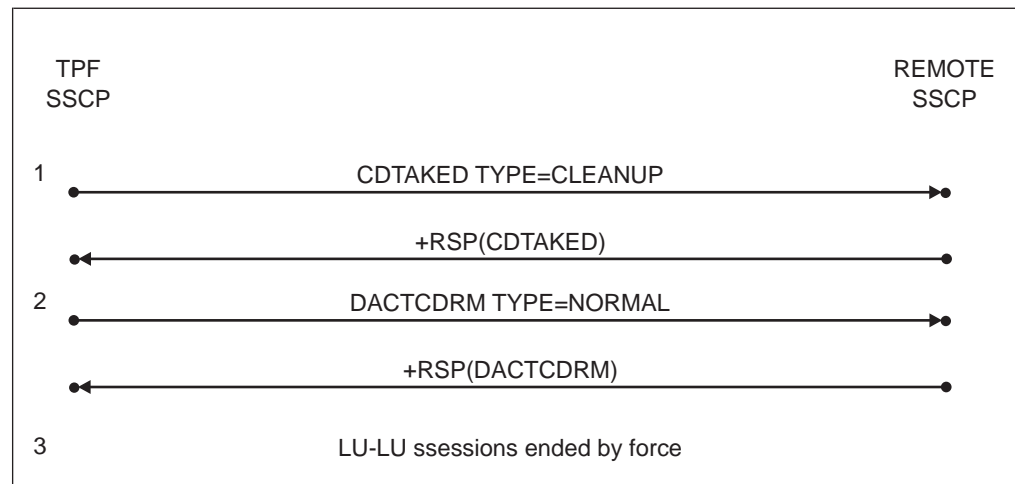


Figure 143. Forced CDRM-CDRM Session Deactivation

**Note:** Either SSCP can send CDTAKED. There are no CDTAKEDC exchanges here. DACTCDRM is sent immediately after receiving a CDTAKED response, and then the LU-LU sessions are ended. For active LU-LU sessions, UNBIND is the only command that flows (a CDSESEND cannot be sent because the CDRM-CDRM session no longer exists). LU-LU sessions that are in the process of being activated are cleaned up without any flows between the LUs.

To deactivate the CDRM-CDRM non-disruptively, enter the following:

```
ZNETW ACT ID=rrrrrrrr,F,SAVESESS
```

Where:

*rrrrrrrr*

The name of the remote CDRM.

In this case, the flows are identical except that active LU-LU are not ended (step #3); only the LU-LU sessions that in the process of being activated are ended.

### Session Deactivation for TPF APPL (PLU) and Remote APPL (SLU)

The following sections show different ways to deactivate an APPL-APPL session.

**Normal Deactivation of APPL-APPL Session:** To deactivate an APPL-APPL session normally from the TPF system, enter the following:

```
ZNETW INACT ID=ssssssss
```

Where:

*ssssssss*

The name of the remote APPL.

To deactivate all sessions with the TPF APPL, enter the following:

```
ZNETW INACT ID=pppp
```

Where:

*pppp*

The name of the TPF APPL.

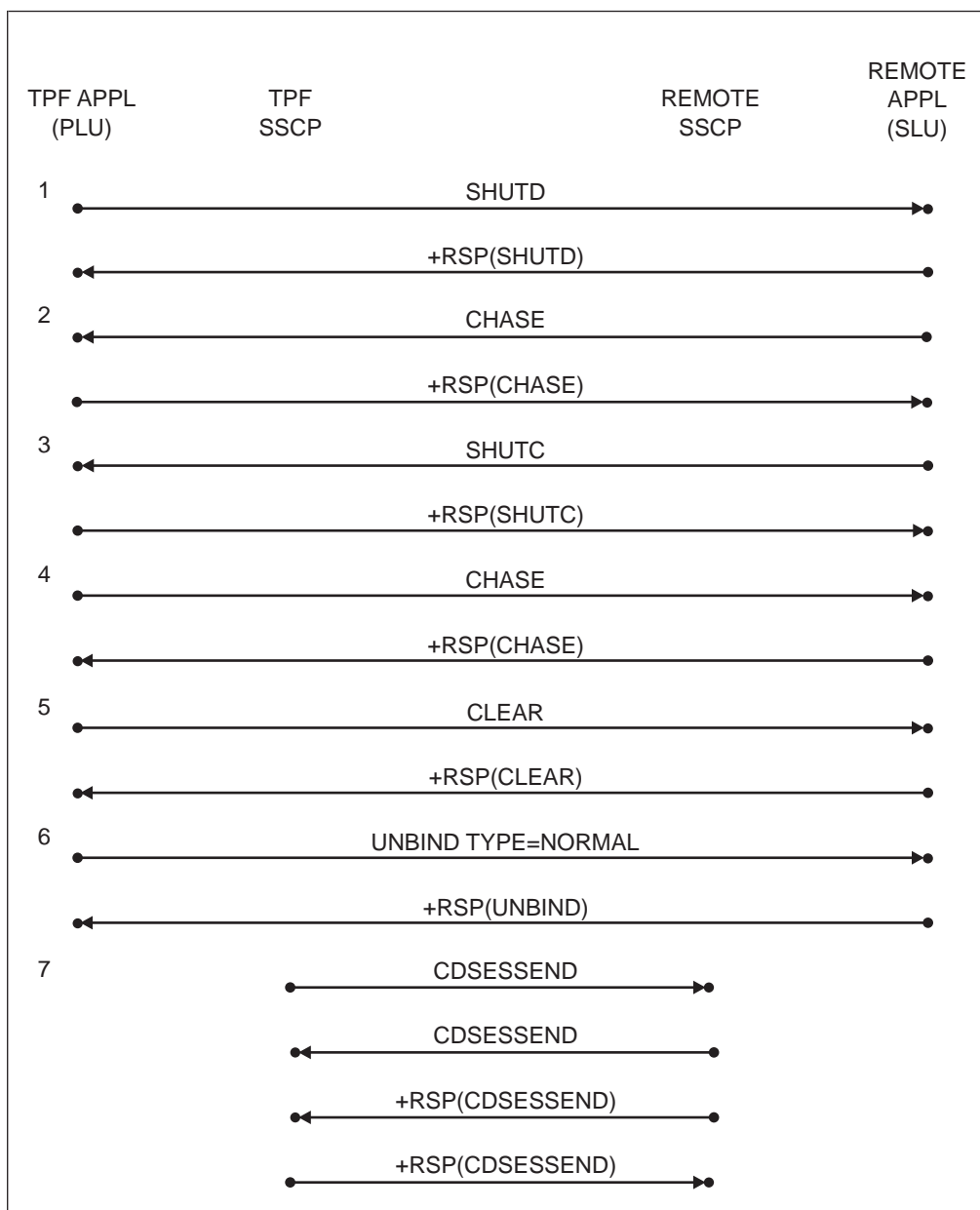


Figure 144. Normal APPL-APPL Session Deactivation

**Notes:**

1. Responses to normal data traffic can flow before step #5. Because the CHASE command is normal flow, once a response to CHASE has been received the LU knows it is not waiting for any more responses to normal flow traffic. The remote LU does not have to send a CHASE request, so step #2 is optional.
2. For PU2.1 sessions, no CDSSESENDs are sent (skip step #7).

**Immediate APPL-APPL Session Deactivation:** To deactivate an APPL-APPL session immediately from the TPF system, enter the following:

```
ZNETW INACT ID=sssssss,I
```

Where:



ssssssss

The name of the remote APPL.

To deactivate all sessions with the TPF APPL, enter the following:

```
ZNETW INACT ID=pppp,I
```

Where:

pppp

The name of the TPF APPL.

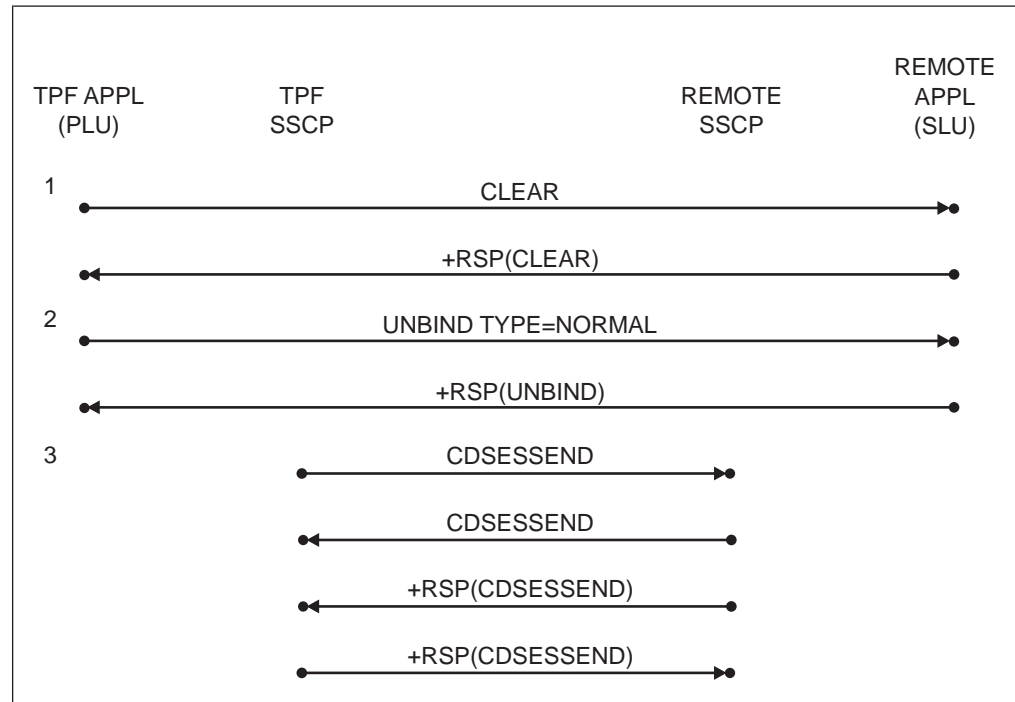


Figure 145. Immediate APPL-APPL Session Deactivation

**Note:** For PU2.1 sessions, no CDESENSENDs are sent (skip step #3).

**Forced APPL-APPL Session Deactivation:** To deactivate an APPL-APPL session with force from the TPF system, enter the following:

```
ZNETW INACT ID=ssssssss,F
```

Where:

ssssssss

The name of the remote APPL.

To deactivate all sessions with the TPF APPL, enter the following:

```
ZNETW INACT ID=pppp,F
```

Where:

pppp

The name of the TPF APPL.

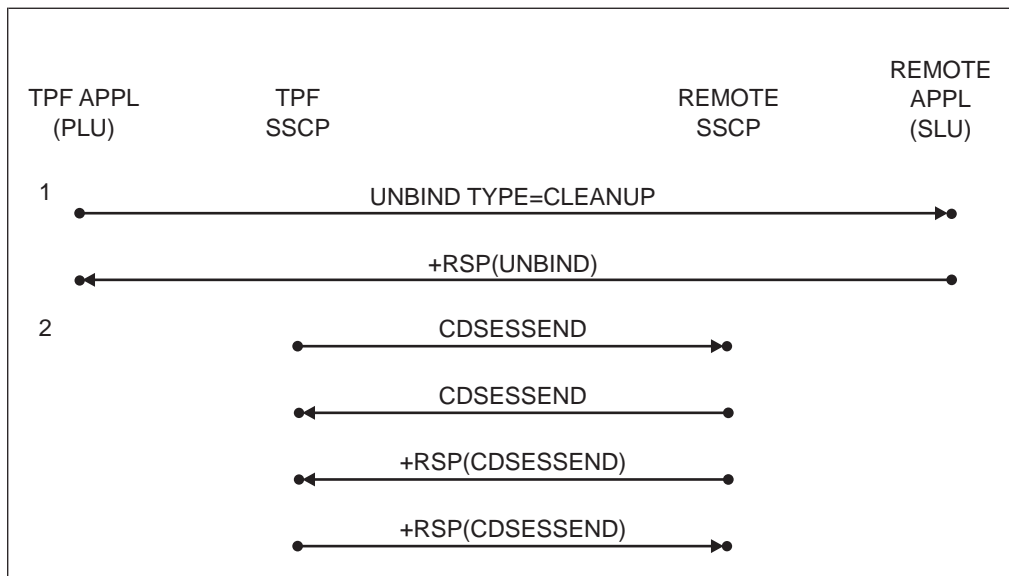


Figure 146. Forced APPL-APPL Session Deactivation

**Note:** For PU2.1 sessions, no CDSESENDS are sent (skip step #2).

### Host-Node SLU Session Deactivation

The following sections show different ways to deactivate a host-node SLU session.

**Normal Deactivation of Host-Node SLU Session:** To deactivate a host-node SLU session normally from the TPF system, enter the following:

```
ZNETW INACT ID=ssssssss
```

Where:

ssssssss

The name of the TPF SLU thread.

To deactivate all sessions with the remote APPL, enter the following:

```
ZNETW INACT ID=pppppppp
```

Where:

pppppppp

The name of the remote APPL.

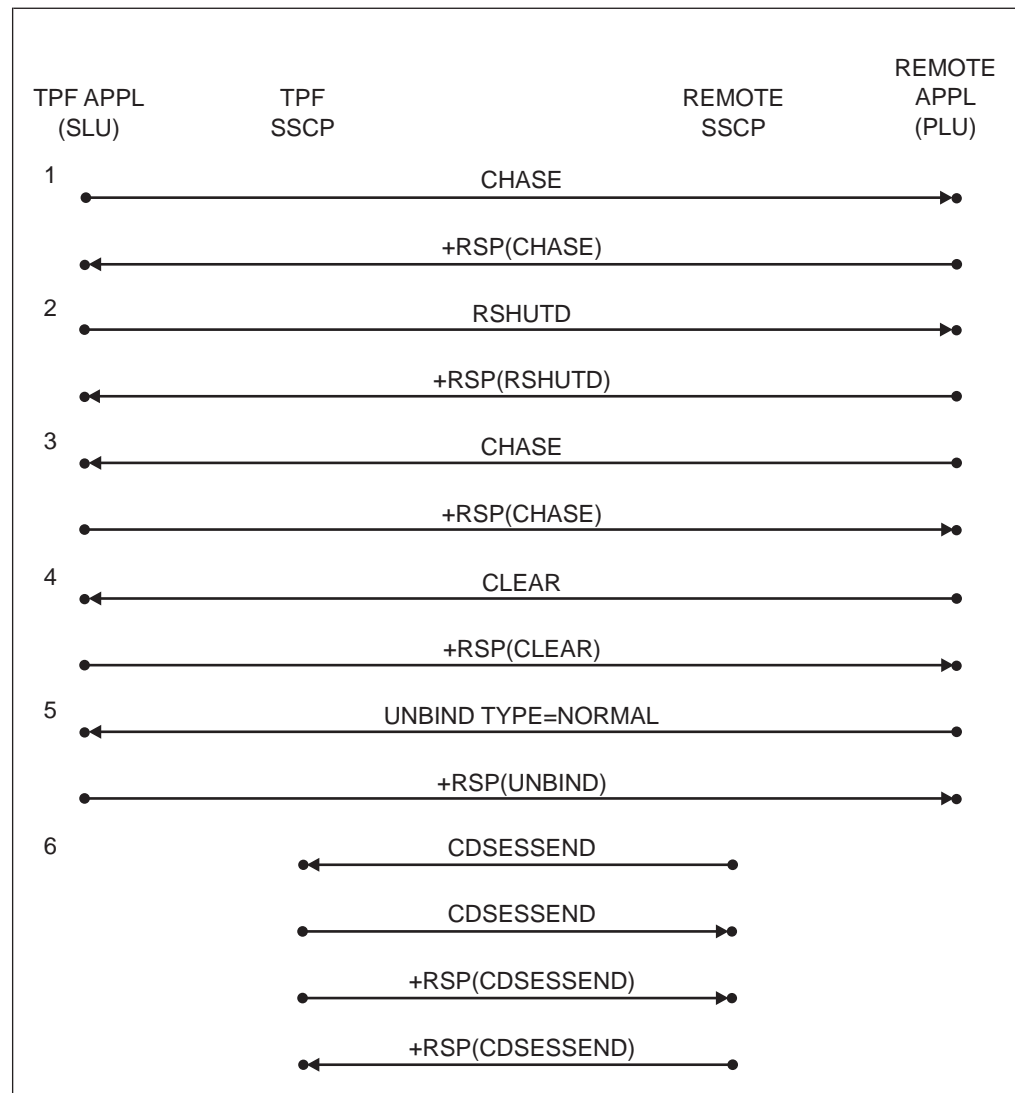


Figure 147. Normal Host-Node SLU Session Deactivation

#### Notes:

- Responses to normal data traffic can flow before step #4. Because the CHASE command is normal flow, once a response to CHASE has been received the LU knows it is not waiting for any more responses to normal flow traffic. The remote LU does not have to send a CHASE request, so step #3 is optional.
- For PU2.1 sessions, no CDSESENDS are sent (skip step #6).

**Immediate Deactivation of Host-Node SLU Session:** To deactivate a host-node SLU session immediately from the TPF system, enter the following:

```
ZNETW INACT ID=ssssssss,I
```

Where:

ssssssss

The name of the TPF SLU thread.

To deactivate all sessions with the remote APPL, enter the following:

```
ZNETW INACT ID=pppppppp,I
```

Where:

*pppppppp*

The name of the remote APPL.

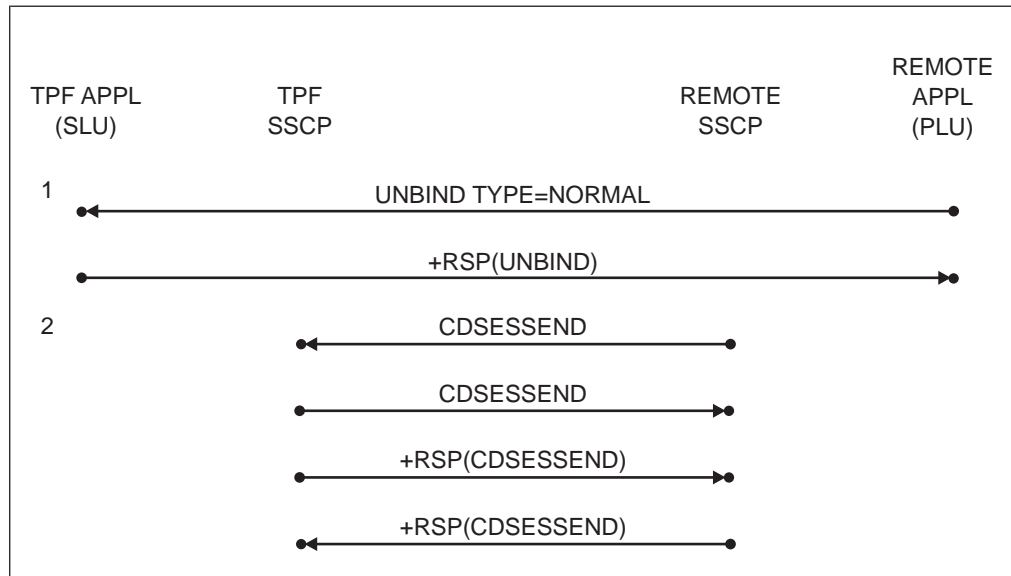


Figure 148. Immediate Host-Node SLU Session Deactivation

**Note:** For PU2.1 sessions, no CDSESENDS are sent (skip step #2).

**Forced Host-Node SLU Session Deactivation:** To deactivate a host-node SLU session with force from the TPF system, enter the following:

```
ZNETW INACT ID=sssssss,F
```

Where:

*sssssss*

The name of the TPF SLU thread.

To deactivate all sessions with the remote APPL, enter the following:

```
ZNETW INACT ID=pppppppp,F
```

Where:

*pppppppp*

The name of the remote APPL.

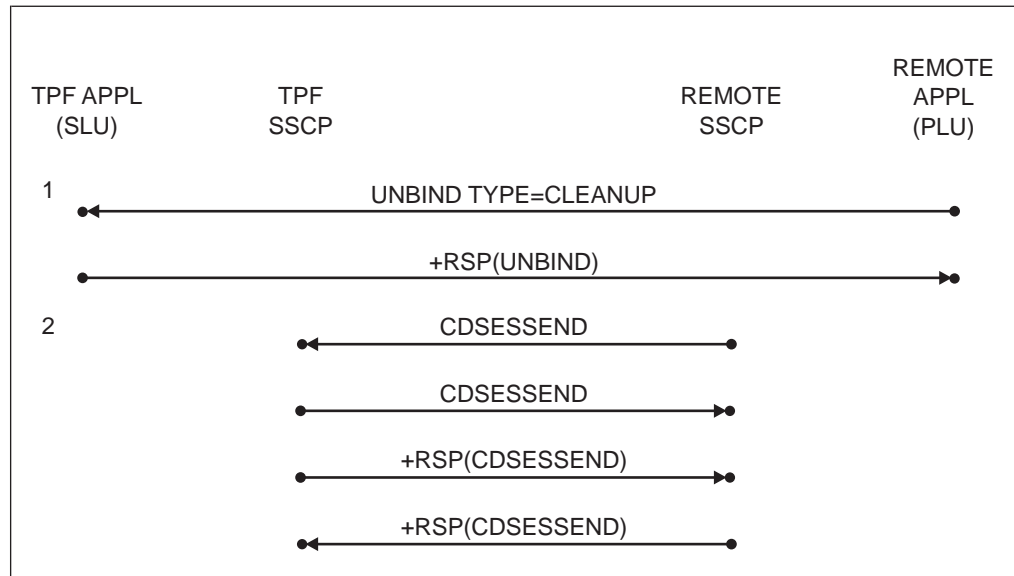


Figure 149. Forced Host-Node SLU Session Deactivation

**Note:** For PU2.1 sessions, no CDSESENDS are sent (skip step #2).

### FMMR-FMMR Session Deactivation

The following sections show different ways to deactivate an FMMR-FMMR session.

**Normal or Immediate Deactivation of FMMR-FMMR Session:** To deactivate an FMMR-FMMR session normally from the TPF system, enter the following:

```
ZNETW INACT ID=rrrrrrrr
```

To deactivate an FMMR-FMMR session immediately from the TPF system, enter the following:

```
ZNETW INACT ID=rrrrrrrr,I
```

Where:

*rrrrrrrr*

The name of the remote FMMR.

To deactivate all FMMR sessions, enter the following:

```
ZNETW INACT ID=llllllll
```

or

```
ZNETW INACT ID=llllllll,I
```

Where:

*llllllll*

The name of the local FMMR.

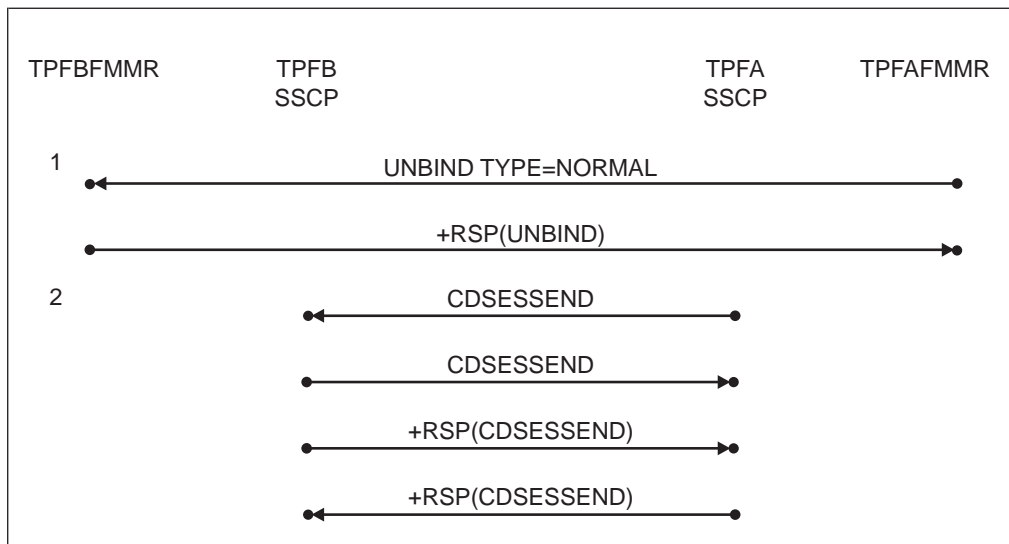


Figure 150. Normal or Immediate FMMR-FMMR Session Deactivation

**Note:** For PU2.1 sessions, no CDSSESENDs are sent (skip step #2).

**Forced FMMR-FMMR Session Deactivation:** To deactivate an FMMR-FMMR session with force from the TPF system, enter the following:

```
ZNETW INACT ID=rrrrrrrr,F
```

Where:

*rrrrrrrr*

The name of the remote FMMR.

To deactivate all FMMR sessions, enter the following:

```
ZNETW INACT ID=llllllll,F
```

Where:

*llllllll*

The name of the local FMMR.

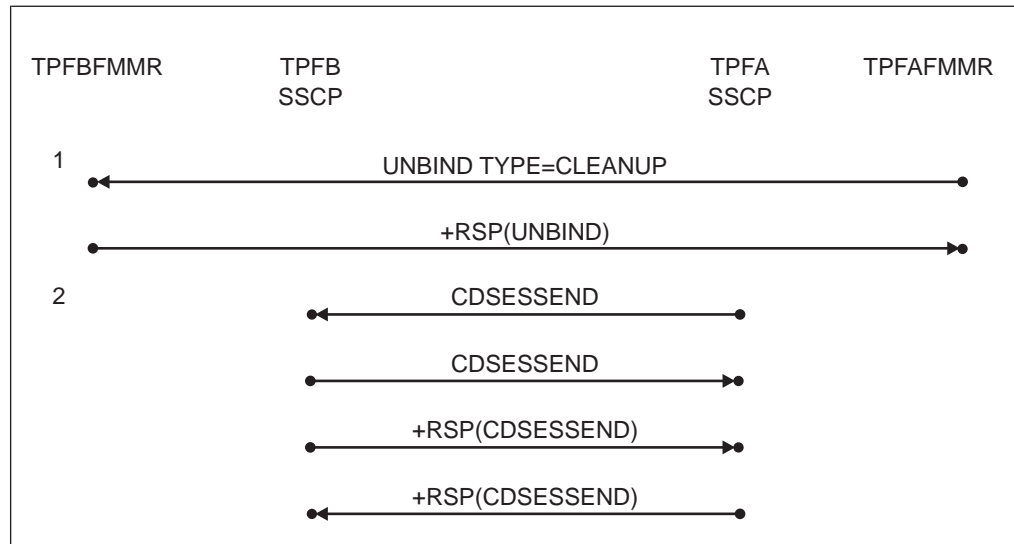


Figure 151. Forced FMMR-FMMR Session Deactivation

**Note:** For PU2.1 sessions, no CDESENDDs are sent (skip step #2).

## LU 6.2 Session Deactivation

If you deactivate an LU 6.2 session normally, the session ends after the active conversation, if any, finishes. The other types of deactivation cause the session to end without waiting for an active conversation to finish.

**Normal LU 6.2 Session Deactivation:** To deactivate LU 6.2 sessions normally for a particular remote LU, enter the following from the TPF system:

```
ZNETW INACT ID=ssssssss
```

or

```
ZNCNS RESET LU=ssssssss,MODE=nnnnnnnn
```

Where:

*ssssssss*

The name of the remote LU

*nnnnnnnn*

The mode name.

To deactivate all LU 6.2 sessions normally between remote LUs and a specific TPF/APPC local LU, enter the following:

```
ZNETW INACT ID=pppp
```

Where:

*pppp*

The name of the local TPF/APPC LU.

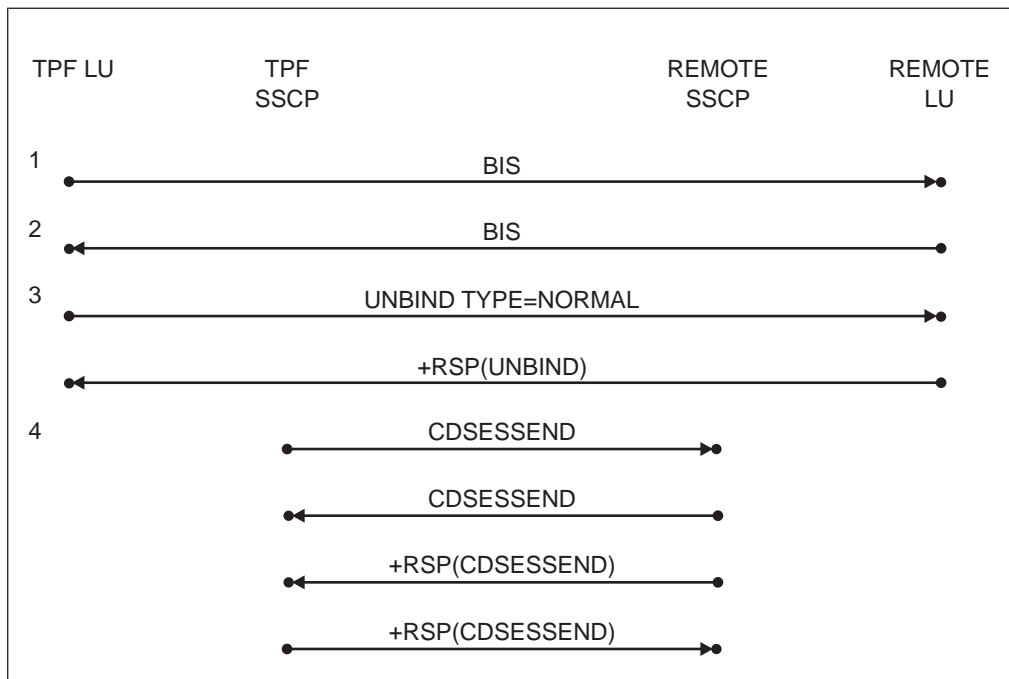


Figure 152. Normal LU 6.2 Session Deactivation

**Note:** For PU2.1 sessions, no CDSSESENDs are sent (skip step #4).

**Immediate LU 6.2 Session Deactivation:** To deactivate all LU 6.2 sessions immediately for a particular remote LU, enter the following from the TPF system:

```
ZNETW INACT ID=sssssss,I
```

Where:

sssssss

The name of the remote LU.

To deactivate all LU 6.2 sessions immediately between remote LUs and a specific TPF/APPC local LU, enter the following:

```
ZNETW INACT ID=pppp,I
```

Where:

pppp

The name of the TPF APPL.



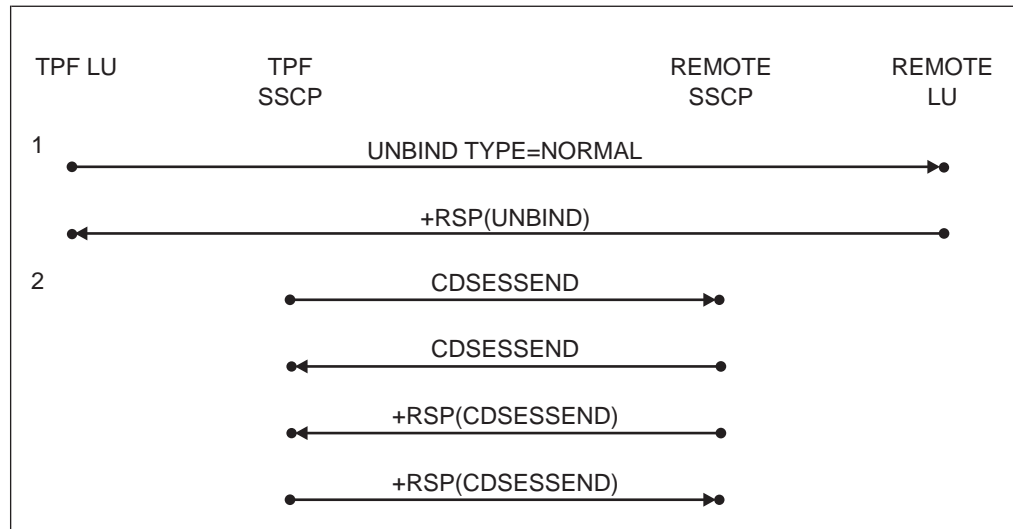


Figure 153. Immediate LU 6.2 Session Deactivation

**Note:** For PU2.1 sessions, no CDSESENDS are sent (skip step #2).

**Forced LU 6.2 Session Deactivation:** To force the deactivation of all LU 6.2 sessions with a particular remote LU, enter the following from the TPF system:

```
ZNETW INACT ID=sssssss,F
```

Where:

*sssssss*

The name of the remote LU.

To force the deactivation of all LU 6.2 sessions between remote LUs and a specific TPF/APPC local LU, enter the following:

```
ZNETW INACT ID=pppp,F
```

Where:

*pppp*

The name of the TPF APPL.

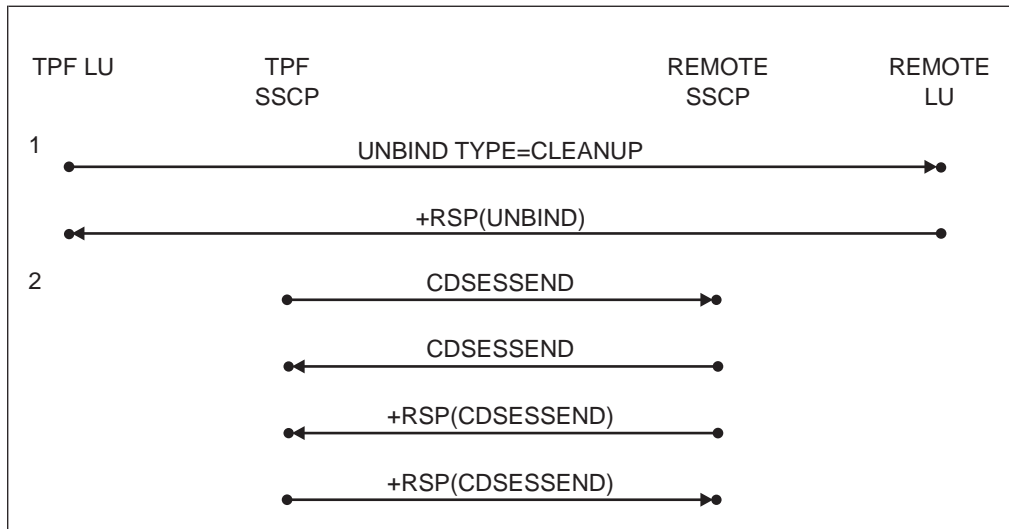


Figure 154. Forced LU 6.2 Session Deactivation

**Note:** For PU2.1 sessions, no CDSESENDS are sent (skip step #2).

## APPN Session Activation

The following sections show examples of session activations when the TPF system is connected to the network as an APPN end node (EN).

### CP-CP Session Activation

This section contains the flows for activating CP-CP sessions between the TPF system and its network node server (NNS).

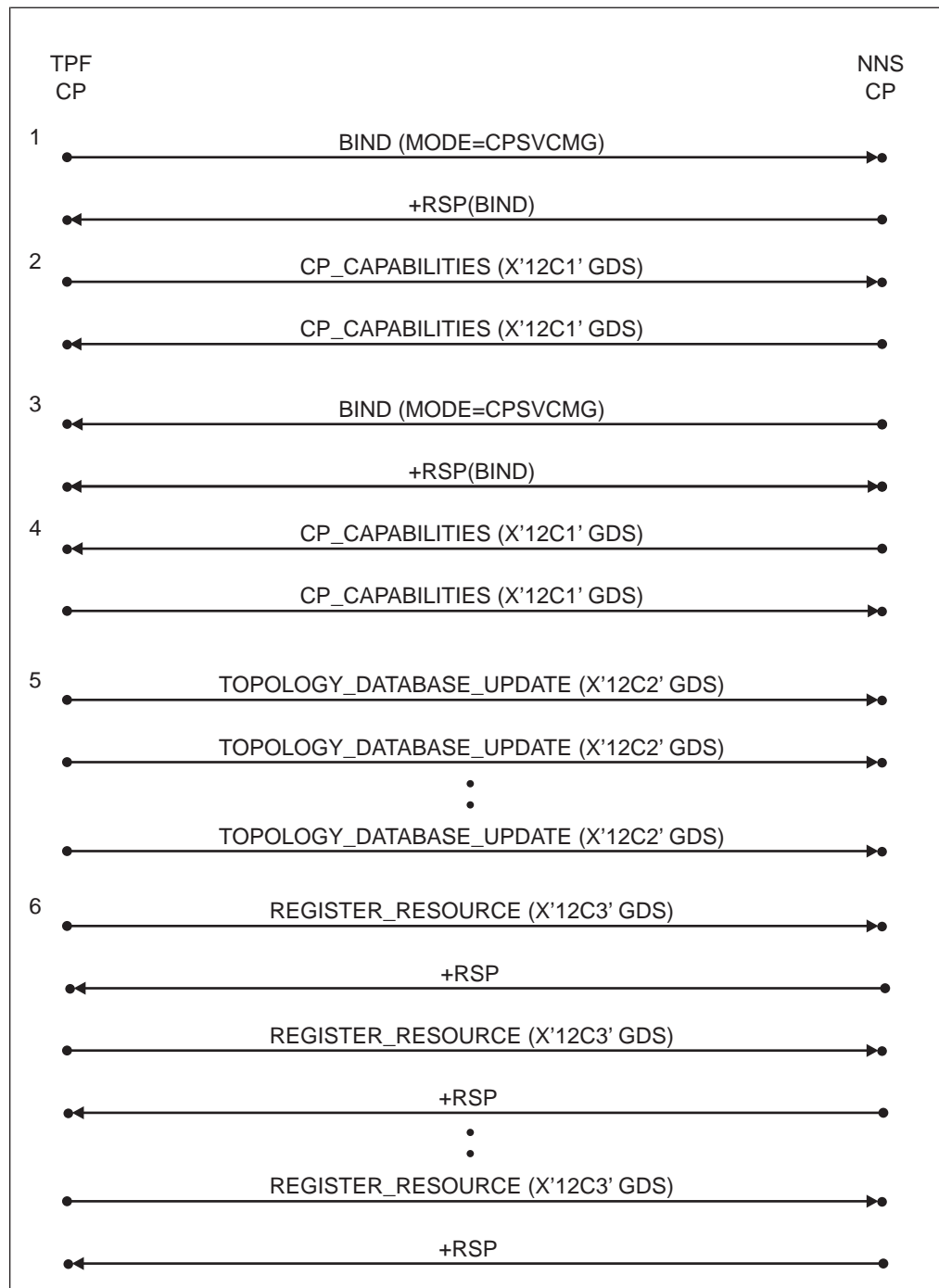


Figure 155. CP-CP Session Activation

**Note:** The CP-CP sessions are not considered active until step 4 completes.

### APPN LU-LU Session Activation

This section contains the flows for activating LU-LU sessions between the TPF system and remote LUs in an APPN network.

Legend for acronyms in the flow diagrams:

**APPL**            Application LU  
**CP**                Control point LU

<b>M1</b>	PCID modifier in the original LOCATE request
<b>M2</b>	PCID modifier used in the subprocedure
<b>NNS</b>	Network node server
<b>PLU</b>	Primary logical unit
<b>RSCV</b>	Route selection control vector (control vector X'2B')
<b>SC</b>	Session characteristics (control vectors X'31' and X'65')
<b>Sessst</b>	Session started notification
<b>SLU</b>	Secondary logical unit
<b>TGs</b>	Transmission groups (pairs of control vectors X'46' and X'47').

**Session Started by a TPF PLU:** The following diagram shows the flows for sessions that are started by a TPF PLU.

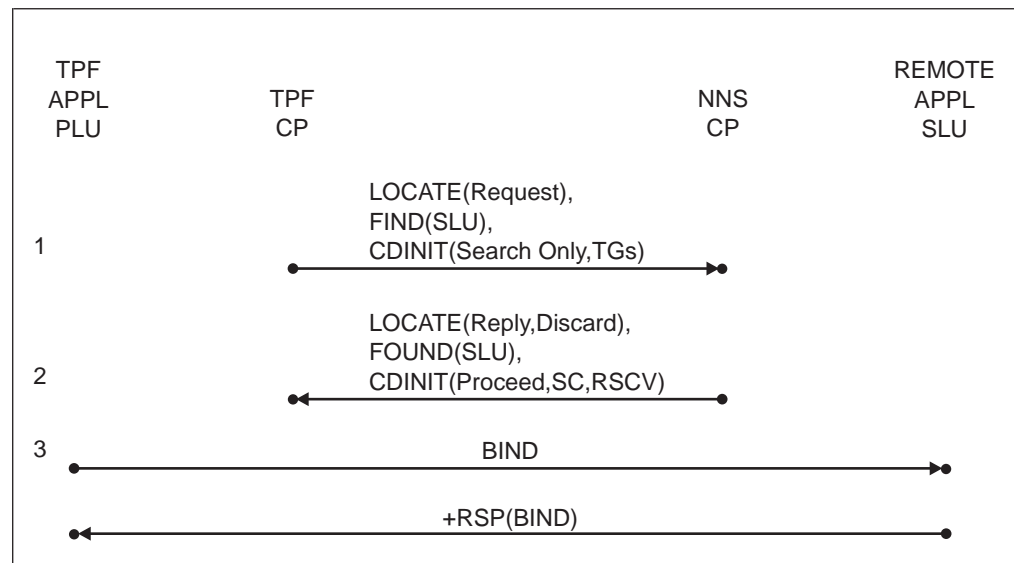


Figure 156. Session Started by TPF PLU (APPN)

**Session Started by a TPF SLU:** The following diagram shows the flows for sessions that are started by a TPF SLU.

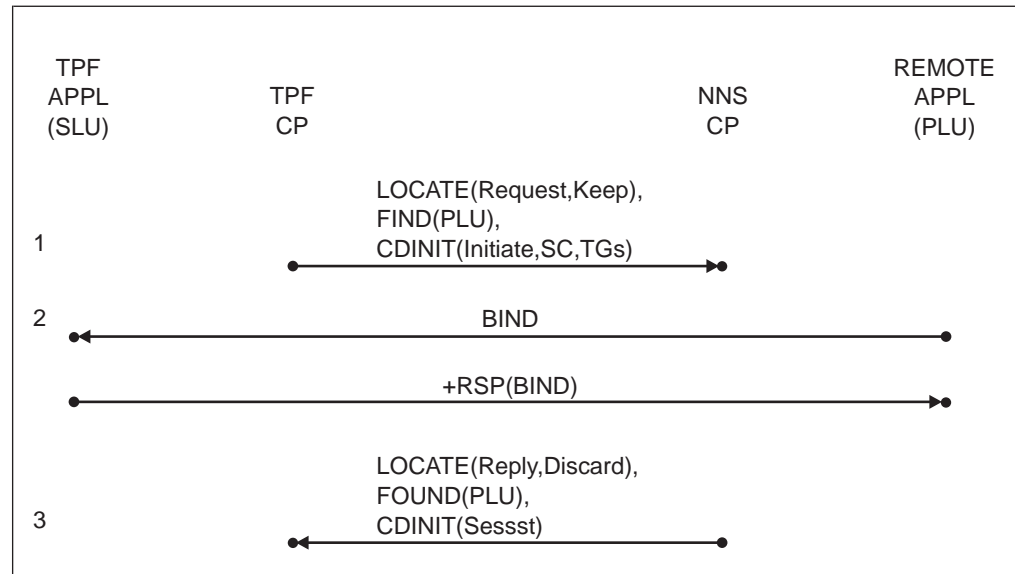


Figure 157. Session Started by TPF SLU (APPN)

**Session Started by a Remote SLU When RSCV is Provided:** The following diagram shows the flows for sessions that are started by a remote SLU when the RSCV is provided.

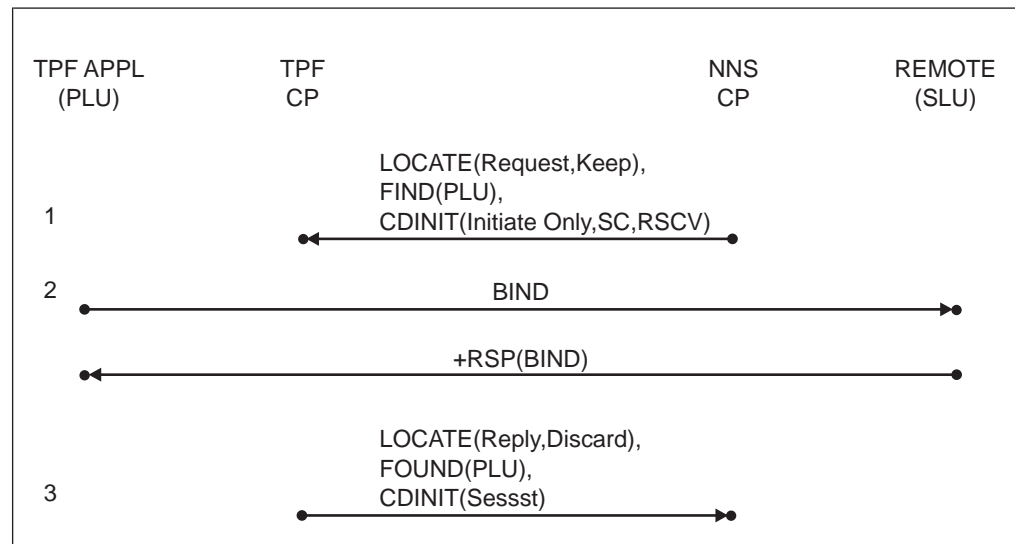


Figure 158. Session Started by Remote SLU (APPN), RSCV Provided

**Session Started by a Remote SLU When RSCV is Not Provided:** The following diagram shows the flows for sessions that are started by a remote SLU when the RSCV is not provided.



378 TPF V4R1 ACF/SNA Data Communications Reference

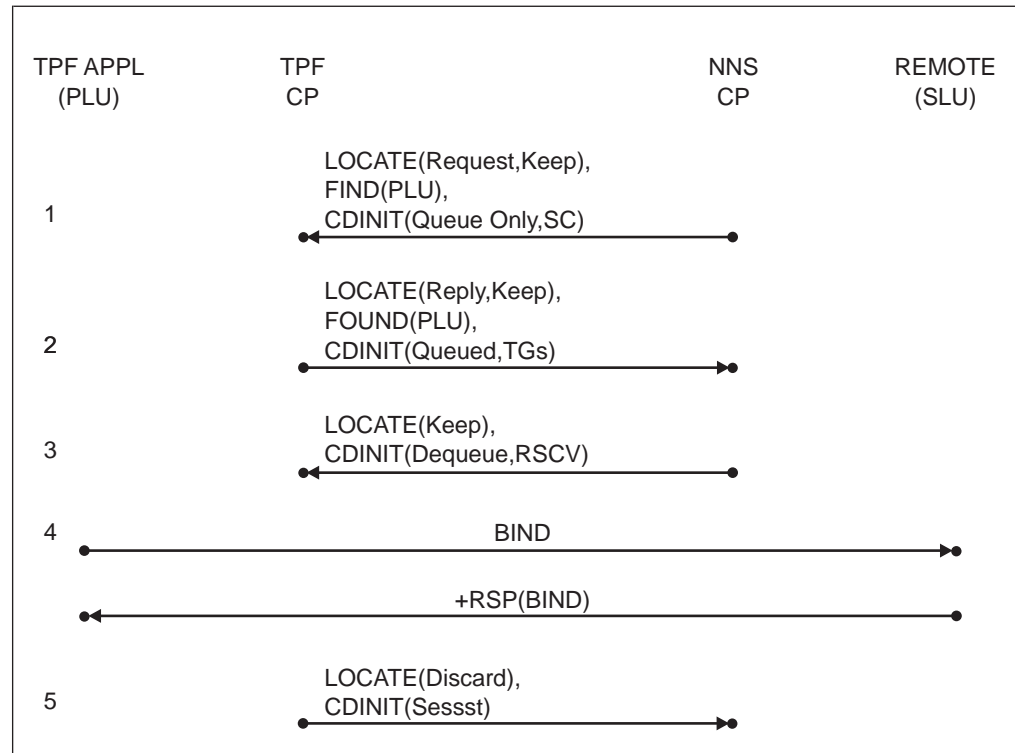


Figure 160. Session Started by Remote SLU (APPN), Session Queued

**Session Started by a Remote PLU:** The following diagram shows the flows for sessions that are started by a remote PLU.

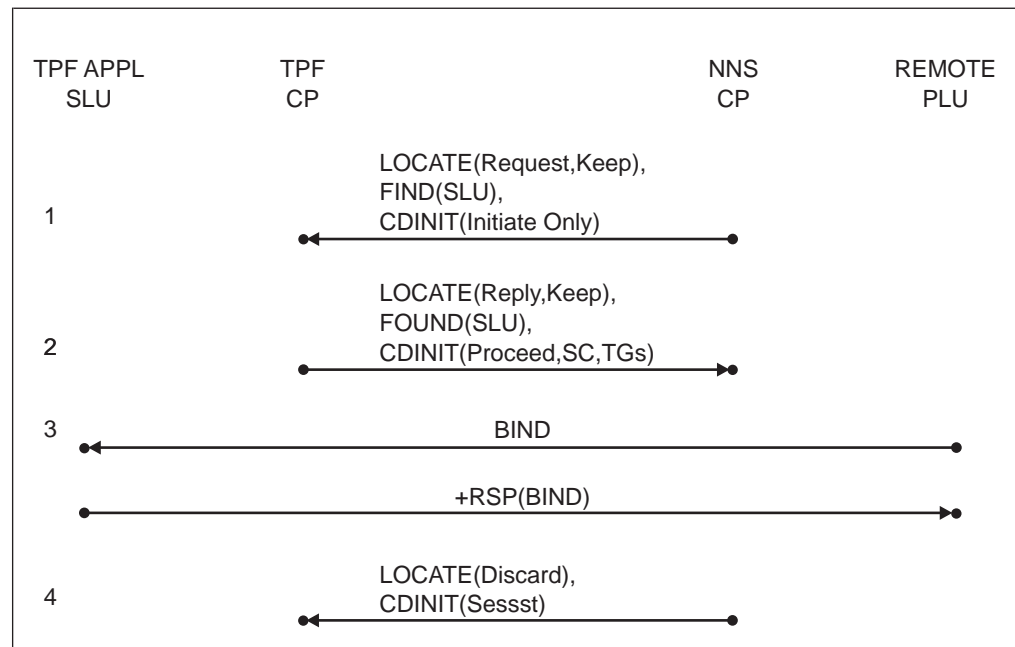


Figure 161. Session Started by Remote PLU (APPN)

**Session Started by a Remote PLU When the Session is Queued:** The following diagram shows the flows for sessions that are started by a remote PLU when the session is queued.

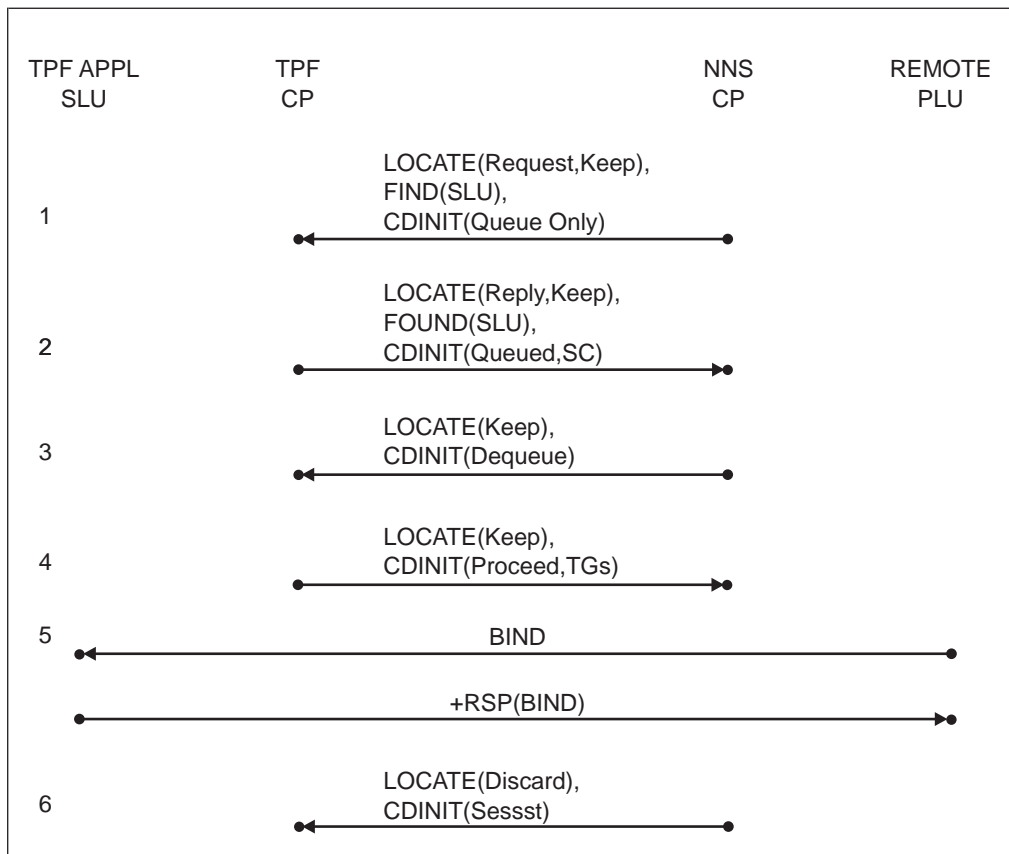


Figure 162. Session Started by Remote PLU (APPN), Session Queued

## LU-LU Session Activation for Printer Sharing

This section contains the flows for establishing an LU-LU session with a shared printer. The TPF system can either request the session with the printer or be asked to release its session with the printer.

Legend for acronyms in the flow diagrams:

<b>CP</b>	Control point LU
<b>I/Q</b>	Initiate or queue
<b>NNS(SLU)</b>	Network node server for the SLU
<b>NNS(TPF)</b>	Network node server for the TPF system
<b>PLU</b>	Primary logical unit
<b>Relreq</b>	Release session request
<b>RSCV</b>	Route selection control vector (control vector X'2B')
<b>SC</b>	Session characteristics (control vectors X'31' and X'65')
<b>Sessst</b>	Session started notification
<b>SLU</b>	Secondary logical unit
<b>TGs</b>	Transmission groups (pairs of control vectors X'46' and X'47').



**Session Requested by the TPF System (Printer Available):** The following diagram shows the flows when the TPF system requests a session with a shared printer that is available.

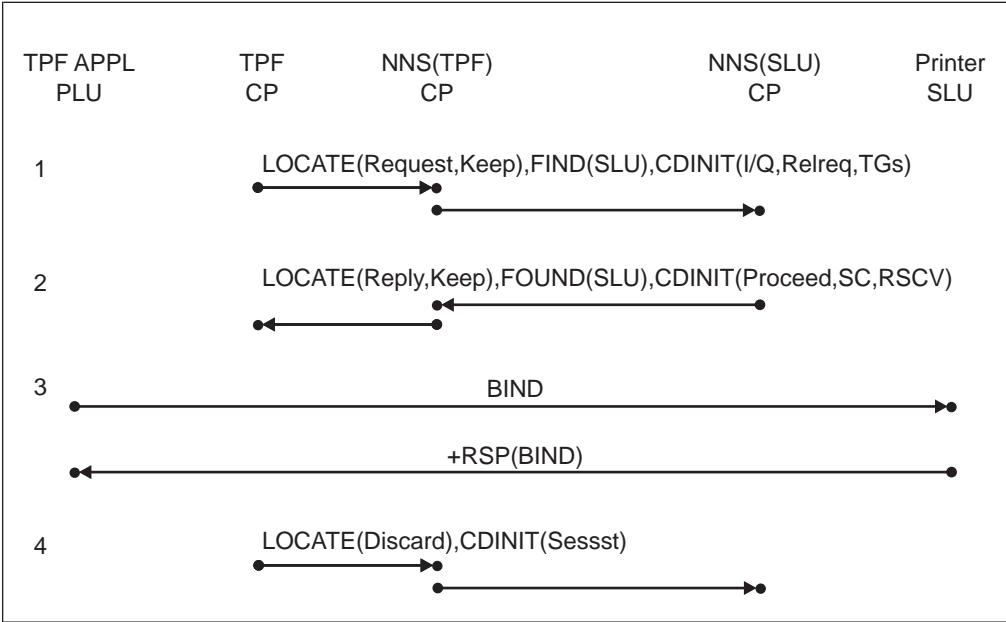


Figure 163. Printer Sharing, Request by TPF, Printer Available

**Session Requested by the TPF System (Printer In Use):** The following diagram shows the flows when the TPF system requests a session with a shared printer that is in use.

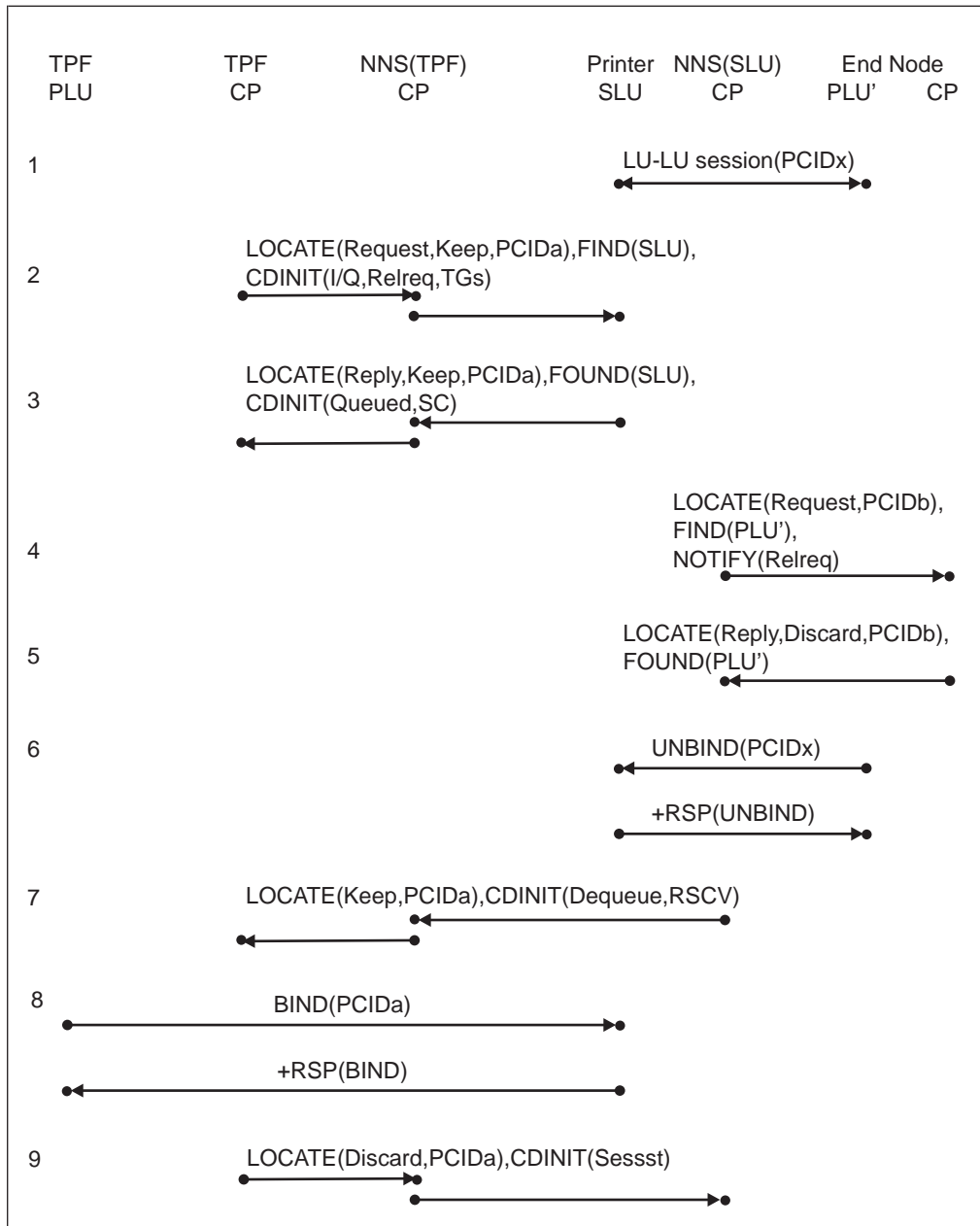


Figure 164. Printer Sharing, Request by TPF, Printer In Use

**TPF System Requested to Release the Shared Printer:** The following diagram shows the flows when the TPF system is requested to release the session with a shared printer.

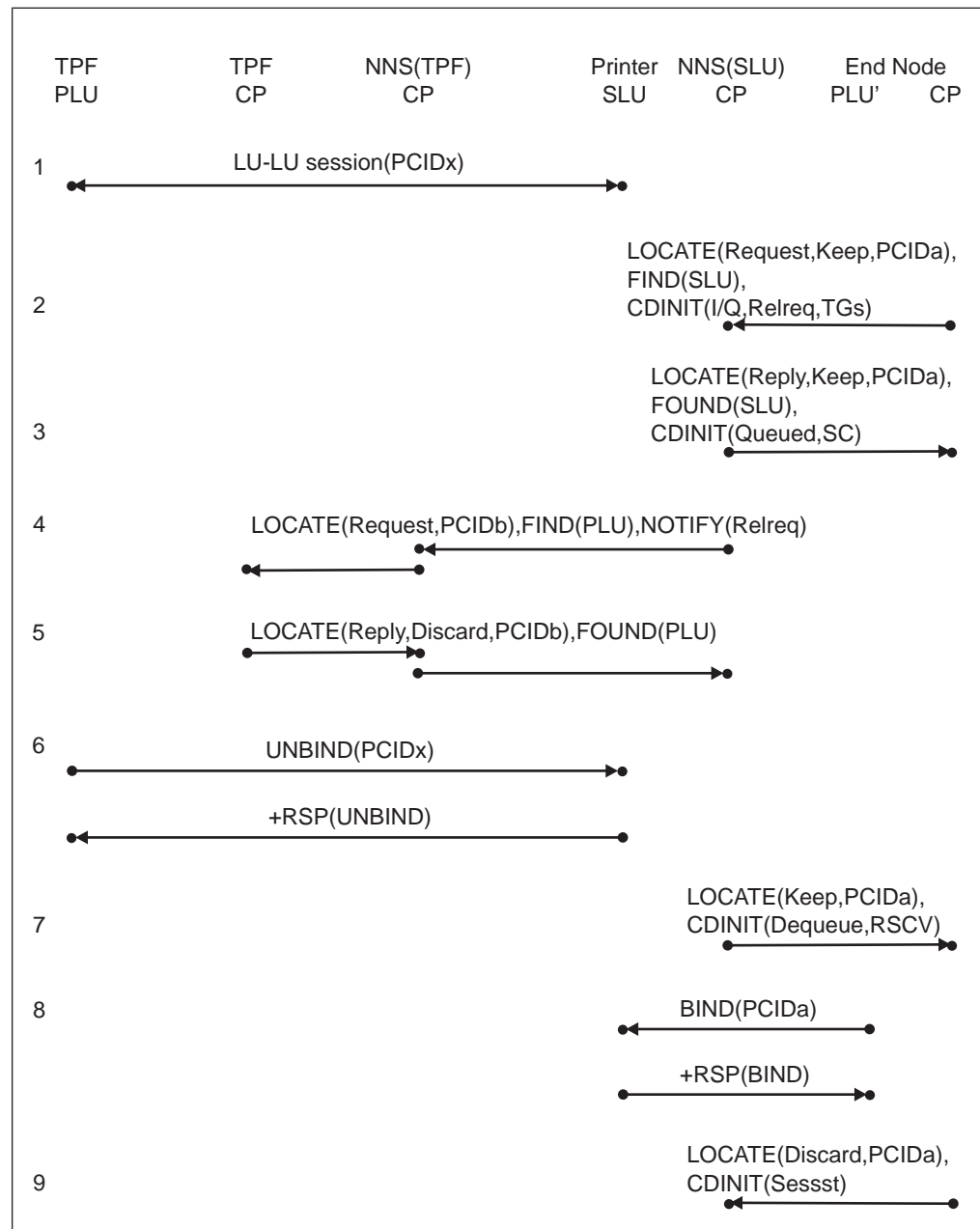


Figure 165. Printer Sharing, TPF Requested to Release the Printer

## APPN LU-LU Session Deactivation

This section contains the flows for deactivating LU-LU sessions in an APPN network. For active LU-LU sessions, the control points (CPs) are not involved except for the case where normal deactivation is done by a SLU. See “PU 5 and PU 2.1 Session Deactivation” on page 360 for the other deactivation flows.

Legend for acronyms in the flow diagrams:

**APPL** Application LU  
**CP** Control point LU  
**NNS** Network node server

**PLU** Primary logical unit

**SLU** Secondary logical unit.

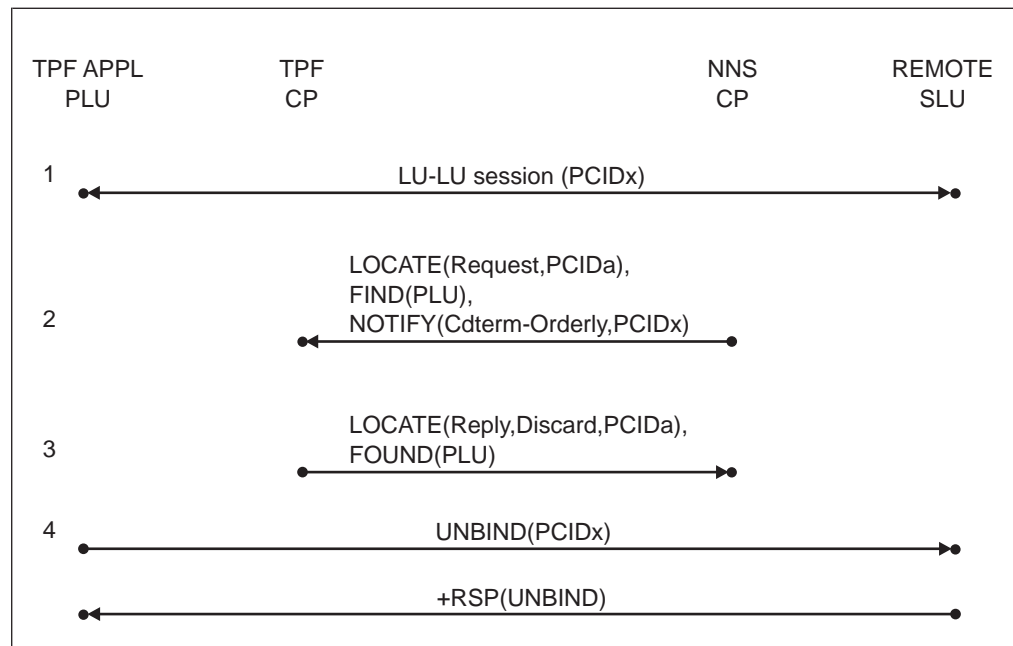


Figure 166. Normal Deactivation by Remote SLU (APPN)

## APPN-Subarea Migration Flows

This section contains the flows for establishing LU-LU sessions between the TPF system in an APPN network and remote LUs in a subarea (PU 5) network.

Legend for acronyms in the flow diagrams:

<b>BF</b>	Boundary function within the NCP
<b>CDRM</b>	Cross-domain resource manager
<b>CP</b>	Control point LU
<b>INN</b>	Interchange network node
<b>NNS</b>	Network node server
<b>PLU</b>	Primary logical unit
<b>RSCV</b>	Route selection control vector (control vector X'2B')
<b>SC</b>	Session characteristics (control vectors X'31' and X'65')
<b>Sessst</b>	Session started notification
<b>SLU</b>	Secondary logical unit
<b>TGs</b>	Transmission groups (pairs of control vectors X'46' and X'47').

### SLU Initiated, SLU in APPN, PLU in Subarea

The following diagram shows the flows for sessions that are initiated by the SLU. The SLU is in an APPN network and the PLU is in a subarea network.

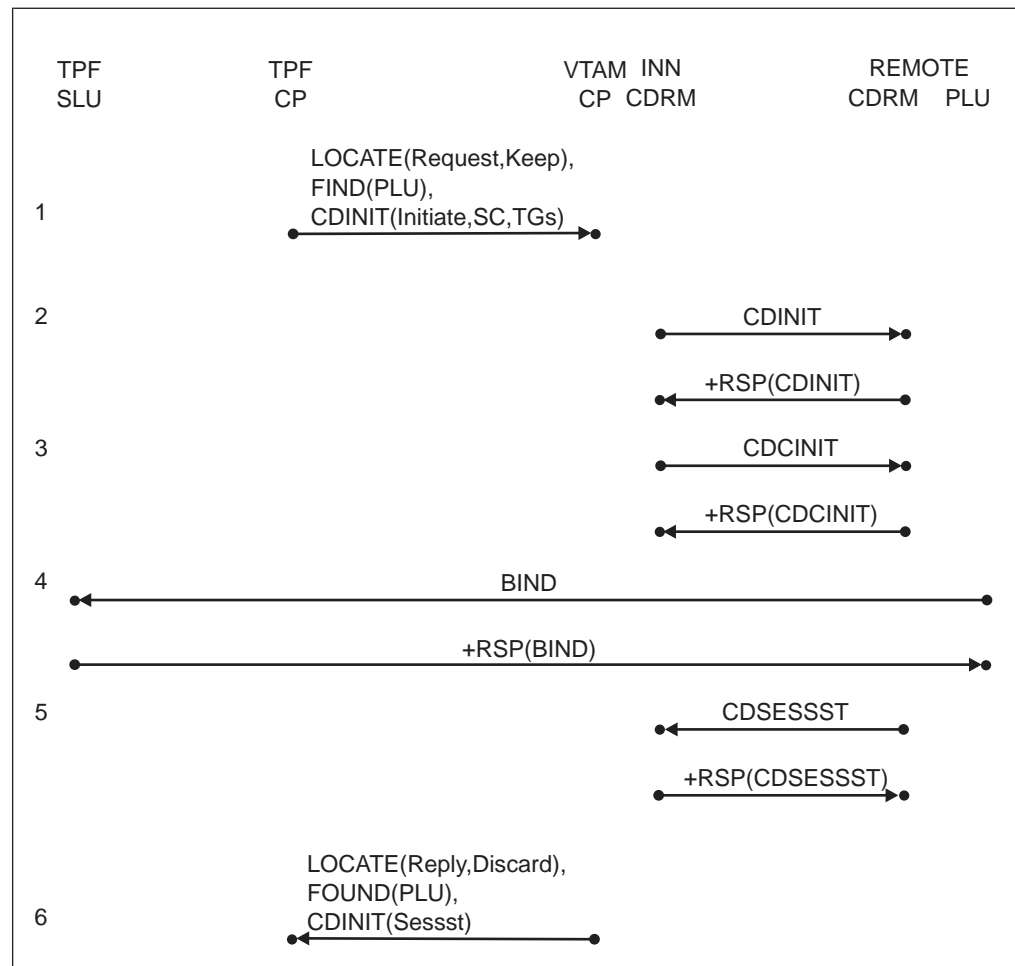


Figure 167. SLU Initiated, SLU in APPN, PLU in Subarea

### SLU Initiated, SLU in APPN, PLU in Subarea (PLU Location Unknown)

The following diagram shows the flows for sessions that are initiated by the SLU. The SLU is in an APPN network and the PLU is in a subarea network but the PLU location is unknown.

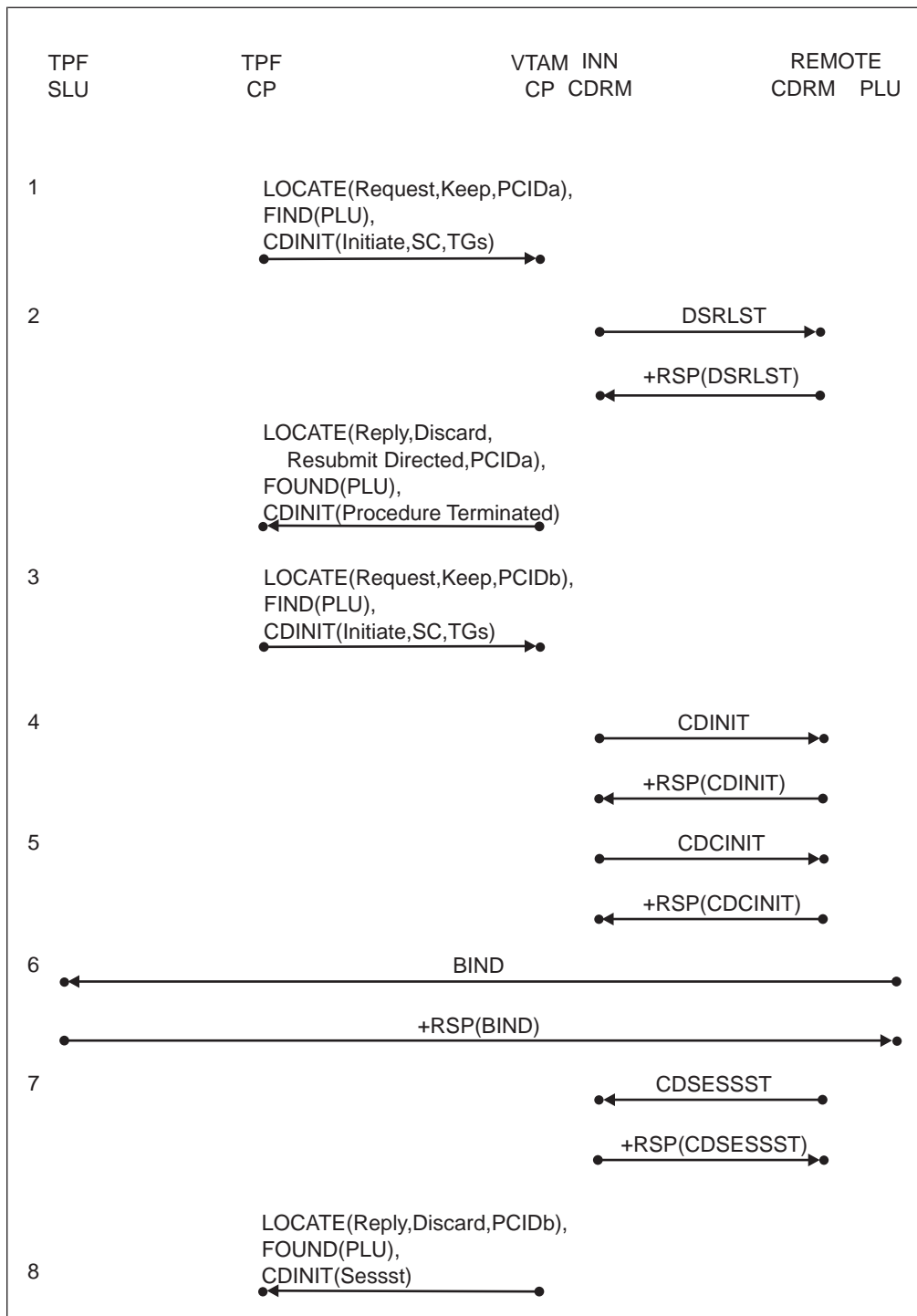


Figure 168. SLU Initiated, SLU in APPN, PLU in Subarea (PLU Location Unknown)

### PLU Initiated, SLU in APPN, PLU in Subarea

The following diagram shows the flows for sessions that are initiated by the PLU. The SLU is in an APPN network and the PLU is in a subarea network.

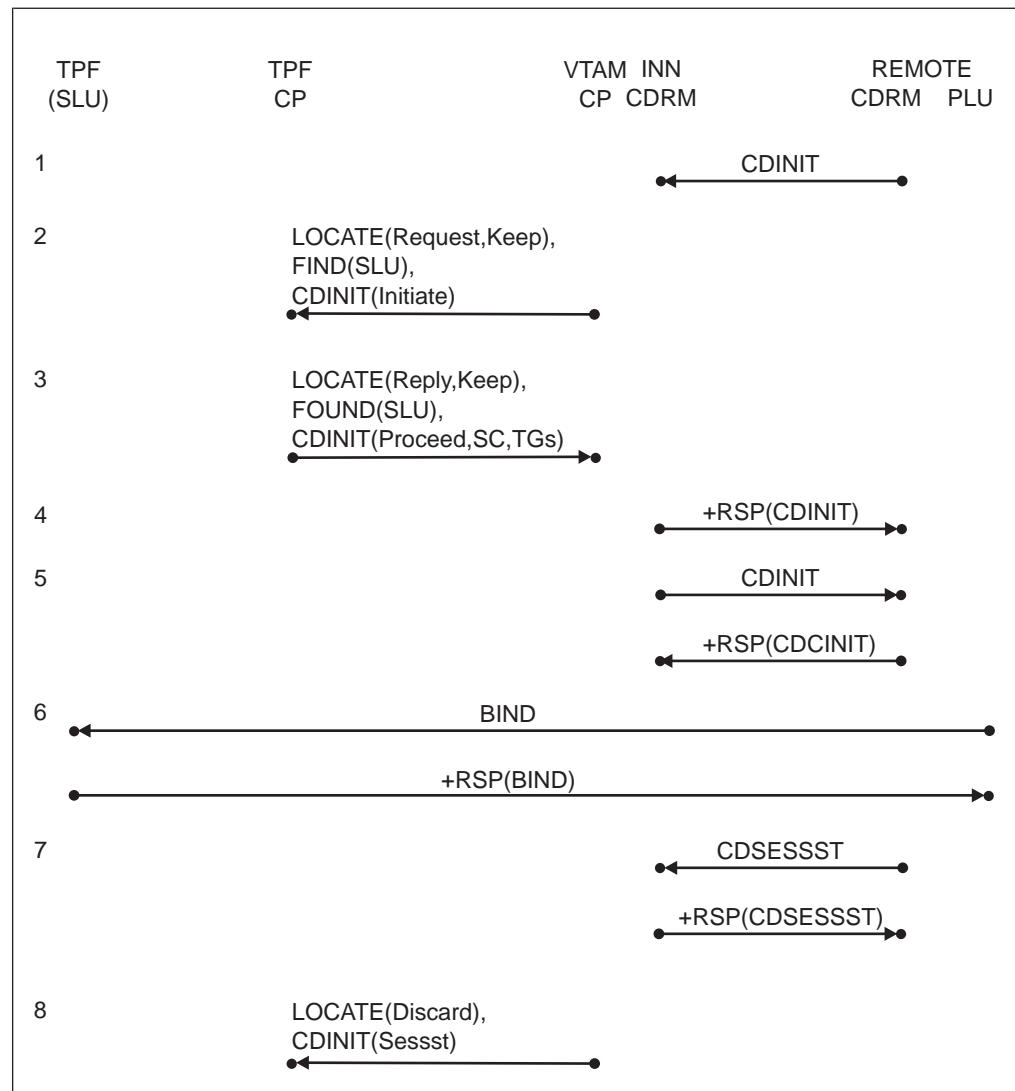


Figure 169. PLU Initiated, SLU in APPN, PLU in Subarea

### SLU Initiated, PLU in APPN, SLU in Subarea (SC Provided)

The following diagram shows the flows for sessions that are initiated by the SLU. The PLU is in an APPN network and the SLU is in a subarea network. Session characteristics are provided.

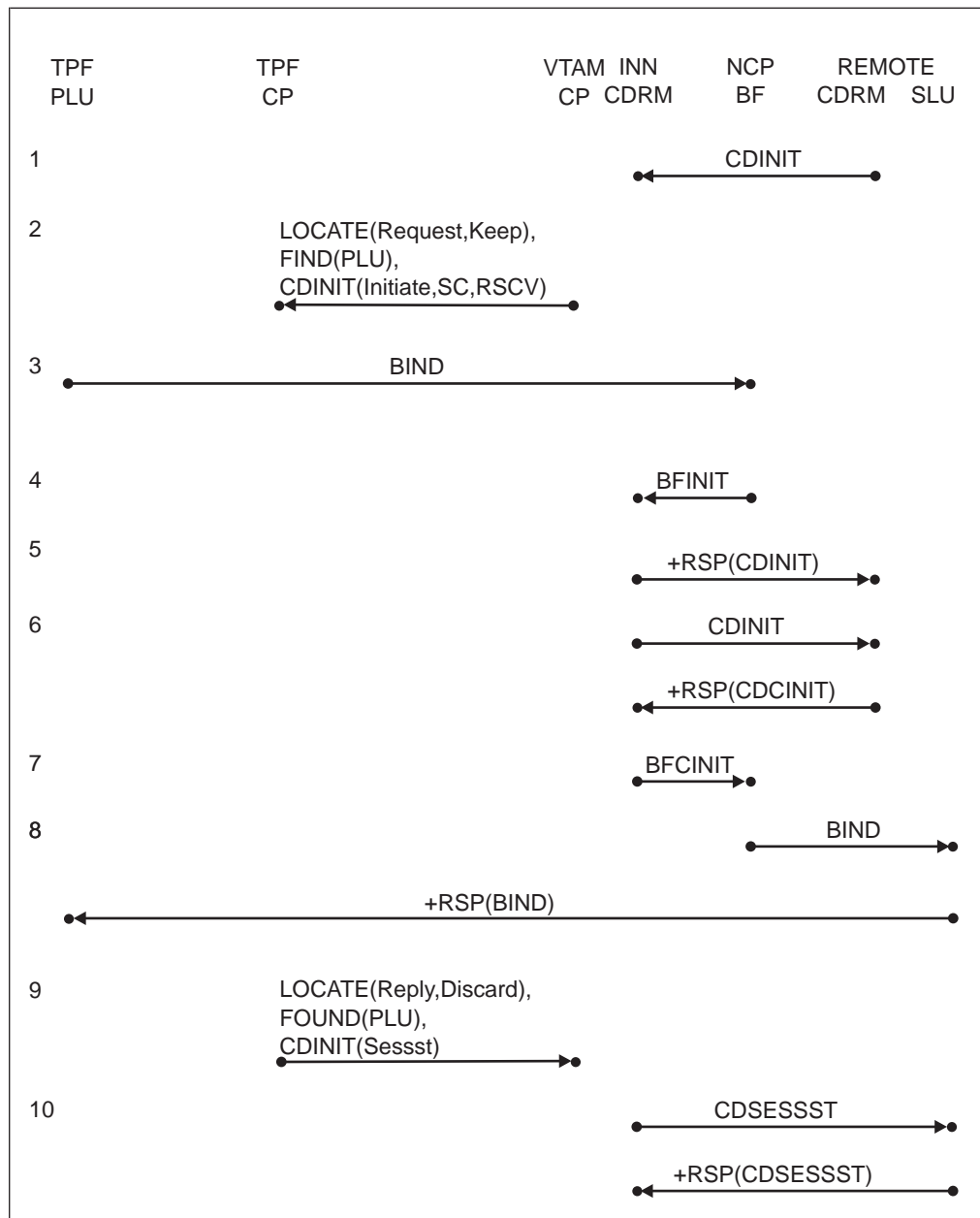


Figure 170. SLU Initiated, PLU in APPN, SLU in Subarea (SC Provided)

### SLU Initiated, PLU in APPN, SLU in Subarea (SC Not Provided)

The following diagram shows the flows for sessions that are initiated by the SLU. The PLU is in an APPN network and the SLU is in a subarea network. Session characteristics are not provided.



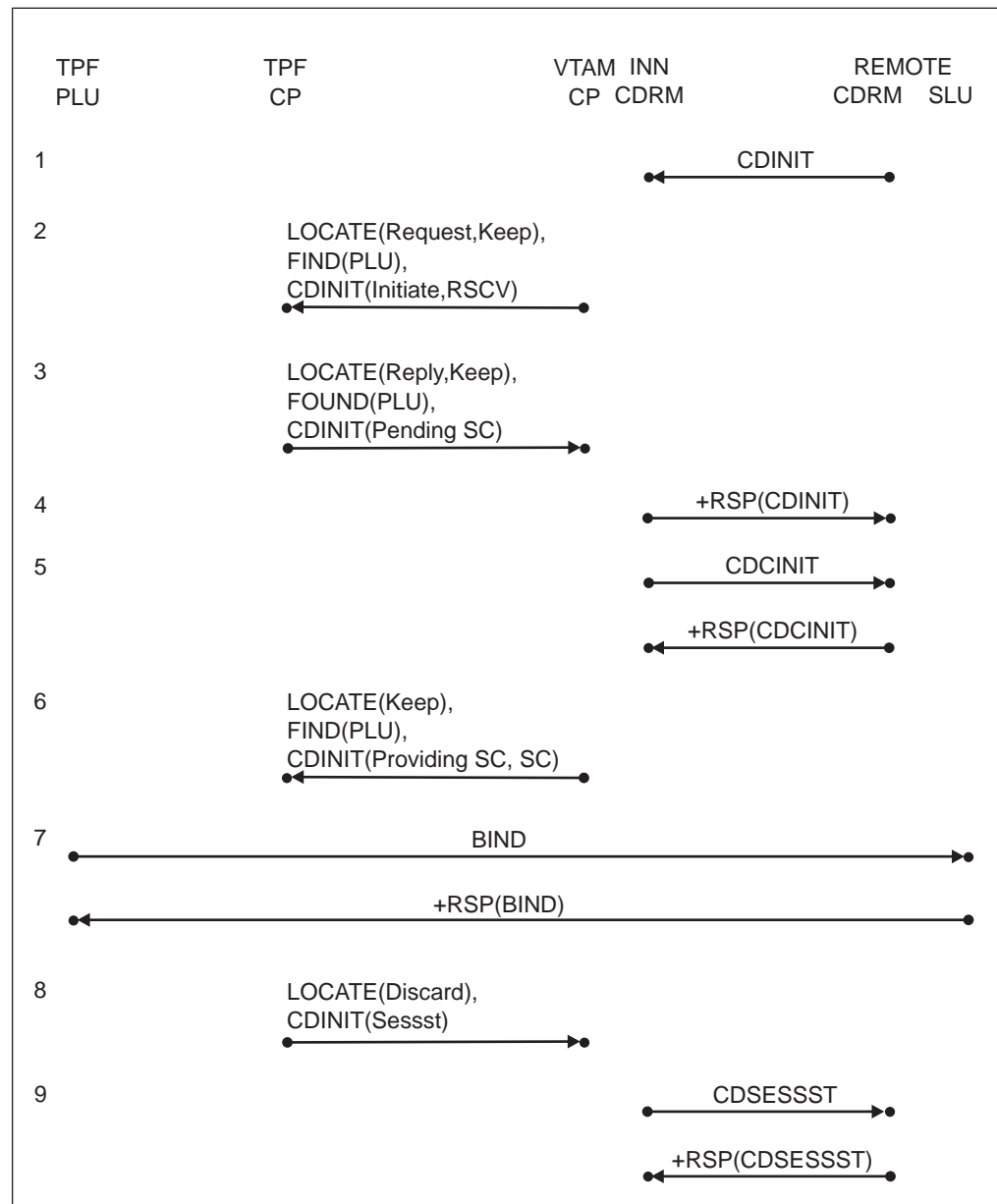


Figure 171. SLU Initiated, PLU in APPN, SLU in Subarea (SC Not Provided)

### PLU Initiated, PLU in APPN, SLU in Subarea (SC Provided)

The following diagram shows the flows for sessions that are initiated by the PLU. The PLU is in an APPN network and the SLU is in a subarea network. Session characteristics are provided.

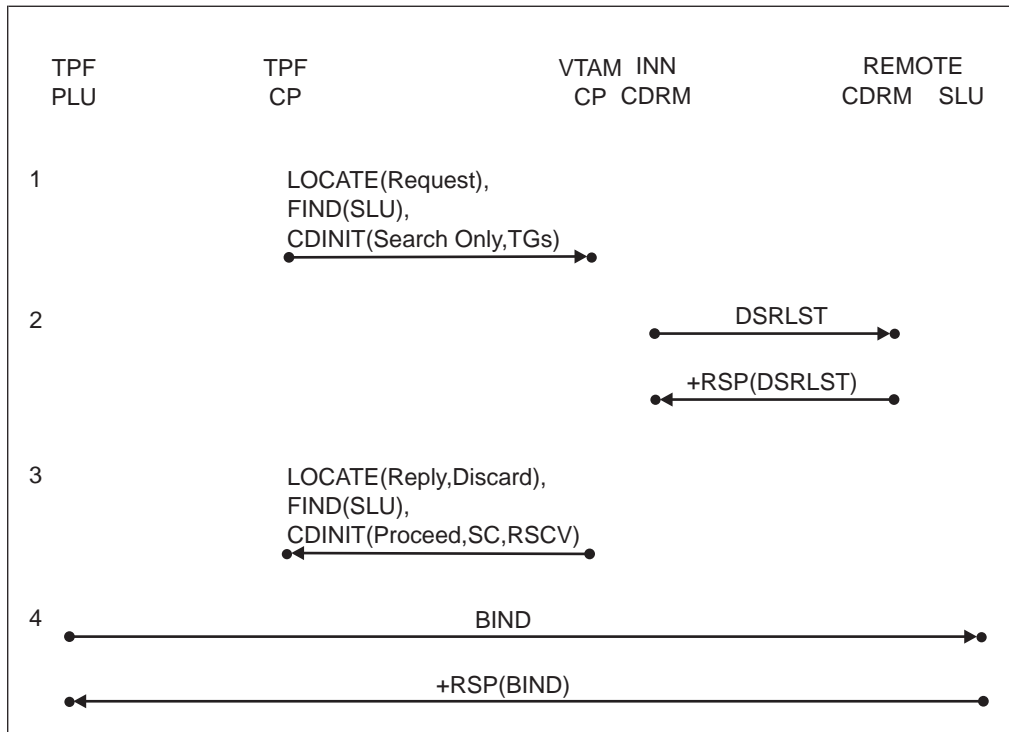


Figure 172. PLU Initiated, PLU in APPN, SLU in Subarea (SC Provided)

### PLU Initiated, PLU in APPN, SLU in Subarea (SC Not Provided)

The following diagram shows the flows for sessions that are initiated by the PLU. The PLU is in an APPN network and the SLU is in a subarea network. Session characteristics are not provided.

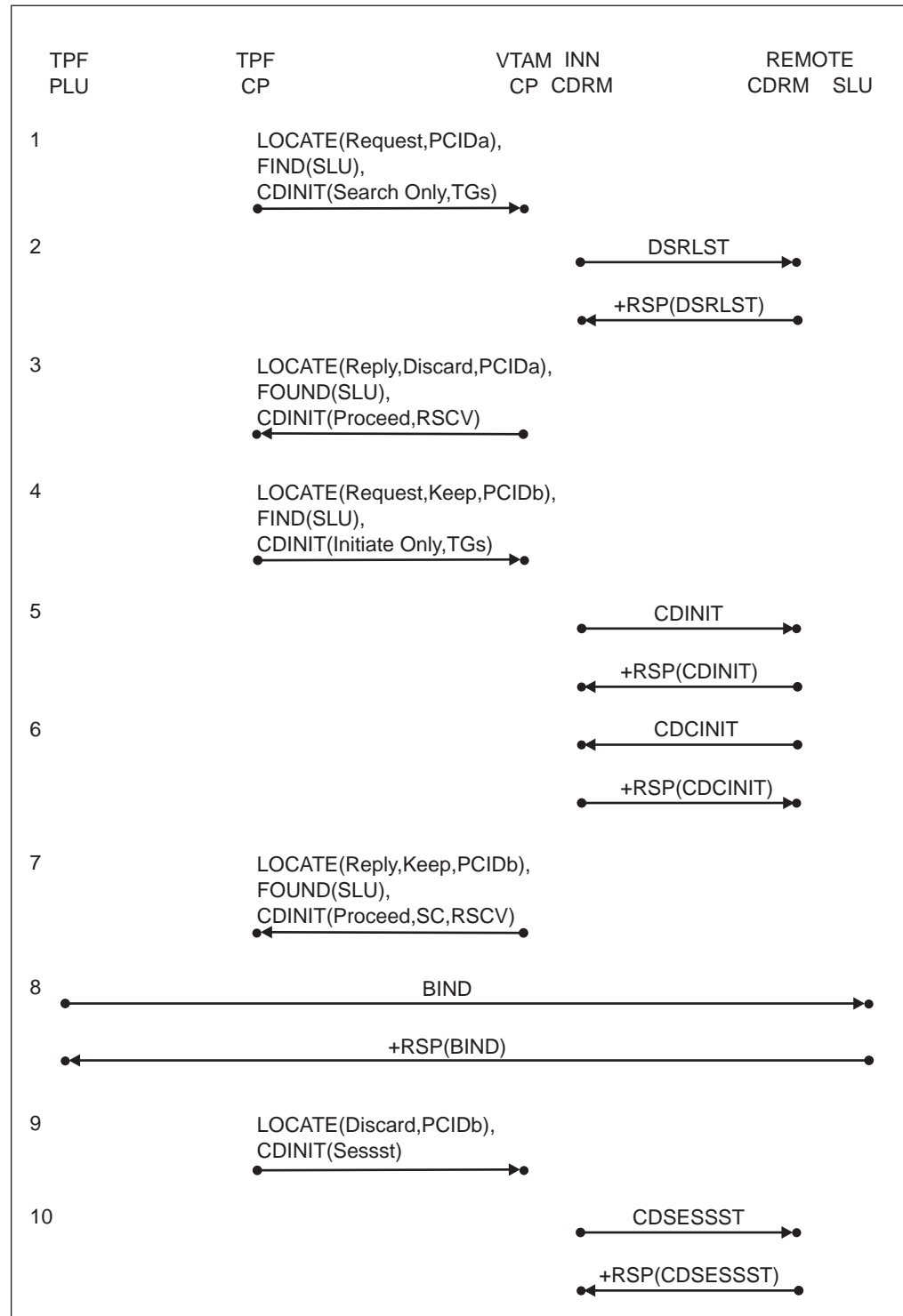


Figure 173. PLU Initiated, PLU in APPN, SLU in Subarea (SC Not Provided)

## More on SNA Command Flow

For additional information about SNA command flows, see the following publications:

- *SNA Formats*

- *IBM Systems Network Architecture Technical Overview.*

## Appendix I. TPF Sense Code Processing

Every SNA command sent by the TPF system has a corresponding response handler that performs sense error handling internally or passes control to the SSCP negative response handler.

The response handler handles a negative response if it is to be treated as a positive response. Generally a negative response to the last command to flow during session deactivation is treated as positive because, whether the response is positive or negative, everything should be cleaned up at this time. Examples of commands that can be the last commands to flow are CDSESEND, CDTERM, DACTCDRM, DACTPU and UNBIND.

The actions that can be taken by the negative response handler are:

- **CLNUP**

If the session is bound or is starting up, the necessary commands are issued to take it down. Control blocks are then cleaned up.

A 511 SNAPC dump is taken if the sense handling table (CSD9) indicates a dump should be taken. The message accompanying the SNAPC gives the sense code, the command, and whether it was a session failure or session activation failure.

An operator message is displayed if the command that failed was part of session initiation due to a ZNETW ACT command. Otherwise no message is displayed. The message that is displayed gives the sense code or 'TIMED OUT', and the command that caused the session activation failure.

- **EXIT**

No action is taken. This is generally used for sense indicating contention.

- **RESND**

If the command has already been re-sent once, then the action is CLNUP. Otherwise, the command is resent.

- **WAIT**

This action indicates that the TPF system should wait for a response to the original transmission because the resend of the command resulted in a negative response indicating that the first transmission is being processed.

- **RESYNC**

If there is no session, the action is CLNUP. Otherwise session resynchronization is scheduled.

Table 26 shows the action taken based on the sense code. The default action for any sense code not listed in this table is CLNUP, a message, and the 511 SNAPC dump.

*Table 26. Sense Codes and Actions*

Sense	Command	Action	Dump
0801	ANY	CLNUP	NO
0805	ACTCDRM	EXIT	NO
	ANY	CLNUP	NO
0806	ANY	CLNUP	NO
0808	ACTCDRM	CLNUP	YES

Table 26. Sense Codes and Actions (continued)

Sense	Command	Action	Dump
0809	ANY	CLNUP	YES
080D	CDINIT	EXIT	NO
	ACTCDRM	EXIT	NO
080E	ANY	CLNUP	NO
0812	ANY	RESND	NO
0815	ANY	CLNUP	YES
0817	ANY	CLNUP	NO
081E	ANY	CLNUP	YES
0821	ANY	CLNUP	YES
0822	ANY	CLNUP	NO
0831	BIND	CLNUP	NO
	CHASE	EXIT	NO
0832	ANY	CLNUP	YES
0833	ANY	CLNUP	YES
0835	ANY	CLNUP	YES
0836	CDINIT	CLNUP	YES
0839	ANY	CLNUP	NO
083B	ANY	CLNUP	YES
0841	CDINIT	CLNUP	YES
084B	CDINIT	CLNUP	NO
0852	BIND	CLNUP	YES
0857	BIND	CLNUP	NO
	CDINIT	CLNUP	NO
0858	ACTCDRM	RESND	NO
0877	BIND	CLNUP	YES
087D	CDINIT	CLNUP	NO
0881	ACTCDRM	CLNUP	NO
088C	ANY	CLNUP	YES
0893	BIND	RESND	NO
0894	BIND	CLNUP	YES
0895	ANY	CLNUP	YES
0896	ANY	CLNUP	YES
0897	ANY	CLNUP	YES
1002	ANY	CLNUP	YES
1003	ANY	CLNUP	YES
1007	ANY	CLNUP	YES
2005	ANY	CLNUP	YES
2007	ANY	CLNUP	YES
8002	ANY	CLNUP	NO
8003	BIND	CLNUP	NO

Table 26. Sense Codes and Actions (continued)

Sense	Command	Action	Dump
8004	BIND	CLNUP	YES
8005	ANY	CLNUP	NO
8006	ANY	CLNUP	YES
8007	ANY	CLNUP	YES
8008	BIND	CLNUP	NO
8009	ANY	CLNUP	NO
800C	ANY	CLNUP	YES
800D	ANY	RESND	NO
8011	ANY	CLNUP	NO
8013	ANY	CLNUP	NO

---

## SNA Timeout Processing

If the TPF system is waiting for a response to a command and no response is received before the SSCP time-out interval expires, a pseudo negative response is created and passed to the command router. The command router passes the response to the appropriate response handler for action. The action taken by the response handler for timeout processing is CLNUP. See Appendix I, "TPF Sense Code Processing" on page 393 for information on negative response processing and the CLNUP action.





---

# Index

## Special Characters

#CCBRU 61  
#CMSIT 61  
#SC1RU 61  
#SC2RU 61

## Numerics

3174 APPN 2, 4, 16, 244  
3270 Application Considerations 26  
    copy (SNA command) 26  
3270 welcome screen 216  
3600 Application Considerations  
    Data Transmission 25  
    exception response request 25  
3601 applications 25  
3601 control program 23  
3614/3624 Message Processing 251  
3614/3624 Session Initiation and Application  
    Considerations 251  
37x5 communications controller 11  
37x5 Considerations 240, 241  
4K read buffers 225  
4K write buffers 225

## A

Acceptable BIND Image For a Local Host Node  
    SLU 284  
action codes 23  
activating a cross-domain link 14  
activating and deactivating a Shared NCP  
    loading NCP on 37x5 201  
activating and deactivating APPN CP-CP sessions 204  
activating and deactivating control LU-Logon Manager  
    sessions 204  
activating and deactivating cross-domain resource  
    managers  
    PU 5 environment 202  
activating and deactivating resources 201  
activating LU-LU sessions 206  
activating sessions 20  
adaptive rate-based pacing  
    ARB algorithm 168  
    description of 168  
address conversion 19  
addressing ALC devices 48  
Adjacent Link Station - ALS 4  
advanced peer-to-peer networking 16  
Advanced Peer-to-Peer Networking 2, 16  
    37x5 Considerations 240  
    considerations with System network Interconnection  
        (SNI) 234  
    CP-CP sessions 204  
    environment 16  
    LU registration 211  
    LU-LU sessions 211

agent assembly area (AAA) 19  
Airlines Line Control (ALC) 2  
    Airlines Line Control Interface (ALCI) 45  
    AX.25 support 45  
    Network Extension Facility (NEF) 45  
    support through SNA 45  
    terminal interchange 46  
Airlines Line Control Interface (ALCI) 2  
ALC (Airlines Line Control) 2  
ALCI (Airlines Line Control Interface) 2, 45  
alive timer  
    defining  
        SNAKEY macro 145  
        ZNKEY command 145  
    description of 145  
    heartbeat message 145  
ALLOCATE 73  
ALS 64, 65  
ALS - Adjacent Link Station 4  
alternate CRAS 49  
AM0SG 100  
AMSG format  
    coding applications 42  
ANR labels 128, 136  
ANR nodes 127  
application development  
    considerations for  
        NCB records 190  
        CSNB segment 190  
application message 25  
application programs  
    defining 201  
    starting/stopping 201  
applications, concurrent 12  
APPN mode 4  
ARB algorithm 168  
architecture comparisons for TPF/APPC  
    ALLOCATE 73  
    CONFIRM 73  
    CONFIRMED 73  
    DEALLOCATE 74  
    FLUSH 74  
    GET\_ATTRIBUTES 74  
    GET\_TYPE 75  
    POST\_ON\_RECEIPT 75  
    PREPARE\_TO\_RECEIVE 75  
    RECEIVE\_AND\_WAIT 76  
    REQUEST\_TO\_SEND 76  
    SEND\_DATA 76  
    SEND\_ERROR 77  
    TEST 77  
    WAIT 77  
assembler, H-level 69  
ATTACH interface 69  
automatic network routing (ANR) labels 128, 136  
automatic network routing (ANR) nodes 127  
AX.25 NCP definition  
    pacing 240

## B

BIND 99  
 bind (SNA command) 22  
 bind command processing  
   relationship to input buffer size 32  
 BIND Images for TPF Supported Secondary Logical  
   Units 283  
 BNN (Boundary Network Node) 29  
 Boundary Network Node (BNN) 29  
 Bracket Support 250

## C

C-language 69  
 CALL\_CONFIRM command  
   received from NPSI/GATE 334  
 CALL\_OUT command  
   TPF initiated to NPSI/GATE 332  
 CALL\_REQUEST command  
   received from NPSI/GATE 330  
 CCB (conversation control block) 67  
 CCB ID (conversation control block identifier) 67  
 CCP (communications control program) 48  
 CCSNAE 100  
 CDRSC 63, 64  
 Chained and Segmented Messages 248  
 chaining (NPSI)  
   request unit (RU) 31  
 change direction indicator 43  
 change number of sessions (CNOS) work block 68  
 channel contact in CTC 223  
 channel-to-channel (CTC)  
   activation of virtual route 229  
   channel contact 223  
   considerations with FID4 222, 223  
   considerations with System network Interconnection  
   (SNI) 234  
   data transfer 11  
   deactivation of virtual route 230  
   pre-channel contact/priming 222  
   slowdown 277  
 Channel-to-Channel (CTC) priming 222  
 CHDD 61, 100  
 check characters  
   cyclic 46  
   synchronization 46  
 CIAA 46  
 CICS 26, 66  
   3790 full function LU support 29  
 CICS compatibility 19  
 CICS Relay Application Considerations 29  
 CIFRC macro 251  
 CK2SN 61  
 class of service (COS) 227  
 Class of Service (COS) 239  
 clear (SNA command) 247  
 CLEAR command  
   received from NPSI/GATE 331

CLEAR command (*continued*)  
   TPF initiated sent to NPSI/GATE 335  
 CLEAR-CONFIRM command  
   received from NPSI/GATE 335  
 CLU 16  
 cluster controller (3274/3276) 4, 43, 250  
 cluster controller (3x74/3276) 26  
 CLXC 48  
 CMC (communications management configuration) 46  
 CNE1 47  
 command content for FTPI 34  
 command flow 345  
 command support in TPF/NEF/AX.25 48  
 Communication and Transmission Control Program  
   (CTCP) 29, 317  
   activating for GATE/FTPI 36  
   control block residence 326  
   definition for GATE/FTPI 36  
   functions 321  
   input data message processing 335  
   output data message processing 335  
   sample implementation using PSVs 321  
   user control blocks 322  
 communication path 19  
 communication rate 19  
 communications control program (CCP) 48  
   macro routine (CLXC) 48  
 communications controller 11  
 communications management configuration (CMC) 46  
 communications program 3  
 communications source program (CIAA) 46  
 computer room agent table (CRAT) 49  
 concurrent message processing 250  
 configuration image 251  
 configuration report 49  
 CONFIRM 73  
 CONFIRMED 73  
 Contact Point Control Block (CPCB) 326  
 control blocks for TPF/APPC 67  
 control information (CI) field in FMH 22  
 control messages, HPR 151  
 Control Point 16  
   activating 204  
   deactivating 204  
 control records 19  
 control unit (3271) 26  
 controlling logical and physical units 3  
 controlling NEF/AX.25/XALCI 47  
 controlling the network 11  
 conversation control block (CCB) 67  
 conversation control block identifier (CCB ID) 67, 71  
 conversation verbs for TPF/APPC  
   basic conversation verbs 71, 72  
   mapped conversation verbs 71  
   type-independent verbs 71, 72  
 COS - Class of Service 239  
 COS (class of service) 227  
 COVX (NAU conversion program) 287  
 CP-CP sessions  
   activating 204  
   deactivating 204

- CPCB (Contact Point Control Block) 326
- CRAS 49
- cross-domain communication 12
- cross-domain links 12
  - activating and enabling 14
- cross-domain resource (CDRSC) 46
- cross-domain resource manager (CDRM) 14
  - establishing path between 12
- cross-domain sessions
  - requesting 14
- cross-domain support 45
- cross-domain takedown (CDTAKED) 202
- CS2A 100
- CSMP 49
- CSNB segment
  - accessing NCB records 190
- CSXA 107
- CSXB 107
- CSXC 107
- CTC (channel-to-channel)
  - activation of virtual route 229
  - channel contact 223
  - considerations with FID4 222, 223
  - considerations with System network Interconnection (SNI) 234
  - data transfer 11
  - deactivation of virtual route 230
  - pre-channel contact/priming 222
  - slowdown 277
- CTC (Channel-to-Channel)
  - data transfer 11
- CTCP (Communication and Transmission Control Program) 29, 317
  - activating for GATE/FTPI 36
  - control block residence 326
  - definition for GATE/FTPI 36
  - functions 321
  - input data message processing 335
  - output data message processing 335
  - sample implementation using PSVs 321
  - user control blocks 322
- CTK2 67
- CTKE 48
- cyclic check characters 46

## D

- data
  - encryption 251
  - exchange 3
  - transmission 20
- data and control message 22
- Data Flow Control (DFC) 22
  - BIND 260
  - services 260
  - User DFC Interface 261
- DATE (Dedicated Access to X.25 Transport Extension) 29
- DEALLOCATE 74
- Dedicated Access to X.25 Transport Extension (DATE) 29

- defining application programs 201
- defining RCPL fields 22
- defining the network 11
- defining TPF/APPC LUs
  - across a PU 2.1 APPN connection 65
  - across a PU 2.1 LEN connection 64
  - across a PU 5 connection 63
  - across a PU 5 CTC connection 64
  - to a VTAM subsystem 66
  - to the network 63
- DELAY parameter 225
- DFC (Data Flow Control) 22, 260
- DFHTCT 66
- DHASHC macro
  - NCB directory records 187
  - RNHPT hash bucket 183
- diagrams for macro models xx
- domain 3
  - communication across 12
  - multiple 12
  - single 12
- domain resource manager (DRM) 14
- Dumps
  - system error 276
- dynamic LU support
  - ALS resources
    - considerations 200
    - defining 200
    - restrictions 200
  - remote LU resources
    - considerations 199
    - defining 199
    - defining PSV routines 199
    - Dynamic LU user exit 199
    - restrictions 199
- Dynamic LU user exit
  - defining remote LU resources 199

## E

- EBTCBID 71
- EBW000 field 21
- ECHO numbers 151
- enabling a cross-domain link 14
- encryption of data 251
- End Point Control Block (EPCB) 325
- EPCB (End Point Control Block) 325
- error 275
  - detection 275
  - detection and feedback 276
  - from LU attached to 3271 247
  - from LU attached to 3274/3276 247
  - invalid pseudo line number 48
  - path malfunctions 203
  - recovery 277
- Error Detection and Feedback 276
- error recovery 3
- establishing a session 20
- establishing LU-LU sessions 4
- exception reached recovery node (RRN) 251
- exchanging data 3

## F

- Fast Transaction Processing Interface (FTPI) 2, 32
  - command content 34
  - message blocking format 33
  - NPSI considerations for 37
  - traces 42
  - VTAM considerations for 37
- FID5 TH 151
- financial service terminals (3606/3608) 250
- finite state machine 80
- FLUSH 74
- FMH7 100
- forward ANR field 138
- FTPI (Fast Transaction Processing Interface) 2, 32
  - command content 34
  - message blocking format 33
  - NPSI considerations for 37
  - traces 42
  - VTAM considerations for 37
- Function Management Header (FMH) 22
  - defining contents of 26
- function management message router (FMMR) 16

## G

- GATE (General Access to X.25 Transport Extension) 2, 29, 32, 317, 322
- gateway resources 233
- General Access to X.25 Transport Extension (GATE) 2, 29, 32, 317, 322
- generating the side information table 83
  - creating the input file 85
    - ADD statement 85, 87
    - comments 85, 86
    - DESCR statement 85, 87
    - general statement syntax 86
    - input file example 89
    - input statements and ZNSID, comparison 91
    - LOAD statement 85, 87
    - REMOVE statement 85, 89
  - loading the table to TPF 96
  - output from CHQI 93
    - output listing, description of 93
    - output listing, example of 94
    - side information data set 93
  - running CHQI 92
    - sample JCL 93
- GET\_ATTRIBUTES 74
- GET\_TYPE 75

## H

- H-level assembler 69
- half-session to presentation services record (HPR) 69
- half-session, TPF/APPC 59
- Hardware Error Recovery 277
- hardware IPL considerations
  - virtual route resynchronization 231
- header length (HL) field in FMH 22
- hierarchic relationship 3

- high-performance routing (HPR) support
  - alive timer 145
  - ANR labels 128, 136
  - benefits of 127
  - control messages 151
  - diagnostic information
    - PIU trace facility 177, 291
    - sense codes 177
  - flow control
    - ARB pacing 168
    - RTP output queue 170
  - forward ANR field 138
  - heartbeat message 145
  - HPRMT
    - defining 156
    - description of 156
    - displaying 157
  - HPRSAT
    - defining 155
    - description of 155
    - displaying 156
  - input messages
    - reassembling 176
  - installing 176
  - IPL considerations 163
  - links
    - activating 130
    - ANR labels 128, 136
    - displaying 130
    - maximum link size (MLS) 174
    - NCE identifiers 129
    - XID flows 130
  - LU-LU sessions
    - CP-CP sessions 167
    - flows 138
    - session addresses 140
    - starting 135
  - NCE identifiers 129, 138
  - network failures
    - alive timer 145
    - detecting 144
    - heartbeat message 145
    - short request timer 144
  - NLP
    - description of 145
    - FID5 TH 151
    - HPR control messages 151
    - network considerations 152
    - NHDR 147
    - RH 151
    - RU 151
    - THDR 149
  - node types
    - ANR nodes 127
    - mobile RTP nodes 143
    - RTP nodes 127
    - stationary RTP nodes 143
  - output messages
    - building 172
    - retransmitting 174
    - segmenting 175

high-performance routing (HPR) support *(continued)*

- path switch
    - CP-CP sessions 167
    - description of 141
    - path switch timer 143
    - process of 141
    - starting 141
  - reassembly
    - description of 174
  - reverse ANR field 138
  - ROUTE\_SETUP process 136
  - RTP connection resynchronization process
    - description of 163
    - enabling 167
  - RTP connections
    - deactivating 134
    - description of 131
    - displaying 134
    - starting 134, 136
    - states of 131
    - TCIDs 134
  - RTPCB table
    - #RT1RI records 155
    - #RT2RI records 155
    - defining 154
    - description of 152
    - displaying 155
    - initializing 155
    - SNA control blocks, relationship with 157
  - segmentation
    - description of 174
    - THDR chaining 175, 176
  - Select an RTP Connection user exit (URTP) 136
  - selective retransmission
    - description of 173
  - sense codes 177
  - session addresses 140
  - short request timer 144
  - smoothed round trip time 144
  - SRTT 144
  - TCIDs 134
  - THDR chaining
    - description of 175, 176
  - tuning 176
  - URTP user exit 136
  - ZNRTTP DISPLAY command 134, 155
  - ZNRTTP INACT command 134
  - ZNRTTP INITIALIZE command 155
  - ZNRTTP ROUTE command 134
  - ZNRTTP SUMMARY command 134
  - ZNRTTP SWITCH command 141
- high-performance routing message table (HPRMT)
- defining
    - SNAKEY macro 156
  - description of 156
  - displaying
    - ZDDCA command 157
    - ZNRTTP SUMMARY command 157

high-performance routing session address table (HPRSAT)

- defining
  - SNAKEY macro 155
- description of 155
- displaying
  - ZDDCA command 156
- host applications 25
- host node 4
- Host Node Application Considerations 26
- host support
  - multiple 49
- HPO (High Performance Option) 3
- HPR (half-session to presentation services record) 69
- HPR control messages 151
- HPR SOUTC type-A block 172
- HPR SOUTC type-B block 172
- HPR SOUTC type-C block 172

I

- ICCB 67
- ICNOS 68
- IDAW (indirect data address word) 11
- IHPR 69
- IMS 26
  - conventions 28
  - errors 28
- IMS compatibility 19
- IMS Relay Application Considerations 28
- inbound message flow 5
- inbound message flow extensions 317
- inbound message queuing 100
- indirect data address word (IDAW) 11
- initiate (SNA command) 22
- INN (Intermediate Network Node) 29
- input buffer size
  - relationship to bind command processing 32
- input message 47
- Input Message Router Exit 265
- input messages, HPR
  - reassembling 176
- installing TPF/APPC 60
- interface
  - NEF support 49
  - standard 21
  - TPF/NEF/AX.25 45
- Intermediate Network Node (INN) 29
- invalid condition
  - See error
- invalid pseudo line number 48
- IPL considerations (hardware)
  - virtual route resynchronization 231
- ISCB 68
- ISHLL macro 213
- ITPNT 62
- IWBL 68

## K

keyboard, locked 27  
keypoint 2 (CTK2) 67

## L

L/C (Loosely Coupled Facility) 3  
LEID (Logical End-Point Identifier) 2, 6, 45, 46, 260  
    value 46  
LEN 63  
    37x5 Considerations 241  
LEN environment 15  
LEN mode 4  
level 2 networking 16  
line number, interchange address, terminal address  
    (LNIATA) 21, 26  
line numbers, symbolic 48  
links, HPR  
    activating 130  
    displaying  
        ZNAPN command 130  
    maximum link size 174  
    XID flows 130  
LLC (Logical Link Control) 3, 29, 321, 335  
load module 11  
locked keyboard 27  
logging onto an application 22  
Logical End-Point Identifier (LEID) 2, 6, 45, 46, 260  
    value 46  
Logical Link Control (LLC) 3, 29, 321  
logical unit (LU) 19  
    3271  
        special processing 250  
logical unit relationship 3  
Logical Unit Status (LUSTAT) 281  
logical units (LU)  
    recoverable 28  
    type P 28  
Logon Manager Considerations 243  
Loosely Coupled considerations for APPN 205  
loosely coupled considerations for TPF/APPC 63, 103  
    installation tasks for TPF/APPC 105  
    loosely coupled complex example 103  
Loosely Coupled Facility (L/C) 3  
lost input resubmitted 27  
LREAD instruction 26  
LU 6.2  
    architecture 73, 80  
    conversations  
        description of 107  
        pipeline 108  
        shared 110, 112  
        traditional 107  
    installation 60  
    prerequisites 59  
LU registration 211  
LU-LU sessions  
    activating 206  
LU-LU sessions in an APPN network 211

LU-LU sessions, HPR  
    CP-CP sessions 167  
    flows 138  
    starting 135  
LUSTAT (Logical Unit Status) 281  
LUTYPE62 66  
LWRITE instruction 25

## M

macro model diagrams xx  
macro routine (CLXC) 48  
macro to define network 11  
macros  
    ROUT-type 48  
    SEND-type 48  
macros supported by NEF/AX.25 48  
mapped conversation support, TPF/APPC 71  
mapping support  
    AMSG format 42  
MAXBFRU parameter 225  
MAXCCB 61  
MAXPRIM 181  
MAXRVT 179  
MAXSCB 61  
message  
    (non-) recoverable 25  
    data and control 22  
    logical unit status 281  
    multiple segment input 22  
    overflow 21  
    receipt 27  
    recovery and synchronization 17, 249  
    responses 26  
    resubmitted 21  
    segmented 26  
    synchronization 22  
    unsolicited 25  
message blocking format for FTPI 33  
message buffering, outbound 99  
message flow  
    for TPF/APPC 99  
    in an XALCI configuration 56  
    inbound 5  
    outbound 7  
message formats  
    high speed 47  
    specified at system initialization 46  
message number 23  
message processing flow overview  
    inbound 5  
    outbound 7  
message processing flow user extensions 317  
    inbound message 317  
    outbound message 319  
message queuing, inbound 100  
message routing 16  
message traffic multiplexing 33  
Messages  
    outbound messages  
        AMMSG fields 262



- Messages *(continued)*
  - outbound messages *(continued)*
    - PSV 262
    - RCPL 262
  - pacing
    - queue 262
    - ROUTC 262
  - undeliverable messages
    - PSV 262
    - RCAT 262
    - ROUTC 262
    - ROUTC EXIT 262
- messages, unsolicited 47
- Mode Name 239
- models of macro invocations xx
- MSAT 67
- MSGRTA 61
- Multi-Thread Processing 250
- multiple logical units 49
- multiple segment input message 22
- multiple-domain networks 12
- multiplexor channel 45
- MVS/VTAM 11
  - loading NCP 201

## N

- NAU
  - node vs. NAU 3
- NCB
  - See node control block (NCB)
- NCB control record
  - description of 184
- NCB directory records
  - #NCBN4 185
  - #NCBN5 185
  - current NCB directory records 185
  - defining 185
  - description of 185
  - DHASHC macro 187
  - displaying 187
  - example of 185
  - increasing number of 188
  - NCB reconciliation function 187
  - NCB reorganization function 188
  - ordinal number 187
  - ordinal numbers 186
  - reclaiming 187
  - staged NCB directory records 185
- NCB initialization function
  - description of 187
  - ZNNCB command 187
- NCB reconciliation function
  - 381-byte long-term pool file NCB records 187
  - description of 187
  - NCB directory records 187
  - procedure for 188
  - running 188
  - ZNNCB RECON command 188
- NCB reorganization function
  - description of 188
- NCB reorganization function *(continued)*
  - ending 190
  - procedure for 189
  - process 188
  - running 189
  - ZNNCB REORG command 190
- NCE identifier 129
- NCE identifiers 138
- NCP Packet Switching Interface (NPSI) 2, 3, 30, 32
- NCP Slowdown 277
- NEF (Network Extension Facility) 2, 45
  - application logical unit
    - not in RCAT 46
  - editing by NEF 47
  - multiple host support 49
  - transaction analysis (TA) program 49
  - translation 47
- NEFx terminology 46
- network 1
  - address 19
  - control 11
  - name 19
- network addressable unit (NAU) 3
  - control point (CP)
    - primary, secondary 3
  - logical unit (LU) 3
  - physical unit (PU) 3
  - system service control point (SSCP) 3
- network connection endpoint (NCE) identifiers 129, 138
- network control program (NCP)
  - modifying input deck 233
  - slowdown 277
- network definition 233
- Network Extension Facility (NEF) 2, 45, 47
  - application logical unit
    - not in RCAT 46
  - editing by NEF 47
  - multiple host support 49
  - transaction analysis (TA) program 49
  - translation 47
- network failures, HPR
  - alive timer 145
  - detecting 144
  - heartbeat message 145
  - short request timer 144
- network generation 250
- Network Interface Adapter (NIA) 29
- network layer header (NHDR) 147
- network layer packet (NLP)
  - data
    - FID5 TH 151
    - RH 151
    - RU 151
  - description of 145
  - HPR control messages 151
  - network considerations 152
  - NHDR 147
  - THDR 149
- network resource 3
- Network Services Application Interfaces 213

- NHDR 147
- NIA (Network Interface Adapter) 29
- node
  - host or non-host 4
- node control block (NCB) 19
- node control block (NCB) records
  - 381-byte fixed file
    - description of 184
  - 381-byte long-term file 184
    - description of 184
    - NCB directory records 184
    - NCB slots 184, 185
  - description of 183
  - displaying 187
  - initializing 187
  - NCB control record 184
  - NCB directory records 185
  - NCB initialization function 187
  - NCB reconciliation function 187
  - NCB slots
    - defining for an application 184, 185
    - description of 184, 186
  - performance 190
  - reclaiming 187
- node types, HPR
  - ANR nodes 127
  - mobile RTP nodes 143
  - RTP nodes 127
  - stationary RTP nodes 143
- node vs. NAU 3
- non-host node 4
- Non-SNA 3270 Application Considerations 26
- NPSI (NCP Packet Switching Interface) 2, 3, 30, 32
- NPSI GATE/FTPI 2
- NSPI considerations for FTPI (Fast Transaction Processing Interface) 37
- null RU 27, 29
- NUMALS 179

## O

- online merge function 196
- OSTG 61
- OSTG program
  - considerations 194
  - defining SNA resources 193
  - IODEV macro 194
  - loading resource definitions 194
    - SNA dynamic load function 195
    - SNA fallback load function 196
    - SNA fresh load function 194
- outbound message buffering 99
- outbound message flow 7
- outbound message flow extensions 319
  - ROUTC exit 319
- output message transmission (OMT) 28, 281
- output Messages 48
- output messages, HPR
  - building 172
  - HPR SOUTC type-A block 172
  - HPR SOUTC type-B block 172

- output messages, HPR (*continued*)
  - HPR SOUTC type-C block 172
  - retransmitting 174
  - segmenting 175

## P

- pacing considerations
  - for AX.25 240
- pacing in SNA 248
- Packet Assembler/Disassembler (PAD) 29
- Packet Switched Data Network (PSDN) 29
- PAD (Packet Assembler/Disassembler) 29
- PARACOS 62
- partner work block for TPF/APPC 69
- path information unit (PIU) 46
  - See PIUPT0 utility
- path switch
  - CP-CP sessions 167
  - description of 141
  - path switch timer 143
  - process of 141
  - starting
    - automatically 141
    - ZNRTTP SWITCH command 141
- path switch timer
  - defining
    - SNAKEY macro 143
    - ZNKEY command 143
  - description of 143
- PCID 238
- permanent virtual circuits (PVC) 29
- PIU trace facility
  - description 291
  - maximum tape queue length
    - defining 293
- PIU trace table
  - description 291
  - displaying offline 301, 307, 309
  - displaying online 295, 296, 297
  - including in system error dumps 313
  - size of, defining 291
  - writing to real-time tape, starting 293
  - writing to real-time tape, stopping 293
- PIUPRT report
  - compacted 307
  - creating 302
  - description 301
  - examples 307, 309
  - formatted 309
- PIUPRT utility
  - description 301
  - PARM= statement 303
  - PIUPRT report 307, 309
  - return codes 312
  - sample JCL 302
- RU
  - specifying amount to trace 293
- starting
  - description 291
  - procedure 292



- PIU trace facility (*continued*)
  - status of
    - displaying 294
    - examples 294
  - stopping 292
  - traces
    - defining 291
    - displaying 294
    - starting 291
    - stopping 292
- PIU trace table
  - compacted display
    - examples 297, 307
    - offline 307
    - online 296
    - PIUPRT utility 307
    - ZNPIU command 296
  - description 291
  - displaying offline
    - compacted report 307
    - examples 307, 309
    - formatted report 309
    - PIUPRT utility 301
  - displaying online
    - compacted display 296
    - examples 297, 300
    - formatted display 297
    - ZNPIU command 295
  - formatted display
    - examples 300, 309
    - offline 309
    - online 297
    - PIUPRT utility 309
    - RH indicators 312
    - ZNPIU command 297
  - including in system error dumps 313
  - maximum tape queue length
    - defining 293
  - size of, defining 291
  - writing to real-time tape
    - starting 293
    - stopping 293
- PIUPRT utility
  - description 301
  - PARM= statement
    - description 303
    - examples 306
  - PIUPRT report
    - compacted report 307
    - creating 302
    - defining 303
    - description 301
    - examples 307, 309
    - formatted report 309
  - return codes
    - description 312
  - sample JCL 302
- PLU (primary logical unit) 3
- POST\_ON\_RECEIPT 75
- PREPARE\_TO\_RECEIVE 75
- presentation handling 6

- presentation services, TPF/APPC 59, 69
- primary host node 4
- primary logical unit (PLU) 3
- prime CRAS 49
- priming (CTC) 222
- process selection vector (PSV) 253
  - input message processing 253
  - output message processing 254
- Process Selection Vector (PSV) 22, 29, 30
  - NPSI/FTPI command type 51
  - NPSI/GATE commands 51
- proctor handling 6
- Programming Considerations for Session Initiation 22
- protocols, SNA 26
- PSDN (Packet Switched Data Network) 29
- pseudo line number
  - user-assigned 49
- pseudo-address
  - in NEF 48
- PSV
  - interface 255
  - output message queueing 259
  - processing
    - input message 253
    - NPSI/FTPI command type 51
    - NPSI/GATE commands 329
    - output message 254
  - routines 260
    - defining 259
    - names 259
    - NPSI/FTPI command type 51
- PSV (process selection vector) 253
- PSV (Process Selection Vector) 22, 29, 30
- PU 2.1
  - 37x5 considerations 240
  - APPN mode 4
  - LEN mode 4
- PU\_2.1 237
  - peripheral node 239
  - Route Selection 239
- PU\_2.1 Environment 237

## Q

- QCE (Queue Control Element) 269
- Queue Control Element (QCE) 269
- Queue Manager
  - DEQUEUE 269, 272
  - detailed return codes 273
  - ENQUEUE 269, 271
  - general return codes 273
  - GET 271
  - PURGE 269, 272
  - WASH 269, 272

## R

- railroad tracks xx
- rapid transport protocol (RTP) connections
  - alive timer 145

- rapid transport protocol (RTP) connections *(continued)*
  - deactivating
    - automatically 134
    - ZNRTIP INACT command 134
  - description of 131
  - displaying
    - ZNDLU command 134
    - ZNMON command 134
    - ZNRTIP DISPLAY command 134
    - ZNRTIP ROUTE command 134
    - ZNRTIP SUMMARY command 134
  - failures
    - alive timer 145
    - detecting 144
    - heartbeat message 145
    - short request timer 144
  - heartbeat message 145
  - IPL considerations 163
  - path switch
    - description of 141
    - path switch timer 143
    - process of 141
    - starting 141
  - resynchronizing 163
  - ROUTE\_SETUP process 136
  - Select an RTP Connection user exit (URTP) 136
  - short request timer 144
  - smoothed round trip time (SRTT) 144
  - SRTT 144
  - starting 134, 136
  - states of 131
  - TCIDs 134
  - URTP user exit 136
- rapid transport protocol (RTP) nodes 127, 143
- rapid transport protocol control block table
  - #RT1RI records 155
  - #RT2RI records 155
  - defining
    - SNAKEY macro 154
  - description of 152
  - displaying
    - ZDDCA command 155
    - ZNRTIP DISPLAY command 155
  - initializing
    - ZNRTIP INITIALIZE command 155
  - SNA control blocks
    - relationship with 157
- RC0PL data macro 22
- read buffers (4K) 225
- read-flags (SMSCRF) field 26
- read-type (SMSCRT) field 26
- real-time trace (RTT) 275
- reassembly, HPR
  - description of 174
- RECEIVE\_AND\_WAIT 76
- Recoverable and Non-recoverable Messages 247
- reentering a message 247
- Registration of LUs in APPN network 211
- release input signal 27
- Reliability and Serviceability 276
- report, configuration 49
- request recovery (SNA command) 23
- request unit (RU) 46
  - chaining for NPSI 31
  - sizes 31
- REQUEST\_TO\_SEND 76
- request-response header (RH) 247
- requesting a cross-domain session 14
- requirements
  - CMC ownership of resources 46
- resource
  - activating and deactivating
    - implicitly 201
  - active and inactive 201
  - gateway 233
- resource identifier (RID) 21, 45
- resource IDs
  - assigning to RVT entries 180
- resource manager, TPF/APPC 59
- resource name hash control table (RNHCT)
  - description of 181
  - displaying 183
  - example of 182
- resource name hash entry table (RNHET)
  - description of 181
  - displaying 183
  - example of 182
  - RVT available list 181
  - RVT termination list 181
  - synonym chain 181
- resource name hash prime table (RNHPT)
  - description of 181
  - displaying 183
  - example of 182
  - hash buckets 181
- resource vector table (RVT) 71
  - ALS section 179
  - defining 179
  - delimiters 181
  - description of 179
  - example of 180
  - LU section 179
  - non-LU section 179
  - organization of 179
  - resource IDs (RIDs) 180
  - resource name hash control table (RNHCT) 181
  - resource name hash entry table (RNHET) 181
  - resource name hash prime table (RNHPT) 181
  - RNHET synonym chain 181
  - RVT available list 181
  - RVT termination list 181
  - spare entries 180
- Resource Vector Table (RVT) 48
- resources
  - defining 193
- Response Protocol
  - 3614/3624 LU 251
  - for device types 247
- response protocols/error handling
  - interface 262
  - LUSTAT sense 262
  - sense codes 262

- response types 247
- restart, system 101
- resubmitted message 21
- retransmitting messages 247
- Retrieving NCB and SPA Data Records (CSNB) 287
- reverse ANR field 138
- RH indicators
  - description 312
- RID and RVT conversions 287
- RISC System/6000 2, 244
- RNHET synonym chain
  - description of 181
- RNHPT hash buckets
  - description of 181
  - DHASHC macro 183
- ROUT-type macro 48
- ROUTC 100
- ROUTC exit 319
- ROUTC macro 21, 27, 250, 251
- ROUTE\_SETUP process 136
- ROUTER processing 6
- routing control application table (RCAT) 22
- routing control block (RCB) 19
- routing control parameter list (RCPL) 21, 27
  - destination field 21
  - indicators 22
  - origin field 21
  - passed between TPF and IMS 28
- RSC 61
- RTP connection resynchronization process
  - description of 163
  - enabling
    - SNAKEY macro 167
    - ZNKEY command 167
- RTP nodes 127, 143
- RTP output queue
  - description of 170
- RU (request unit)
  - chaining for NPSI 31
  - sizes 31
- RVT (resource vector table) 71
- RVT available list
  - description of 181
- RVT termination list
  - description of 181

## S

- SCB (session control block) 21, 68
- scratch pad area (SPA) 19
- SDAT (Symbolic Device Address Table) 233
- SDLC support 49
- secondary host node 4
- secondary logical unit
  - SLU THREAD 19
- secondary logical unit (SLU) 3
- segmentation, HPR
  - description of 174
  - THDR chaining 175, 176
- segmented messages 248
- Select an RTP Connection user exit (URTP) 136

- selective retransmission, HPR
  - description of 173
- SEND\_DATA 76
- SEND\_ERROR 77
- SEND-type macro 48
- sequence number 23
- service LU, TPF/APPC 103
- service transaction program, TPF/APPC 103
- session 3
- session activation 20
- session activation flow 345
  - APPN 374
    - APPN-Subarea 384
  - CDRM-CDRM session 345
  - CP-CP sessions 374
  - FMMR-FMMR session 352
  - host-node SLU session 350
  - LEN 345, 359
  - LU 6.2 session 359
  - LU-LU sessions 375
  - printer sharing 380
  - PU 2.1 LEN 345, 359
  - PU 5 345
    - PU 5 LU 6.2 session (TPF PLU) 356
    - PU 5 LU 6.2 session (TPF SLU) 358
  - Subarea-APPN 384
  - TPF APPL (PLU) and remote APPL (SLU) 347
  - TPF APPL and remote 3270 354
- session addresses 140
- session awareness 215
- session control block (SCB) 21, 68
- Session Control via VTAM SSCP 46
- session deactivation flow
  - APPN 383
    - CDRM-CDRM session 360
    - FMMR-FMMR session 369
    - host-node SLU session 366
    - LU 6.2 session 371
    - PU 2.1 360
    - PU 5 360
      - TPF APPL (PLU) and remote APPL (SLU) 363
- session ended notification 216
  - starting 217
- Session Identifiers - SID 237
- Session Management Request Services
  - session resynchronization 213, 214
  - session termination 213, 215
  - starting 213
- session manager, TPF/APPC 59
- session resynchronization 213
  - requesting 214
- session started notification 216
  - starting 216
- Session Status Awareness Services
  - activating 215
  - CTCP 327
    - session ended notification 215, 217
    - session started notification 215, 216
- session termination 99, 213
  - requesting 215
- set and test sequence numbers (SNA command) 23

- short request timer
  - description of 144
  - smoothed round trip time (SRTT) 144
- SID - session identifier 237
- side information table 62, 71, 83
  - generating
    - creating the input file 85
    - loading the table to TPF 96
    - output from CHQI 93
    - running CHQI 92
- SIGNAL 281
- simultaneous message processing 250
- single-domain networks 12
- Single-Segment Messages 247
- SINGMODE 62
- slowdown
  - channel-to-channel (CTC) 277
- slowdown, NCP 277
- SLU
  - SLU THREAD 19
- SLU (secondary logical unit) 3
- SLU threads
  - and TPF/APPC 61
- smoothed round trip time (SRTT) 144
- SNA
  - data records 45
- SNA 3270 Application Considerations 26
- SNA command flow 345
- SNA commands 22
- SNA communications
  - NCB records 183
  - RVT 179
- SNA data transfer
  - channel-to-channel (CTC) characteristics 11
  - NCP characteristics 11
- SNA dynamic load function
  - description 195
  - displaying status information 198
  - inactive processors 196
  - procedure 196
  - restrictions 196
  - ZNOPL LOAD command 195, 196
  - ZNOPL MERGE command 195, 196
  - ZNOPL UPDATE command 195, 196
- SNA fallback function
  - description 196
  - displaying status information 198
  - using dynamic load function 197
  - using fresh load function 197
  - ZNOPL FALLBACK command 197
  - ZNOPL MERGE command 197
- SNA fresh load function
  - considerations 194
  - description 194
  - displaying status information 198
  - forcing 195
  - procedure 194
  - ZNOPL BUILD command 195
  - ZNOPL LOAD command 195
- SNA I/O trace facility 275
- SNA network Interconnection (SNI)
  - Considerations 233
- SNA online merge function 196
- SNA protocols 26
- SNA resources
  - changing definitions
    - ZNDYN CHANGE command 199
  - defining
    - ALS resources 193
    - CDRM resources 193
    - CTC resources 193
    - local resources 193
    - NCP resources 193
    - OSTG program 193
    - remote LU resources 193
    - shared printers 193
    - SSCP resources 193
    - TPF applications 193
    - ZNDYN ADD command 198
  - dynamic load function 195
  - dynamic LU support 199
  - fallback function 196
  - fresh load function 194
  - tracing 291
- SNAKEY 61
- SNI/TPF restrictions 234
- SOUTC macro 250
- SOUTC macro service routine 48
- SPA for a dynamic LU
  - allocating 190
  - retrieving 190
- spare RVT entries
  - description of 180
  - RVT available list 181
- SRTT 144
- standard communications interface 21
- start data traffic (SNA command) 22, 247
- Starting and Stopping Application Programs 201
- subarea 234
- subarea address table (SAT) 234
- supported verb functions, TPF/APPC 72
- switched virtual circuits (SVC) 29
- Symbolic Device Address Table (SDAT) 233
- symbolic line number 48
  - invalid 49
- SYNC numbers 151
- synchronization check characters 46
- Synchronizing Messages with Sequence Numbers 23
- synchronous data link (SDLC) 12, 26
- synonym chain
  - description of 181
- syntax diagrams xx
- system communication configuration table (CTKE) 48
- system communication keypoint records 48
- system initialization program (SIP) 21, 26
- system message 25
- system message processor (CSMP) 49
- System network Interconnect (SNI) Considerations
  - modifying NCP input deck 233
  - SNI NCP generation deck 233
  - with APPN considerations 234

- System network Interconnect (SNI) Considerations  
(*continued*)
  - with channel-to-channel (CTC) considerations 234
- system recovery table (SRT) 27
- system restart
  - and TPF/APPC 101
- system services control point (SSCP) 14, 46

## T

- TCB ID (transaction control block identifier) 67
- TCIDs 134
- terminal address table (WGTA) 45, 48
- terminal-type values 26
- termination, session 99
- TEST 77
- TG (transmission group) 223
- THDR chaining
  - description of 175, 176
- tightly coupled considerations for TPF/APPC 102
- TP (transmission priority) 227, 229
- TPALLOC 62
- TPF as Host Node SLU 19
- TPF Mapping Support 42
- TPF SNA overview 1
- TPF/APPC (TPF Advanced Program-to-Program Communications) 59
  - APPC overview 59
  - architecture comparison 73
    - ALLOCATE 73
    - CONFIRM 73
    - CONFIRMED 73
    - DEALLOCATE 74
    - FLUSH 74
    - GET\_ATTRIBUTES 74
    - GET\_TYPE 75
    - POST\_ON\_RECEIPT 75
    - PREPARE\_TO\_RECEIVE 75
    - RECEIVE\_AND\_WAIT 76
    - REQUEST\_TO\_SEND 76
    - SEND\_DATA 76
    - SEND\_ERROR 77
    - TEST 77
    - WAIT 77
  - ATTACH interface 69
  - change number of sessions 78
  - change number of sessions work block 68
  - components 59
  - control blocks 67
  - conversation control block (CCB) 67
  - conversation verbs 71
    - basic conversation verbs 72
    - mapped conversation verbs 71
    - type-independent verbs 72
  - defining LUs to a VTAM subsystem 66
  - defining LUs to the network 63
  - finite state machine 80
  - half-session to presentation services record (HPR) 69
  - inbound message queuing 100
  - installation checklist 60

- TPF/APPC (TPF Advanced Program-to-Program Communications) (*continued*)
  - loosely coupled considerations 63, 103
  - mapped conversation support 62, 71
    - side information table 62, 71
  - side information table offline program (CHQI) 83
  - message flow 99
  - partner work block 69
  - presentation services 69
  - resource manager 98
  - sample transaction programs 113
  - service LU 103
  - service transaction program 103
  - session control block (SCB) 68
  - session manager 99
  - side information table 62
  - SLU threads 61
  - system restart 101
  - tightly coupled considerations 102
  - transaction program name table 62, 70
  - user exits 107
  - work blocks 68
- TPF/APPC components
  - half-session 59
  - presentation services 59
  - resource manager 59
  - session manager 59
- TPF/APPC service LU 103
- TPF/APPC service transaction program 103
- TPF/NEF/AX.25 host interfaces 45
- TPF/SNA structures
  - NCB records 183
  - RVT 179
- TPNT (transaction program name table) 62, 70
- TPPCC macros 73
- TPRECV 62
- TPWAIT 62
- trace functions 275
- trace, realtime
  - agent 47
  - nodename 47
- traces (for Fast Transaction Processing Interface) 42
- tracing and testing in TPF 275
- transaction control block identifier (TCB ID) 67
- transaction program name table (TPNT) 62, 70
- transaction programs
  - design considerations 107
  - samples of
    - requested TPF transaction program 116, 121
    - requesting TPF transaction program 114, 117
- translation by NEF 47
- Transmission Control (TC)
  - services 260
- transmission group (TG) 223
- transmission priority (TP) 227, 229
- transport connection identifiers (TCIDs) 134
- transport header (THDR)
  - description of 149
  - ECHO numbers 151
  - optional segments 150
  - SYNC numbers 151

- TRMEQ macro 26
- tuning SNA network performance 32
- type P SLU support 28
- type-A block, HPR SOUTC 172
- type-B block, HPR SOUTC 172
- type-C block, HPR SOUTC 172

## U

- UNBIND 99
- Unsolicited Messages
  - notification methods 250
- unsolicited messages/requests 47
- URTP user exit 136
- User DFC Interface
  - RCPL 261
  - Request / Response Header (RH) 261
  - Sequence Numbers 261
- user exits for TPF/APPC 107
- user terminal control blocks 325

## V

- VARY command 99
- VC (virtual circuits) 29
- VCCB (Virtual Circuit Control Block) 324
- verb functions, TPF/APPC 72
- Virtual Circuit Control Block (VCCB) 324
- virtual circuits (VC)
  - Permanent Virtual Circuits (PVC) 29
  - switched virtual circuits (SVC) 29
- virtual route (VR) activation 228
- virtual route assignment for CTC
  - for LU-LU session for CTC 228
- virtual route resynchronization 231
- Virtual Telecommunications Access Method (VTAM) 46
- VR (virtual route) activation 228
- VTAM (Virtual Telecommunications Access Method) 46, 233
- VTAM Considerations 243
  - for FTPI (Fast Transaction Processing Interface) 37
- VTAM DELAY operand 225

## W

- WAIT 77
- work blocks for TPF/APPC 68
- write buffers (4K) 225
- write-flags (SMSCWF) field 25
- write-type (SMSCWT) field 25

## X

- X.25
  - command message flows 328
  - data message flows 335
  - interface 3
  - link control block 324
  - message 30
  - network control block 324

- X.25 (*continued*)
  - protocols 253
- X.25 ALC interface (XALCI) 2
  - configuration 56
  - for switched or permanent virtual circuits 50
  - header formats 54
  - message flow 56
- X.25 gateway considerations 53
- X.25 link control block (XLCB) 324
- X.25 network control block (XNCB) 324
- XALCI (X.25 ALC interface) 2
  - configuration 56
  - for switched or permanent virtual circuits 50
  - header formats 54
  - message flow 56
- XLCB (X.25 link control block) 324
- XNCB (X.25 network control block) 324

## Z

- ZNCNS CHANGE 99
- ZNCNS RESET 99
- ZNDYN ADD command
  - considerations 198
  - defining SNA resources 198
  - IODEV macro 198
  - restrictions 198
- ZNDYN CHANGE command 199
- ZNDYN DISPLAY command 183
- ZNETW command 201
- ZNETW INACT 99
- ZNKEY command 201
- ZNNCB command 187
- ZNNCB DISPLAY command 187
- ZNNCB RECON command 188
- ZNNCB REORG command 190
- ZNOPL BUILD command 195
- ZNOPL FALLBACK command 197
- ZNOPL LOAD command 195, 196
- ZNOPL MERGE command 195, 196, 197
- ZNOPL STATUS command 195, 196, 197, 198
- ZNOPL UPDATE command 195, 196
- ZNPOL command 201
- ZNRTP DISPLAY command 134, 155
- ZNRTP INACT command 134
- ZNRTP INITITALIZE command 155
- ZNRTP ROUTE command 134
- ZNRTP SUMMARY command 134, 157
- ZNRTP SWITCH command 141
- ZNSID command 91
- ZROUT command 201
- ZSNDU command 250
- Zxxxx commands 47, 49





File Number: S370/30XX-30  
Program Number: 5748-T14



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SH31-0168-06

